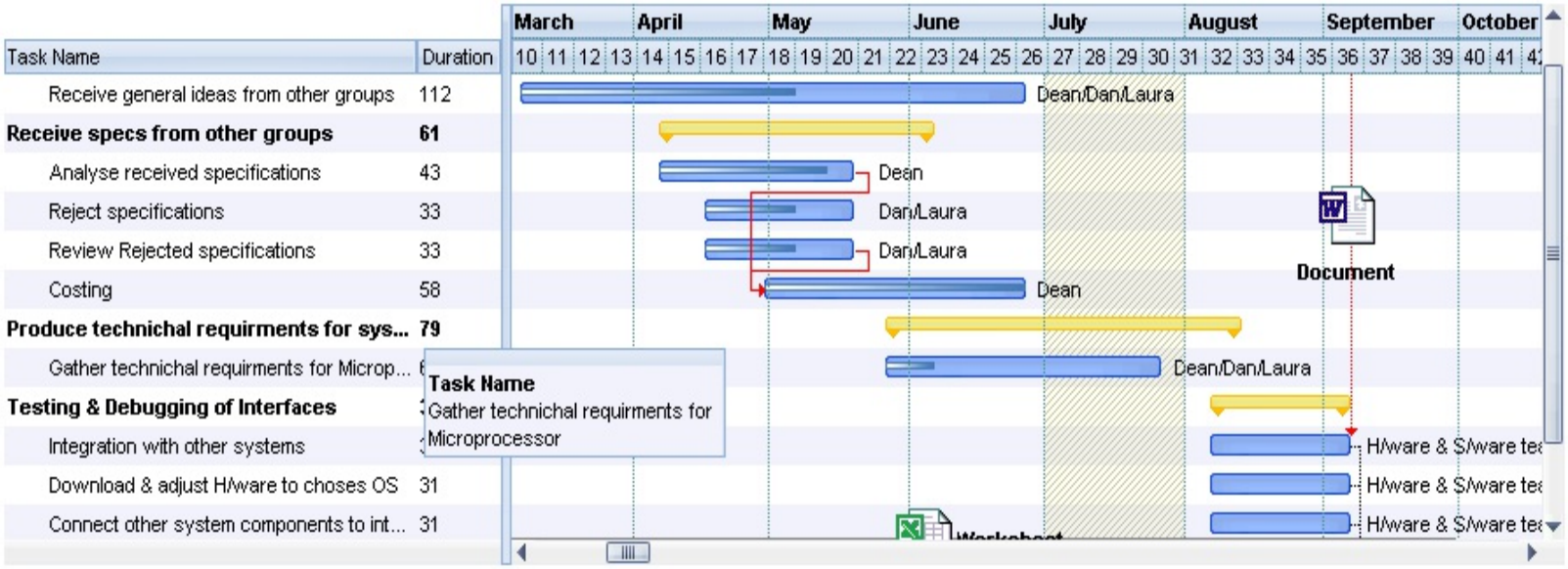
 ExG2antt

The Exontrol's ExG2antt component is our approach to create timeline charts (also known as Gantt charts). Gantt chart is a time-phased graphic display of activity durations. Activities are listed with other tabular information on the left side with time intervals over the bars. Activity durations are shown in the form of horizontal bars. The ex(G)rid-ex(G)antt, shortly exG2antt, combines the ExGrid and ExGantt components in a standalone component. The exG2antt component shows timeline charts on a multi-columns tree control. The ExG2antt component lets the user changes its visual appearance using **skins**, each one providing an additional visual experience that enhances viewing pleasure. Skins are relatively easy to build and put on any part of the control.

Features include:

- **Print** and Print Preview support
- **ADO** and **DAO** support, **DataSets** for /NET
- **Skinable Interface** support (ability to apply a skin to the parts of the control)
- **Custom Row Designer** (Have your rows display however you want with the control row layout capabilities)
- Ability to save/load the control's data to/from **XML** files
- **EMF Format** support (Ability to save the control's content to Enhanced Metafile (EMF) file, and so to any BMP, JPG, GIF or PNG formats)
- **Overview Layout/Map** support
- **Histogram** support
- **Conditional Format** support
- **Computed Fields** support
- **Filter** support
  - **Filter-Prompt** support, allows you to filter the items as you type while the filter bar is always visible on the bottom part of the list area.
  - **Filter-On-Type** support**.** Ability to filter items by a column, as you type.
- **Editors** support
- **Scroll Line by Line** support, for smoothing scroll items with different heights
- **Multiple Columns**
- **Sorting by Single or Multiple Column** support
- **Locked/Fixed columns** support
- **Split, Merge cells** support
- **Single/ Multiple Lines/Levels/Expandable Header** support
- **Regional and Language Options** support to display date and times.
- Alternative HTML labels support for best fit in the level's time unit.
- Ability to show the control's element from right-to-left for Hebrew, Arabic and other **RTL** languages

- Ability to specify **multiple levels**, using custom built-in HTML format for each of level
- Ability to insert **hyperlinks** anywhere in the cells, bars or links
- Zoom and Scale support (including at run-time too)
- Ability to **enlarge** or magnify (zoom-in,zoom-out) the entire chart, by dragging the header or resizing it using the middle mouse button, with or without re-scalling the chart.
- Ability to **magnify** only a portion of the chart, so the rest of the chart stay unchanged, ie shows hours of selected day(s).
- Ability to show the position of **current date-time**, using different styles including EBN files
- Ability to highlight or mark different **date-time zones**, using different colors, EBNs, patterns, multiple HTML captions
- Nonworking support
  - **Nonworking Days, Nonworking Hours** support
  - Ability to specify non-working parts for any item
  - Ability to specify bars that are treated as non-working parts of the items
- Draw lines or links between bars support
- **Semi-Transparent and Opaque Bars** support
- Ability to **summarize** the bars, so they get updated as soon as child bars are moved or resized
- Ability to **group** bars, preserve the length of the bars, fixing (or within a specified range) the distance between bars
- Ability to show the position of current date-time, using different styles including EBN files
- Ability to select bars and links at runtime
- Ability to move and resizes the bars on the fly
- Ability to customize the **overlaid bars** using different offset, transparency, colors, patterns, shapes, and so on
- Ability to link bars using the mouse
- Ability to display pictures or HTML text on any link
- Ability to assign multiple bars to a single item
- Predefined list of bars like task, milestone and so on
- Ability to define your own type of bars using custom shapes and patterns
- Ability to define the starting and ending corners from icons
- Multi-Lines Built-In HTML Tooltip support
- **Undo/Redo** support
- Precedence Diagramming Method (**PDM** or scheduling activities in a project plan)
- Ability to associate a bar/date with fully customizable, movable **boxes or notes**, including HTML text, images, links and so on



How to get support?

To keep your business applications running, you need support you can count on.

Here are few hints what to do when you're stuck on your programming:

- Check out the samples - they are here to provide some quick info on how things should be done
- Check out the how-to questions using the eXHelper tool
- Check out the help - includes documentation for each method, property or event
- Check out if you have the latest version, and if you don't have it send an update request here.
- Submit your problem(question) here.

Don't forget that you can contact our development team if you have ideas or requests for new components, by sending us an e-mail at support@exontrol.com (please include the name of the product in the subject, ex: exgrid) . We're sure our team of developers will try to find a way to make you happy - and us too, since we helped.

Regards,
Exontrol Development Team

https://www.exontrol.com

How to start?

The following screen shot, shows a general idea how parts and objects of the control are arranged:

click to enlarge

The following steps shows you progressively how to start programming the Exontrol's ExG2antt component:

- **Load / Save Data**. The control provides several ways to serialize your data, as listed:
  - LoadXML / SaveXML methods, to load / save data using XML format.
  - DataSource property, to load / update / save data from a table, query, dataset and so on.
  - GetItems / PutItems methods, to load / save data from a/to safe array of data.

  For instance,

```
With G2antt1
   .LoadXML "https://www.exontrol.net/testing.xml"
End With
```

loads control's data from specified URL.

- **Chart**. The control's chart displays tasks based on the time-unit scale, using a multiple-levels header.

  - UnitScale property, determines the base time-unit scale to be displayed on the chart.
  - Label property, indicates the predefined format of the level's label for a specified unit, to be shown on the chart.
  - LevelCount property, specifies the number of levels to be shown on the chart's header.

  For instance,

```
With G2antt1
   With .Chart
      .LevelCount = 2
      .UnitScale = exDay
   End With
End With
```

  specifies that the chart's header should display two levels, and the base time-unit scale to be day.

- **Bars**. The chart's bars collection holds the types of the bars the chart can display. By default, it includes Task, Milestone, Summary, Project Summary, ...

  - Add method, adds a new type of bar, including a combination of any of already predefined bars to display split or/and progress bars.
  - Copy property, clones an already predefined bar.

  For instance,

```
With G2antt1
   .Chart.Bars.Add("Task%Progress").Shortcut = "TProgress"
End With
```

  defines a new task bar to display a progress bar inside. See **Item-Bars**, to see

how you can add tasks/bars to the control's chart panel.

- **Links**. See **Item-Links**, to see how you can add links between tasks/bars to the control's chart panel.
- **Notes**. See **Item-Notes**, to see how you can add notes on the control's chart panel.
- **Columns**. The control supports multiple columns, so always you can add / remove / move / hide any column

  - Add method, adds a new column.
  - ExpandColumns property specifies the columns to be shown/hidden when the column is expanded or collapsed.

  For instance,

```
With G2antt1
  With .Columns.Add("Check")
    .Position = 0
    .Def(exCellHasCheckBox) = True
  End With
End With
```

  adds a new column that displays check-boxes, and that's the first visible column.

- **Editors**. Any cell / column of the control supports built-in editors, that let user edits data

  - EditType method, specifies the built-in to be assigned to a cell or column.
  - Editor property, gets access to the column's built-in editor
  - CellEditorVisible property specifies the built-in editor for a particular cell.

  For instance,

```
With G2antt1
  With .Columns.Add("Date")
    .Editor.EditType = DateType
  End With
End With
```

  adds a new column that displays and edits column's data as date type.

- **Items**. Any item can hold a collection of child items. Any item is divided in cells, once cell for each column in the control.

- AddItem method, adds a new item.
- InsertItem method, inserts a child item
- InsertControlItem method, inserts a child item that hosts another control inside.

For instance,

```
With G2antt1
   With .Items
      .AddItem "new item"
   End With
End With
```

adds a new item.

- **Cells**. An item contains a collection of cells, one cell for each column in the control. Any cell can be split or merge with one or more neighbor cells.

  - CellValue property, specifies the cell's value.

  For instance,

```
With G2antt1
   With .Items
      h = .InsertItem(.FocusItem,"","item 1.1")
      .CellValue(h,1) = "item 1.2"
      .CellValue(h,2) = "item 1.3"
      .ExpandItem(.FocusItem) = True
   End With
End With
```

  adds a new child item of the focused item, and fills the cell's value for the second and third column.

- **Item-Bars**. Any item can display one or more tasks/bars.

  - AddBar method, adds a new bar of specified type, giving its time interval.
  - ItemBar property, updates properties of specified bar, like caption, effort, and so on
  - DefineSummaryBars method, defines child-bars of a summary bar.

  For instance,

```
With G2antt1
   With .Items
      .AddBar .FocusItem,"Task",#4/1/2006#,#4/14/2006#,"new"
   End With
End With
```

adds a new task to the focus item, with the key "new".

- **Item-Links**. Any two-bars of the chart, can be linked.
  - AddLink method, links two bars.
  - Link property, gets access to the link's properties

  For instance,

```
With G2antt1
   With .Items
      .AddBar .FocusItem,"Task",#4/1/2006#,#4/14/2006#,"A"
      .AddBar .FocusItem,"Task",#4/18/2006#,#4/30/2006#,"B"
      .AddLink "AB",.FocusItem,"A",.FocusItem,"B"
   End With
End With
```

  adds two linked bars A and B in the same item.

- **Item-Notes**. The chart panel of the control supports notes, that can be associated with any date or bar in the chart.
  - Add method, associates a note to a date or task/bar.

  For instance,

```
With G2antt1
   With .Chart.Notes
      With
.Add("D1",G2antt1.Items.FirstVisibleItem,G2antt1.Chart.FirstVisibleDate,"Date
<br><%dd%>/<%mm%><br><b><%yyyy%></b>")
         .PartCanMove(exNoteEnd) = True
         .PartVOffset(exNoteEnd) = 20
         .PartHOffset(exNoteEnd) = 20
```

```
        End With
    End With
End With
```

adds a note associated with first visible date.

constants AlignmentEnum

The Column object uses the AlignmentEnum enumeration to align a column. See the Alignment property of the Column and Alignment properties related.

Name	Value	Description
LeftAlignment	0	The source is left aligned.
CenterAlignment	1	The source is centered.
RightAlignment	2	The source is right aligned.
exHOutside	16	The caption is displayed outside of the source.

constants AllowSplitPaneEnum

The AllowSplitPaneEnum type specifies the number of splitting panels the control's chart supports. The AllowSplitPane property specifies whether the chart panel supports splitting. Once the AllowSplitPane property is set, the user can click the lower-right split bar, and drag to a new position to add a new split to the current chart. The AllowSplitPaneEnum type supports the following values.

Name	Value	Description
exNoSplitPane	0	No split panel.
exAllowOneSplitPane	1	The chart allows adding one splitting panel.
exAllowTwoSplitPane	2	The chart allows adding two splitting panel.

constants AppearanceEnum

The AppearanceEnum enumeration is used to specify the appearance of the control's header bar. See also the HeaderAppearance property.

Name	Value	Description
None2	0	No border (while the HeaderAppearance property is 0, the user can't resize the columns while cursor hovers the control's header bar)
Flat	1	Flat border
Sunken	2	Sunken border
Raised	3	Raised border
Etched	4	Etched border
Bump	5	Bump border

constants ArrowHandleEnum

The ArrorHandleEnum expression specifies the options for exLeftArrow, exRightArrow, exDownArrow or exUpArrow values when the Option property is used.

Name	Value	Description
exHandleEditor	0	The editor handles the arrow key. The key moves the cursor, if exists, inside the edit control. If the editor displays a caret, the F2 key selects or unselects the entire text.
exHandleControl	-1	The control handles the arrow key. The key moves the focus to a new cell. If the editor displays a caret, the F2 key selects or unselects the entire text. If the entire text is selected the key moves the focus to a new cell. If the text is not fully selected, the key moves the cursor to the next position, and if it is not available the next cell is focused.
exHandleEditSel	1	The editor handles the arrow key. The key moves the focus to a new cell, if the editor displays a caret and the key is pressed. If the text is not fully selected, the key moves the caret inside the editor. The F2 key selects or unselects the text inside the editor.

constants AutoDragEnum

The AutoDragEnum type indicates what the control does when the user clicks and start dragging a row or an item. The AutoDrag property indicates the way the component supports the AutoDrag feature. The AutoDrag feature indicates what the control does when the user clicks an item and start dragging. For instance, using the AutoDrag feature you can automatically lets the user to drag and drop the data to OLE compliant applications like Microsoft Word, Excel and so on. The SingleSel property specifies whether the control supports single or multiple selection. The drag and drop operation starts once the user clicks and moves the cursor up or down, if the SingleSel property is True, and if SingleSel property is False, the drag and drop starts once the user clicks, and waits for a short period of time. If SingleSel property is False, moving up or down the cursor selects the items by drag and drop.

- The flag that ends on **...OnShortTouch** indicates the action the control does when the user short touches the screen
- The flag that ends on **...OnRight** indicates the action the control does when the user right clicks the control.
- The flag that ends on **...OnLongTouch** indicates the action the control does when the user long touches the screen

The AutoDragEnum type supports the following values:

Name	Value	Description
exAutoDragNone	0	AutoDrag is disabled. You can use the OLEDropMode property to handle the OLE Drag and Drop event for your custom action.
exAutoDragPosition	1	The item can be dragged from a position to another, but not outside of its group. If your items are arranged as a flat list, no hierarchy, this option can be used to allow the user change the item's position at runtime by drag and drop. This option does not change the parent of any dragged item. The dragging items could be the focused item or a contiguously selection. Click the selection and moves the cursor up or down, so the position of the dragging items is changed. The draggable collection is a collection of sortable items between 2 non-sortable items (SortableItem property). The drag and drop operation can not start on a non-sortable or non-selectable item (SelectableItem property). In other words, you can specify a range where an item can be dragged using the SortableItem

		property. Just set the SortableItem property on False, for margins, and so the items can be dragged between these items only.
exAutoDragPositionKeepIndent	2	The item can be dragged to any position or to any parent, while the dragging object keeps its indentation. This option can be used to allow the user change the item's position at runtime by drag and drop. In the same time, the parent's item could be changed but keeping the item's indentation. The dragging items could be the focused item or a contiguously selection. Click the selection and moves the cursor up or down, so the position or parent of the dragging items is changed. The drag and drop operation can not start on a non-sortable or non-selectable item (SelectableItem property). In other words, you can specify a range where an item can be dragged using the SortableItem property. Just set the SortableItem property on False, for margins, and so the items can be dragged between these items only.
exAutoDragPositionAny	3	The item can be dragged to any position or to any parent, with no restriction. The dragging items could be the focused item or a contiguously selection. The parent of the dragging items could change with no restrictions, based on the position of the dragging item. Click the selection and moves the cursor up or down, so the position or parent of the dragging items is changed. Click the selection and moves the cursor left or right, so the item's indentation is decreased or increased. The drag and drop operation can not start on a non-sortable or non-selectable item (SelectableItem property). In other words, you can specify a range where an item can be dragged using the SortableItem property. Just set the SortableItem property on False, for margins, and so the items can be dragged between these items only. Click here to watch a movie on how exAutoDragCopyText works.
		Drag and drop the selected items to a target

exAutoDragCopy	8	application, and paste them as image or text. Pasting the data to the target application depends on the application. You can use the exAutoDragCopyText to specify that you want to paste as Text, or exAutoDragCopyImage as an image.
exAutoDragCopyText	9	Drag and drop the selected items to a target application, and paste them as text only. Ability to drag and drop the data as text, to your favorite Office applications, like Word, Excel, or any other OLE-Automation compliant. The drag and drop operation can start anywhere Click here to watch a movie on how exAutoDragCopyText works.
exAutoDragCopyImage	10	Drag and drop the selected items to a target application, and paste them as image only. Ability to drag and drop the data as it looks, to your favorite Office applications, like Word, Excel, or any other OLE-Automation compliant. The drag and drop operation can start anywhere Click here to watch a movie on how exAutoDragCopyImage works.
exAutoDragCopySnapShot	11	Drag and drop a snap shot of the current component. This option could be used to drag and drop the current snap shot of the control to your favorite Office applications, like Word, Excel, or any other OLE-Automation compliant.
exAutoDragScroll	16	The component is scrolled by clicking the item and dragging to a new position. This option can be used to allow user scroll the control's content with NO usage of the scroll bar, like on your IPhone. Ability to smoothly scroll the control's content. The feature is useful for touch screens or tables pc, so no need to click the scroll bar in order to scroll the control's content. Use the ScrollBySingleLine property on False, to allow scrolling pixel by pixel when user clicks the up or down buttons on the vertical scroll bar. By default, the scrolling starts as soon as user

		clicks an item. If the cursor hovers a bar or AllowCreateBar property is not exNoCreateBar, click and wait for a second to start scrolling the chart. Click here or to watch a movie on how exAutoDragScroll works.
exAutoDragPositionOnShortTouch	256	The object can be dragged from a position to another, but not outside of its group.
exAutoDragPositionKeepIndentOnShortTouch	512	The object can be dragged to any position or to any parent, while the dragging object keeps its indentation.
exAutoDragPositionAnyOnShortTouch	768	The object can be dragged to any position or to any parent, with no restriction.
exAutoDragCopyOnShortTouch	2048	Drag and drop the selected objects to a target application, and paste them as image or text.
exAutoDragCopyTextOnShortTouch	2304	Drag and drop the selected objects to a target application, and paste them as text only.
exAutoDragCopyImageOnShortTouch	2560	Drag and drop the selected objects to a target application, and paste them as image only.
exAutoDragCopySnapShotOnShortTouch	2816	Drag and drop a snap shot of the current component.
exAutoDragScrollOnShortTouch	4096	The component is scrolled by clicking the object and dragging to a new position.
exAutoDragPositionOnRight	65536	The object can be dragged from a position to another, but not outside of its group.
exAutoDragPositionKeepIndentOnRight	131072	The object can be dragged to any position or to any parent, while the dragging object keeps its indentation.
exAutoDragPositionAnyOnRight	196608	The object can be dragged to any position or to any parent, with no restriction.
exAutoDragCopyOnRight	524288	Drag and drop the selected objects to a target application, and paste them as image or text.
exAutoDragCopyTextOnRight	589824	Drag and drop the selected objects to a target application, and paste them as text only.
exAutoDragCopyImageOnRight	655360	Drag and drop the selected objects to a target application, and paste them as image only.

exAutoDragCopySnapShotOnRight	720896	Drag and drop a snap shot of the current component.
exAutoDragScrollOnRight	1048576	The component is scrolled by clicking the object and dragging to a new position.
exAutoDragPositionOnLongTouch	16777216	The object can be dragged from a position to another, but not outside of its group.
exAutoDragPositionKeepIndentOnLongTouch	33554432	The object can be dragged to any position or to any parent, while the dragging object keeps its indentation.
exAutoDragPositionAnyOnLongTouch	67108864	The object can be dragged to any position or to any parent, with no restriction.
exAutoDragCopyOnLongTouch	134217728	Drag and drop the selected objects to a target application, and paste them as image or text.
exAutoDragCopyTextOnLongTouch	150994944	Drag and drop the selected objects to a target application, and paste them as text only.
exAutoDragCopyImageOnLongTouch	167772160	Drag and drop the selected objects to a target application, and paste them as image only.
exAutoDragCopySnapShotOnLongTouch	184549376	Drag and drop a snap shot of the current component.
exAutoDragScrollOnLongTouch	268435456	The component is scrolled by clicking the object and dragging to a new position.

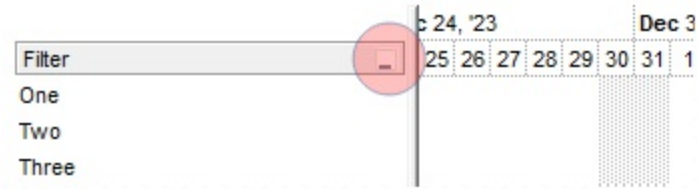
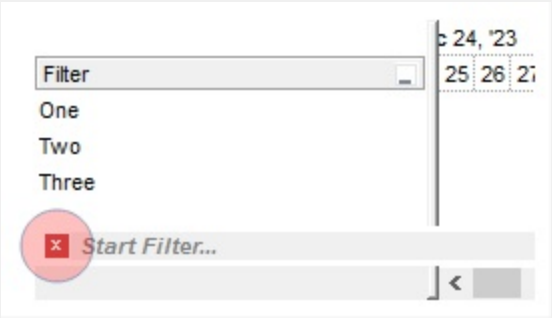
constants AutoSearchEnum

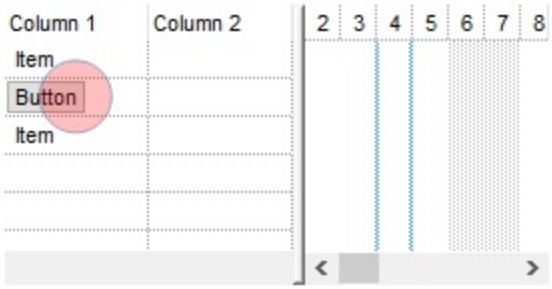
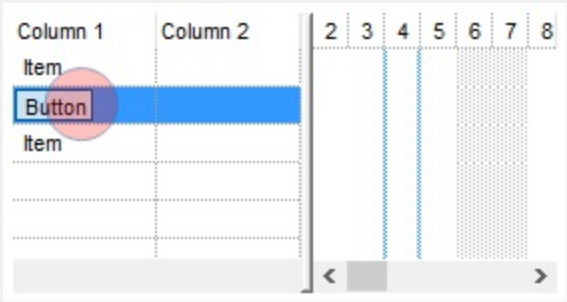
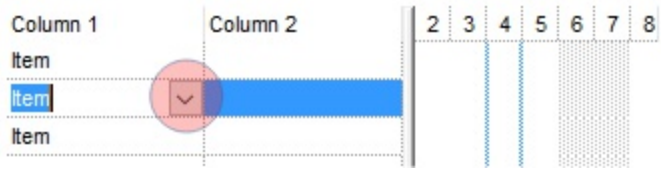
Specifies the kind of searching while user types characters within a column. Use the AutoSearch property to allow 'start with' incremental search or 'contains' incremental search feature in the control.

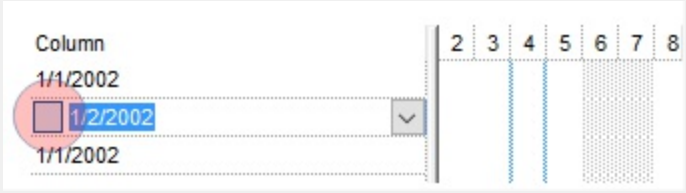
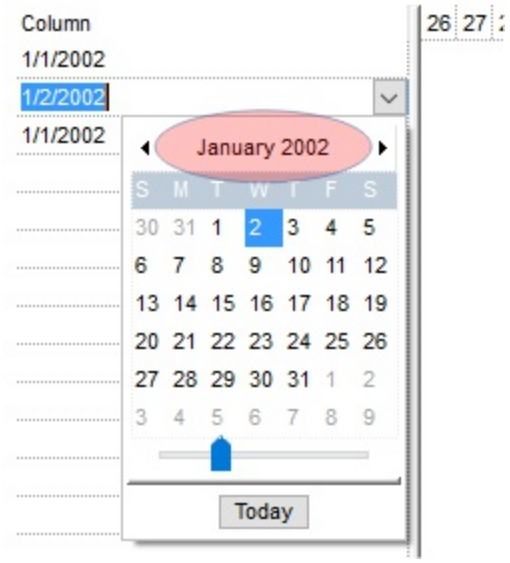
Name	Value	Description
exStartWith	0	Defines the 'starts with' incremental search within the column. If the user type characters within the column the control looks for items that start with the typed characters.
exContains	1	Defines the 'contains' incremental search within the column. If the user type characters within the column the control looks for items that contain the typed characters.

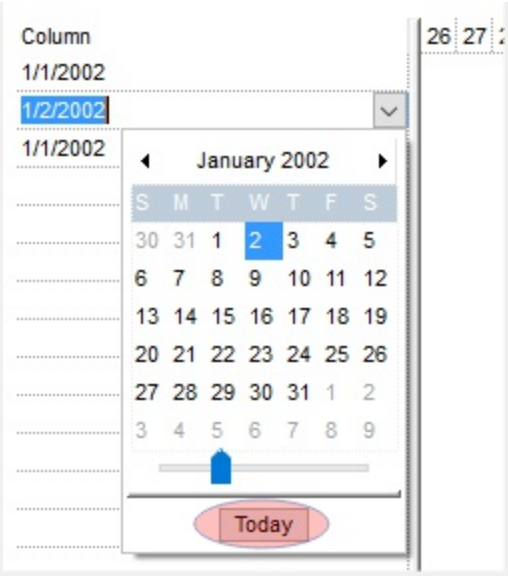
constants BackgroundPartEnum

The BackgroundPartEnum type indicates parts in the control. Use the Background property to specify a background color or a visual appearance for specific parts in the control. A Color expression that indicates the background color for a specified part. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

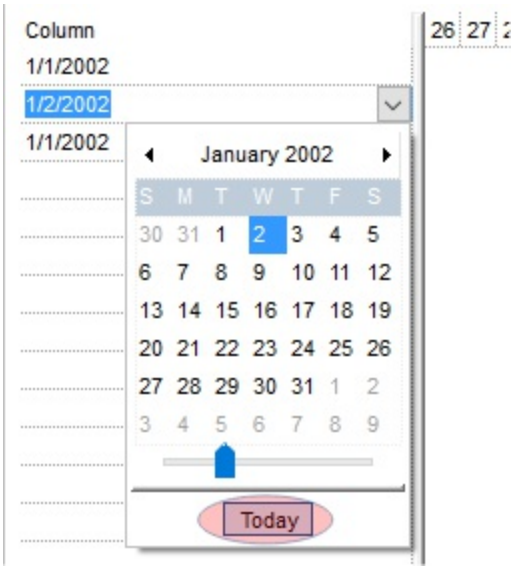
Name	Value	Description
exHeaderFilterBarButton	0	Specifies the background color for the drop down filter bar button. Use the DisplayFilterButton property to specify whether the drop down filter bar button is visible or hidden. 
exFooterFilterBarButton	1	Specifies the background color for the closing button in the filter bar (-1 hides the closing button in the filter bar). Use the ClearFilter method to remove the filter from the control. 
exCellButtonUp	2	Specifies the background color for the cell's button, when it is up. Use the CellHasButton property to assign a button to a cell.

		
exCellButtonDown	3	Specifies the background color for the cell's button, when it is down. Use the CellHasButton property to assign a button to a cell. 
exDropDownButtonUp	4	Specifies the visual appearance for the drop down button, when it is up. Usually the editors with a drop down portion displays a drop down button. 
exDropDownButtonDown	5	Specifies the visual appearance for the drop down button, when it is down. Usually the editors with a drop down portion displays a drop down button. 
exButtonUp	6	Specifies the visual appearance for the button inside the editor, when it is up. Use the AddButton method to add new buttons to an editor.

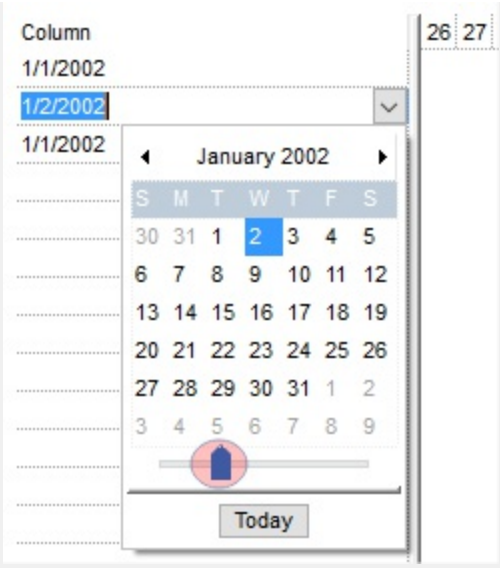
exButtonDown	7	Specifies the visual appearance for the button inside the editor, when it is down. Use the AddButton method to add new buttons to an editor. 
exDateHeader	8	Specifies the visual appearance for the header in a calendar control. The DateType editor allows user to select dates from a drop down calendar panel. 
exDateTodayUp	9	Specifies the visual appearance for the today button in a calendar control, when it is up. The DateType editor allows user to select dates from a drop down calendar panel.



exDateTodayDown	10	Specifies the visual appearance for the today button in a calendar control, when it is down. The DateType editor allows user to select dates from a drop down calendar panel.

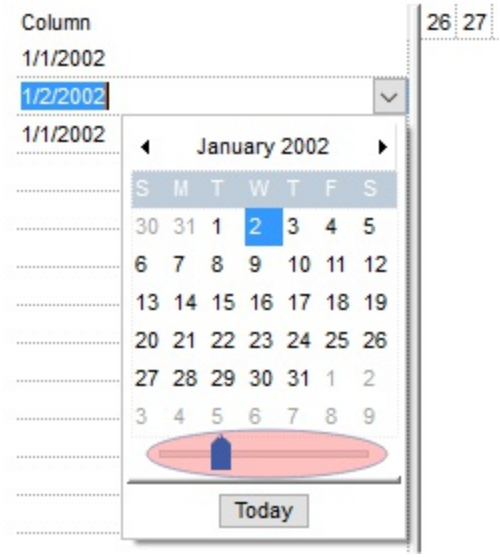


exDateScrollThumb	11	Specifies the visual appearance for the scrolling thumb in a calendar control. The DateType editor allows user to select dates from a drop down calendar panel.



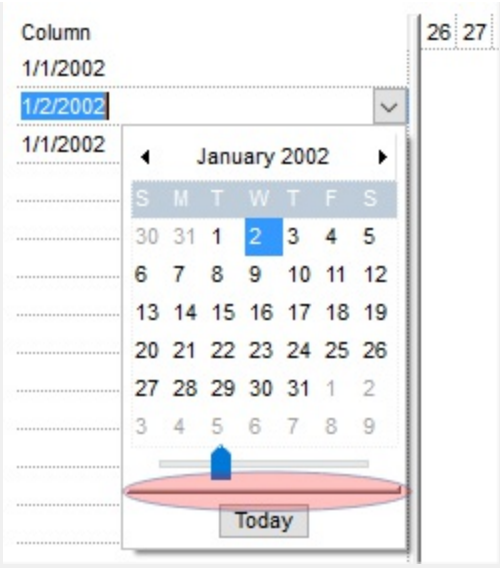
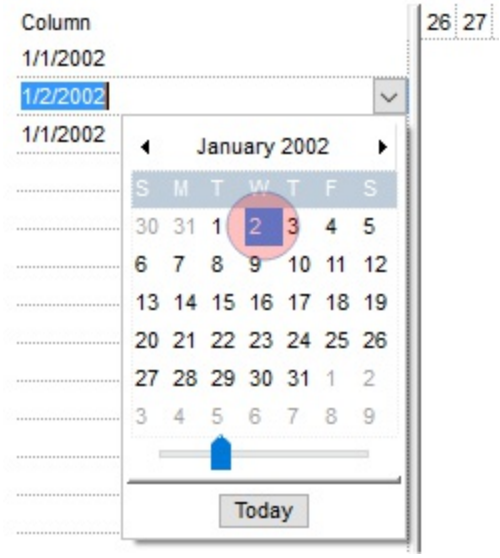
Specifies the visual appearance for the scrolling range in a calendar control. The DateType editor allows user to select dates from a drop down calendar panel.

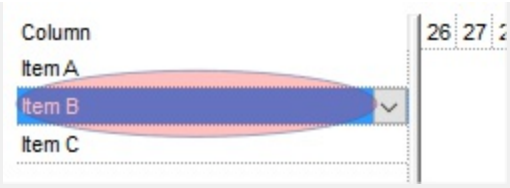
exDateScrollRange 12

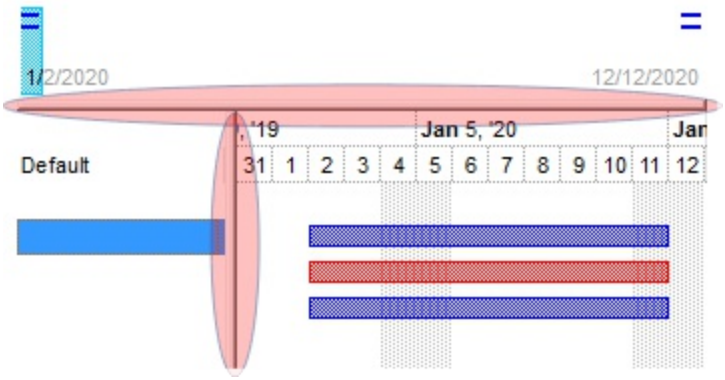
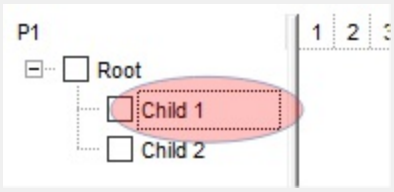


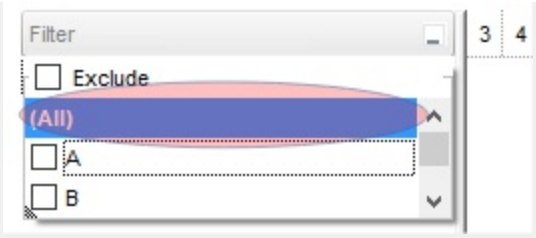
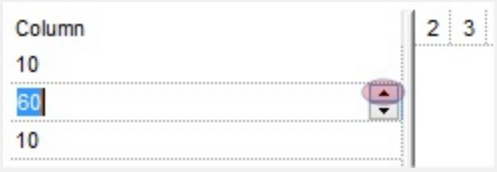
Specifies the visual appearance for the separator bar in a calendar control. The DateType editor allows user to select dates from a drop down calendar panel.

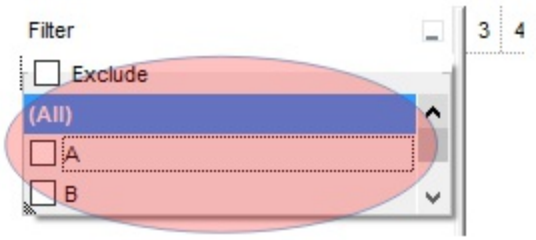
exDateSeparatorBar 13

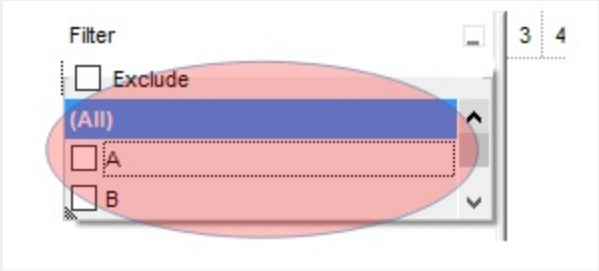
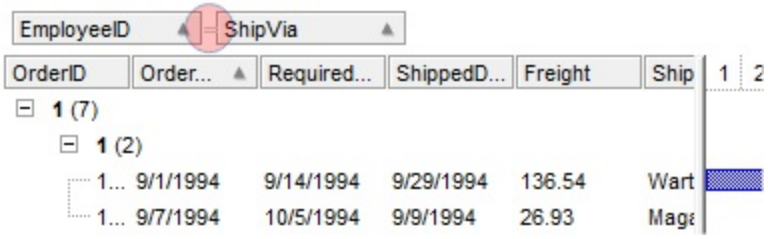
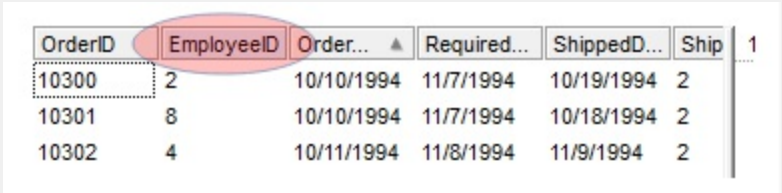
		
exDateSelect	14	Specifies the visual appearance for the selected date in a calendar control. The DateType editor allows user to select dates from a drop down calendar panel. 
exSliderRange	15	exSliderRange. Specifies the visual appearance for the slider's bar.
exSliderThumb	16	exSliderThumb. Specifies the visual appearance for the thumb of the slider.
exSelectInPlace	17	Specifies the visual appearance for the selection when a drop down editor is focused and closed. The option is valid for drop-down list editors (CheckListType, DropDownList).



exSplitBar	18	Specifies the visual appearance for control's split bar. 
exShowFocusRect	19	Specifies the visual appearance to display the cell with the focus. The ShowFocusRect property retrieves or sets a value indicating whether the control draws a thin rectangle around the focused item. 
exSelBackColorFilter	20	Specifies the visual appearance for the selection in the drop down filter window. Use the exBackColorFilter option to specify the background color in the drop down filter window. 
		Specifies the foreground color for the selection in the drop down filter window. Use the exForeColorFilter option to specify the foreground color in the drop down filter window.

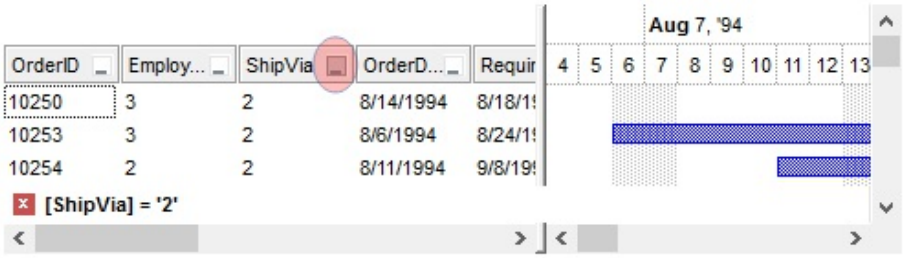
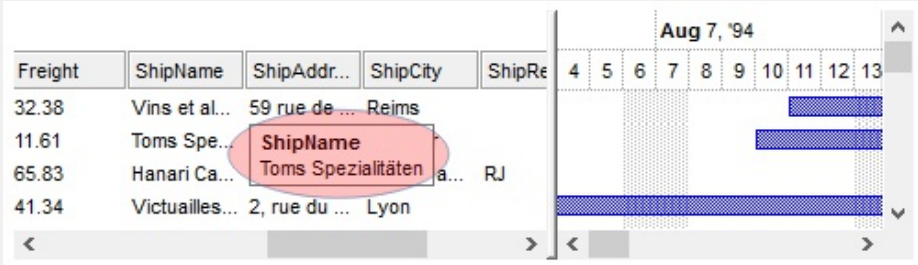
exSelForeColorFilter	21	
exSpinUpButtonUp	22	Specifies the visual appearance for the up spin button when it is not pressed. 
exSpinUpButtonDown	23	Specifies the visual appearance for the up spin button when it is pressed. 
exSpinDownButtonUp	24	Specifies the visual appearance for the down spin button when it is not pressed. 
exSpinDownButtonDown	25	Specifies the visual appearance for the down spin button when it is pressed. 
exBackColorFilter	26	Specifies the background color for the drop down filter window. If not specified, the BackColorHeader property specifies the drop down filter's background color. Use the exSelBackColorFilter option to specify the selection background visual appearance in the drop down filter window.

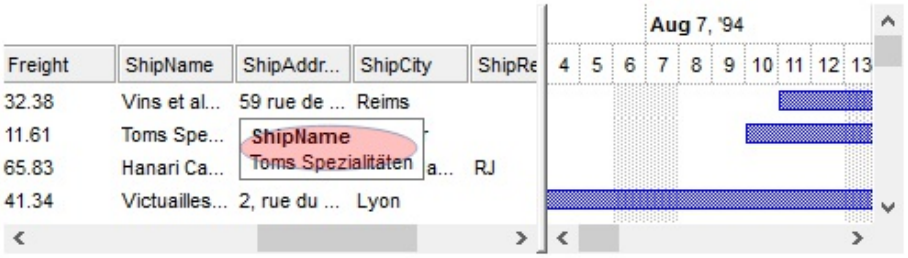
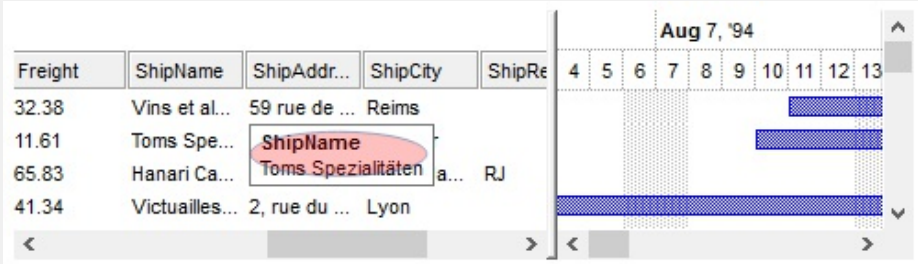
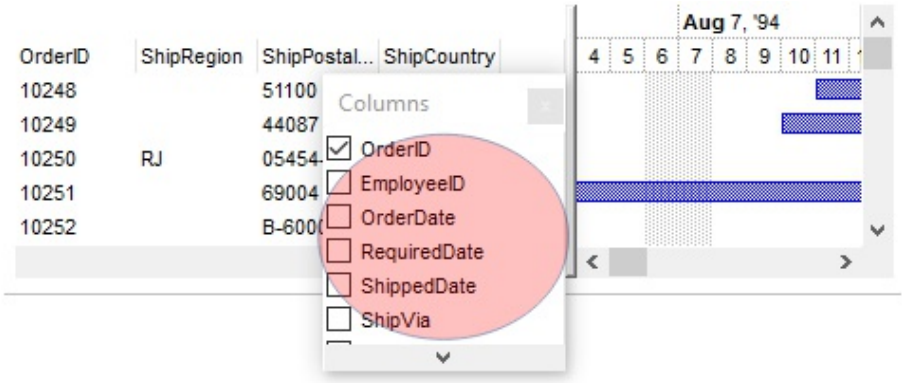
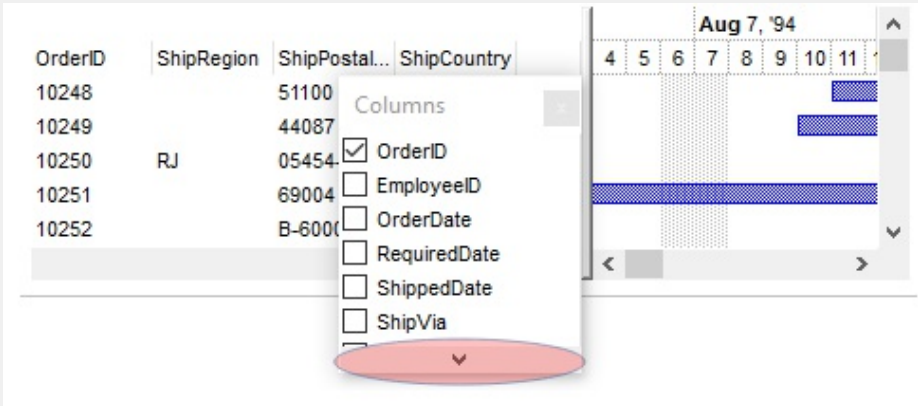


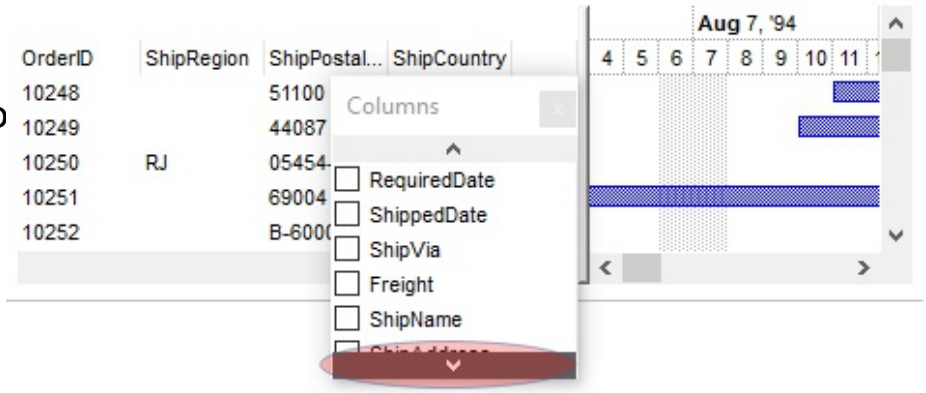
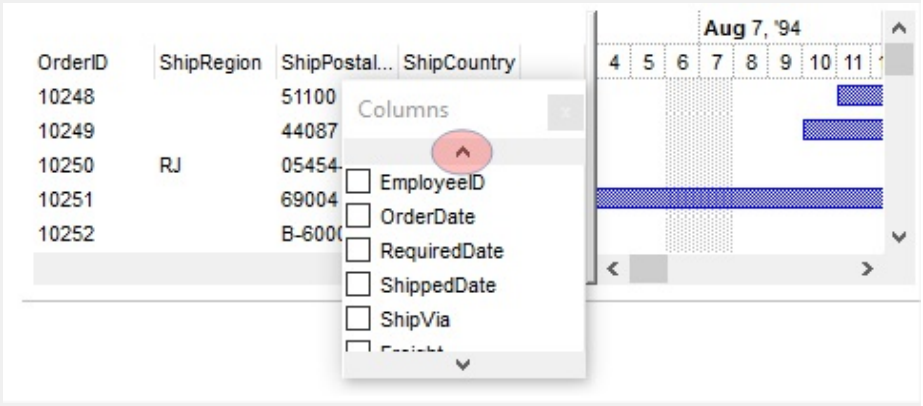
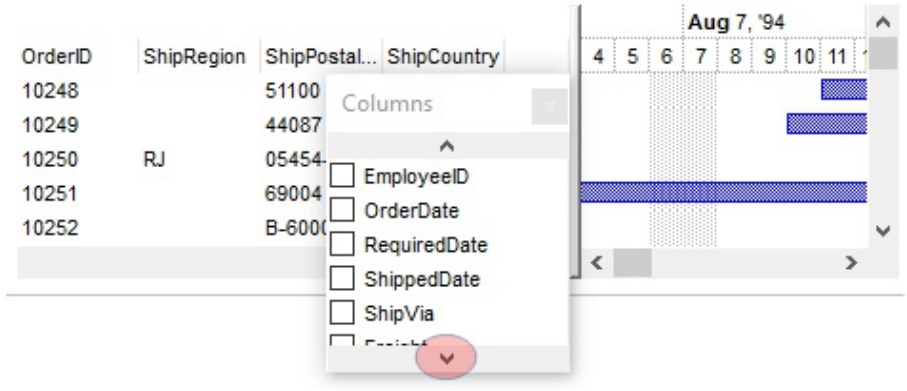
exForeColorFilter	27	Specifies the foreground color for the drop down filter window. If not specified, the ForeColorHeader property specifies the drop down filter's foreground color. Use the exSelForeColorFilter option to specify the selection foreground color in the drop down filter window. 
exSortBarLinkColor	28	Indicates the color or the visual appearance of the links between columns in the control's sort bar. 
exCursorHoverColumn	32	Specifies the visual appearance for the column when the cursor hovers the column. By default, the exCursorHoverColumn property is zero, and it has no effect, so the visual appearance for the column is not changed when the cursor hovers the header. 
exDragDropBefore	33	Specifies the visual appearance for the drag and drop cursor before showing the items. This option

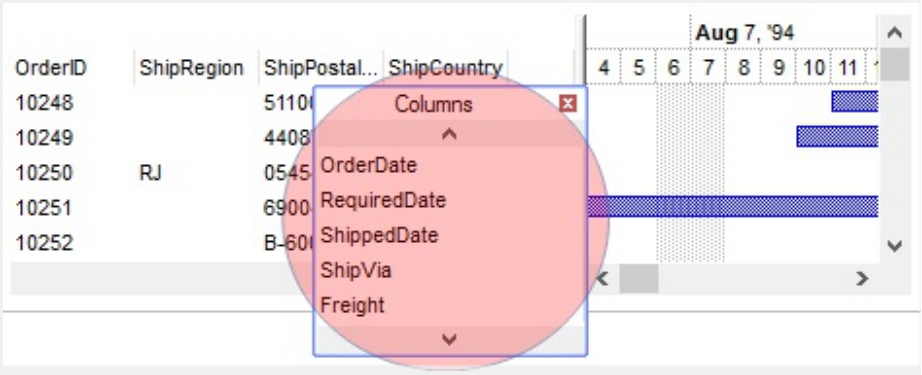
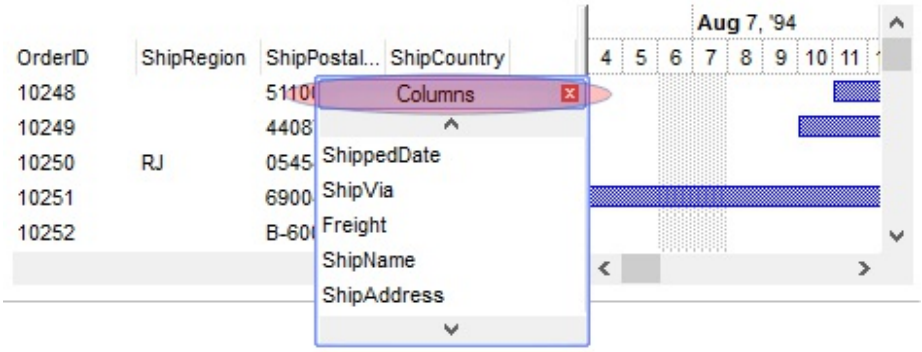
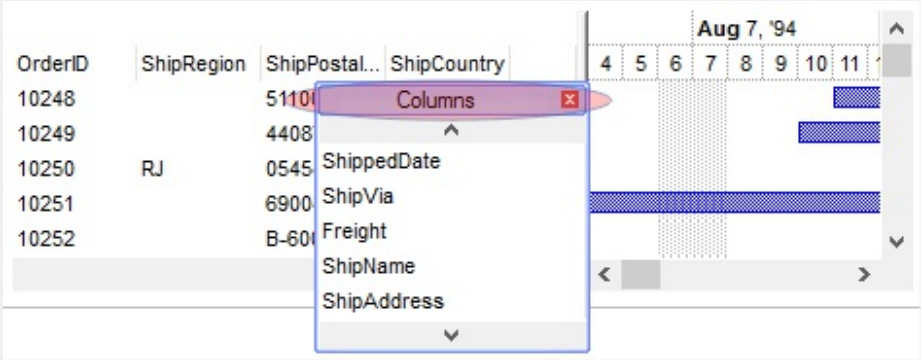
		can be used to apply a background to the dragging items, before painting the items.
exDragDropAfter	34	Specifies the visual appearance for the drag and drop cursor after showing the items. This option can be used to apply a semi-transparent/opaque background to the dragging items, after painting the items. If the exDragDropAfter option is set on white (0x00FFFFFF), the image is not showing on OLE Drag and drop.
exDragDropListTop	35	Specifies the graphic feedback of the item from the drag and drop cursor if the cursor is in the top half of the row. *Please note, that if a visual effect is specified for exDragDropListOver AND exDragDropListBetween states, and a visual effect is specified for exDragDropListTop OR/AND exDragDropListBottom state(s), the exDragDropListTop visual effect is displayed ONLY if the cursor is over the first visible item, and the exDragDropListBottom visual effect is shown ONLY for the last visible item. Use the ItemFromPoint property to retrieve the hit test code for the part from the cursor. This option can be changed during the OLEDragOver event to change the visual effect for the item from the cursor at runtime.*
exDragDropListBottom	36	Specifies the graphic feedback of the item from the drag and drop cursor if the cursor is in the bottom half of the row. *Please note, that if a visual effect is specified for exDragDropListOver AND exDragDropListBetween states, and a visual effect is specified for exDragDropListTop OR/AND exDragDropListBottom state(s), the exDragDropListTop visual effect is displayed ONLY if the cursor is over the first visible item, and the exDragDropListBottom visual effect is shown ONLY for the last visible item. Use the ItemFromPoint property to retrieve the hit test code for the part from the cursor. This option can be changed during the OLEDragOver event to change the visual effect for the item from the cursor at runtime.*

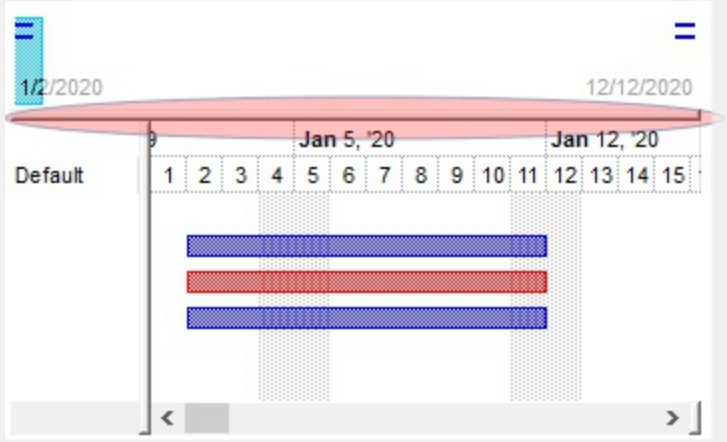
exDragDropForeColor	37	Specifies the foreground color for the items being dragged. By default, the foreground color is black.
exDragDropListOver	38	Specifies the graphic feedback of the item from the cursor if it is over the item. *Please note, that if a visual effect is specified for exDragDropListOver AND exDragDropListBetween states, and a visual effect is specified for exDragDropListTop OR/AND exDragDropListBottom state(s), the exDragDropListTop visual effect is displayed ONLY if the cursor is over the first visible item, and the exDragDropListBottom visual effect is shown ONLY for the last visible item. Use the ItemFromPoint property to retrieve the hit test code for the part from the cursor. This option can be changed during the OLEDragOver event to change the visual effect for the item from the cursor at runtime.*
exDragDropListBetween	39	Specifies the graphic feedback of the item when the drag and drop cursor is between items. *Please note, that if a visual effect is specified for exDragDropListOver AND exDragDropListBetween states, and a visual effect is specified for exDragDropListTop OR/AND exDragDropListBottom state(s), the exDragDropListTop visual effect is displayed ONLY if the cursor is over the first visible item, and the exDragDropListBottom visual effect is shown ONLY for the last visible item. Use the ItemFromPoint property to retrieve the hit test code for the part from the cursor. This option can be changed during the OLEDragOver event to change the visual effect for the item from the cursor at runtime.*
		Specifies the alignment of the drag and drop image relative to the cursor. By default, the exDragDropAlign option is 0, which initially the drag and drop image is shown centered relative to the position of the cursor. The valid values are listed as follows (hexa representation): • 0x00000000, (default), the drag and drop

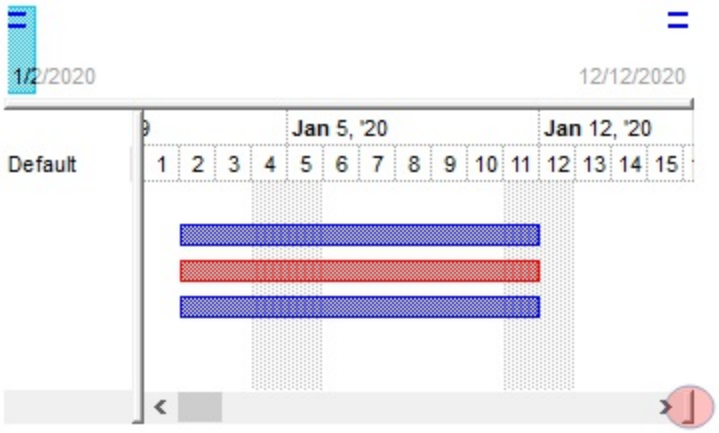
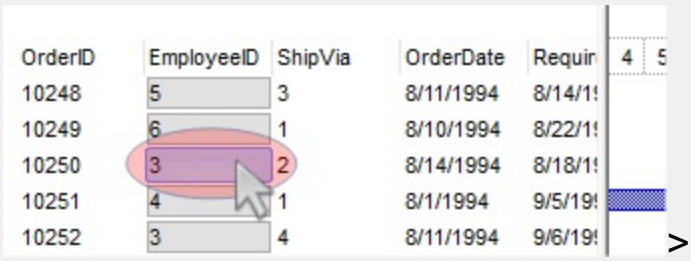
exDragDropAlign	40	image is shown centered relative to the cursor, and shows up. • 0x01000000, (left), the drag and drop image is shown to the left of the cursor. • 0x02000000, (right), the drag and drop image is shown to the right of the cursor. • 0x04000000, (center-down), the drag and drop image is shown centered relative to the cursor, and shows down. • 0xFF000000, (as- is), the drag and drop image is shown as it is clicked.
exHeaderFilterBarActive	41	Specifies the visual appearance of the drop down filter bar button, while filter is applied to the column. 
exToolTipAppearance	64	Indicates the visual appearance of the borders of the tooltips. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the CellToolTip property to specify the cell's tooltip. Use the ToolTipWidth property to specify the width of the tooltip window. Use the ItemBar(.,exBarToolTip) property to specify a tooltip for a bar. Use the Link(,exLinkToolTip) property to specify the tooltip to be shown when the cursor hovers the link. Use the ShowToolTip method to display a custom tooltip 

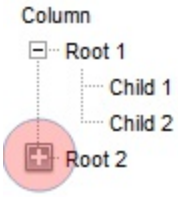
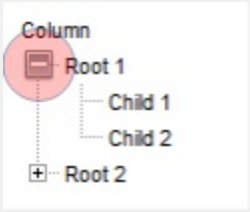
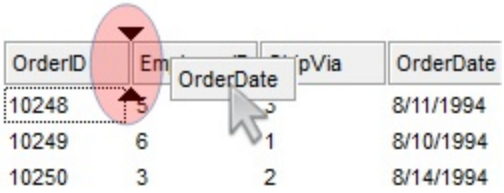
exToolTipBackColor	65	Specifies the tooltip's background color. 
exToolTipForeColor	66	Specifies the tooltip's foreground color. 
exColumnsFloatBackColor	87	Specifies the background color for the Columns float bar. 
exColumnsFloatScrollBackColor	88	Specifies the background color for the scroll bars in the Columns float bar. 
		Specifies the background color for the scroll bars in

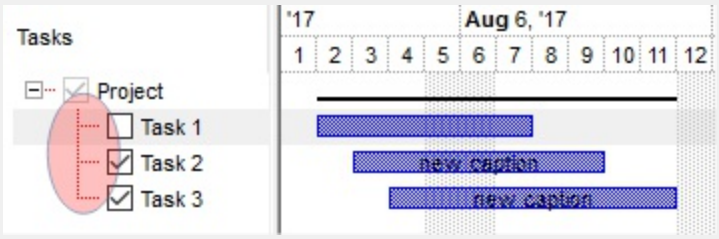
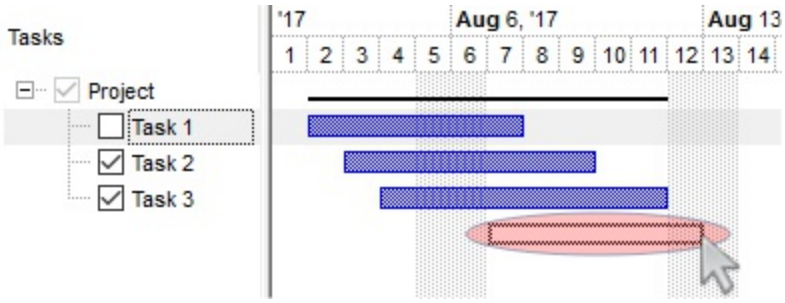
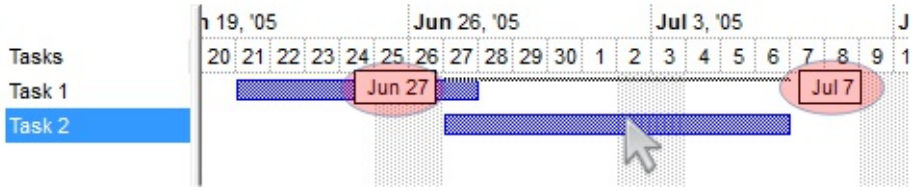
exColumnsFloatScrollPressBackColo	89	the Columns float bar, while the scroll part is pressed. 
exColumnsFloatScrollUp	90	Specifies the visual appearance of the up scroll bar. 
exColumnsFloatScrollDown	91	Specifies the visual appearance of the down scroll bar. 
		Specifies the visual appearance for the frame/borders of the Column's float bar. The option

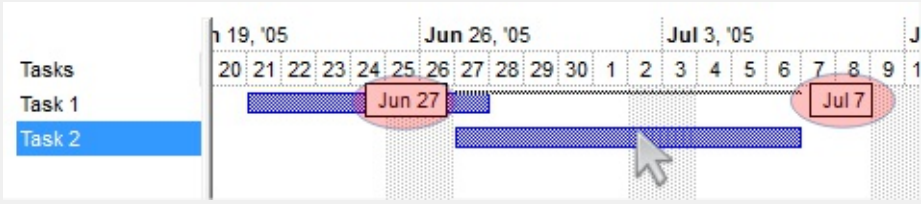
		has effect only if set before calling the ColumnsFloatBarVisible property.
exColumnsFloatAppearance	92	
exColumnsFloatCaptionBackColor	93	Specifies the visual appearance for caption, if the Background(exColumnsFloatAppearance) property is specified.
		
exColumnsFloatCaptionForeColor	94	Specifies the foreground color for the caption, if the Background(exColumnsFloatAppearance) property is specified.
		
exColumnsFloatCloseButton	95	exColumnsFloatCloseButton. Specifies the visual appearance for the closing button, if the Background(exColumnsFloatAppearance) property

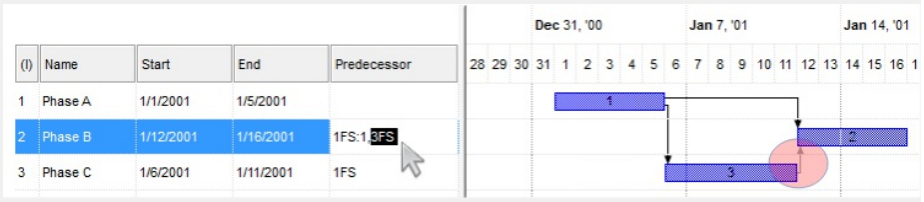
		is specified.
exListOLEDropPosition	96	By default, the exListOLEDropPosition is 0, which means no effect. Specifies the visual appearance of the dropping position over the list part of the control, when it is implied in a OLE Drag and Drop operation. The exListOLEDropPosition has effect only if different than 0, and the OLEDropMode property is not exOLEDropNone. For instance, set the Background(exListOLEDropPosition) property on RGB(0,0,255), and a blue line is shown at the item where the cursor is hover the list part of the control, during an OLE Drag and Drop position. The OLEDragDrop event notifies your application once an object is drop in the control.
exChartOLEDropPosition	97	By default, the exChartOLEDropPosition is 0, which means no effect. Specifies the visual appearance of the dropping position over the chart part of the control, when it is implied in a OLE Drag and Drop operation. The exChartOLEDropPosition has effect only if different than 0, and the OLEDropMode property is not exOLEDropNone. For instance, set the Background(exChartOLEDropPosition) property on RGB(0,0,255), and a blue line is shown at the date-time position where the cursor is hover the chart part of the control, during an OLE Drag and Drop position. The OLEDragDrop event notifies your application once an object is drop in the control.
exHSplitBar	141	Specifies the visual appearance for horizontal split bar. 
		Specifies the solid color / visual appearance of the

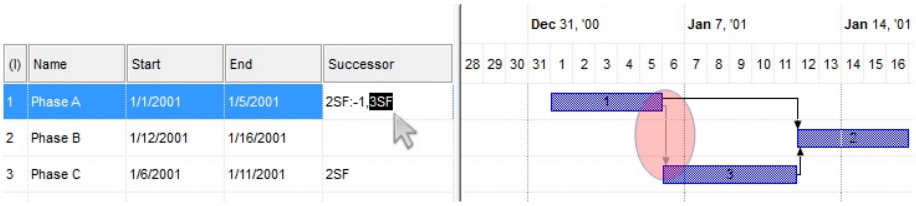
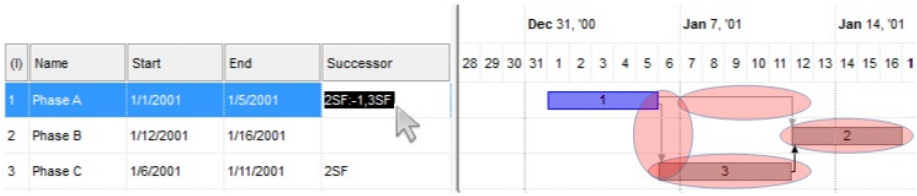
exCSplitBar	142	split bar that creates new views. The AllowSplitPane property specifies whether the chart panel supports splitting. 
exCursorHoverCellButton	157	Specifies the visual appearance for the cell's button when the cursor hovers it. The CellHasButton property specifies whether the cell display a button inside. 
exSelBackColorHide	166	Specifies the selection's background color, when the control has no focus. This has effect while the control's HideSelection property is False 
		Specifies the selection's foreground color, when the control has no focus. This has effect while the control's HideSelection property is False

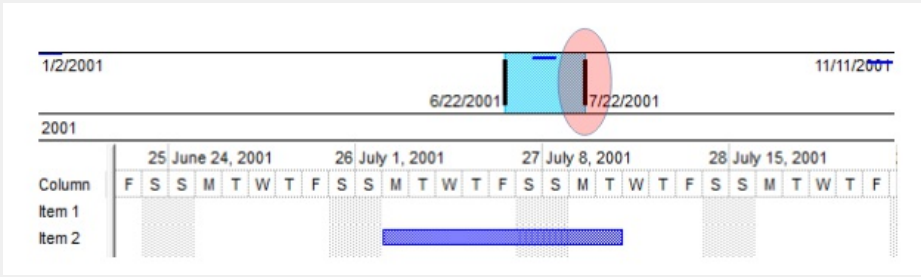
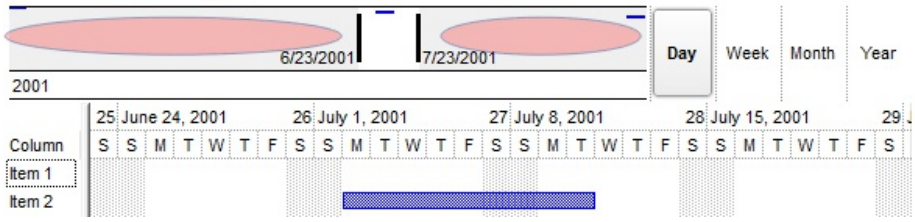
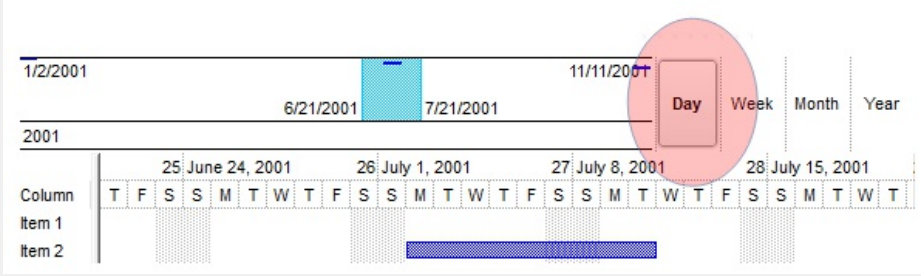
exSelForeColorHide	167	
exTreeGlyphOpen	180	Specifies the visual appearance for the +/- buttons when it is collapsed. This option is valid while HasButtons property is exPlus (by default), and any of Background(exTreeGlyphOpen)/Background(exTreeG is not-zero. 
exTreeGlyphClose	181	Specifies the visual appearance for the +/- buttons when it is expanded. This option is valid while HasButtons property is exPlus (by default), and any of Background(exTreeGlyphOpen)/Background(exTreeG is not-zero 
exColumnsPositionSign	182	Specifies the visual appearance for the position sign between columns, when the user changes the position of the column by drag an drop. The AllowDragging property specifies whether the user can change the column's position by drag an drop. 

exTreeLinesColor	186	Specifies the color to show the tree-lines (connecting lines from the parent to the children). The HasLines property enhances the graphic representation of a tree control's hierarchy by drawing lines that link child items to their corresponding parent item. 
exChartCreateBar	188	Specifies the visual appearance to show the frame to create newly bars by drag and drop in the chart panel. The AllowCreateBar property specifies whether the user can create new bars by drag and drop. 
exCreateBarHeight	189	Specifies the height of the frame to create newly bars by drag and drop in the chart panel. The AllowCreateBar property specifies whether the user can create new bars by drag and drop.
exDateTickerLabelBack	192	Specifies the visual appearance to display the date label, while create, resize or move a bar. The background of the date label is not applied if -1. By default, the background of the date label is white (0). The DateTickerLabel property specifies the label to show the start/end margins of the bar being created, resize or moved by drag and drop. 

exDateTickerLabelFore	193	Specifies the label's foreground color, while create, resize or move a bar. By default, the background of the date label is black (0). The DateTickerLabel property specifies the label to show the start/end margins of the bar being created, resize or moved by drag and drop. 
exDateTickerLabelHAlign	194	Specifies the default horizontal alignment of the date label, while create, resize or move a bar. The DateTickerLabel property specifies the label to show the start/end margins of the bar being created, resize or moved by drag and drop. The exDateTickerLabelHAlign property supports the following values: • 0, (left,default) the lines of the date-label are left-aligned • 1, (center) the lines of the date-label are horizontally centered • 2, (right) the lines of the date-label are right-aligned
exDateTickerLabelVAlign	195	Specifies the default vertical alignment of the date label, while create, resize or move a bar. The DateTickerLabel property specifies the label to show the start/end margins of the bar being created, resize or moved by drag and drop. The exDateTickerLabelVAlign property supports the following values: • 0, (top,default) the date-label is shown right below the chart's header bar • 1, (center) the date-label is following the bar being created, resized or moved
		Specifies the distance between the date-label and

exDateTickerLabelHMargin	196	the bar. The DateTickerLabel property specifies the label to show the start/end margins of the bar being created, resize or moved by drag and drop.
exPSLinkColorEditSel	197	Specifies the color to highlight the links being selected within an editable predecessor/successor column. The exPSLinkColorEditSel property has effect if it is not zero. The following properties must be set to add an editable predecessor/successor column: • Items.AllowCellValueToItemBar = True • Column.Def(exCellValueToItemBarProperty) = exBarPredecessor(270) or Column.Def(exCellValueToItemBarProperty) = exBarSuccessor(271) (property of Column object) • Editor.EditType = EditType(1) or Editor.EditType = MaskType(8) (property of Editor object) The following screen shot highlights (in gray) the selected-link within a predecessor column:  The following screen shot highlights the selected-link and its incoming-bars (in gray) within a predecessor column: 
		Specifies the color to highlight the incoming/outgoing bars of the links being selected within an editable predecessor/successor column.

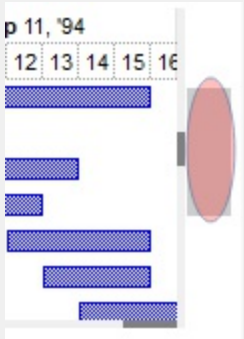
		The exPSBarColorEditSel property has effect if it is not zero. The following properties must be set to add an editable predecessor/successor column: • Items.AllowCellValueToItemBar = True • Column.Def(exCellValueToItemBarProperty) = exBarPredecessor(270) or Column.Def(exCellValueToItemBarProperty) = exBarSuccessor(271) (property of Column object) • Editor.EditType = EditType(1) or Editor.EditType = MaskType(8) (property of Editor object)
exPSBarColorEditSel	198	The following screen shot highlights the outgoing bar (in gray) of a selected link, within a successor column:  The following screen shot highlights the selected-link and its outgoing-bars (in gray) within a successor column: 
exOverviewSelResize	199	Specifies the visual appearance to show and use the left/ resize-margins of the overview's selection. By default, the Background(exOverviewSelResize) property is 0 which indicates no effect. The Background(exOverviewSelResize) property can be a solid color such as RGB(0,0,1) or an EBN color should as 0x1000000. The user can resize the chart by drag and drop the left or right resize-margins of the overview-selection, while the Background(exOverviewSelResize) property is not zero. The AllowResizeChart property specifies whether the user can enlarge (zoom-in,zoom-out) or

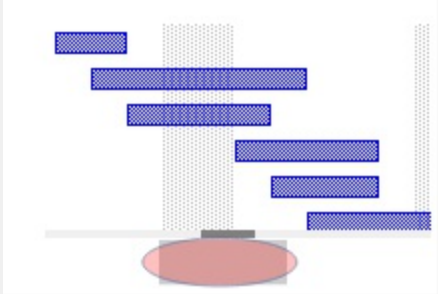
		resize the chart using the control's header, middle mouse button. 
exOverviewSelOut	200	Specifies the visual-appearance to display the left/right parts outside of the overview-selection. By default, the Background(exOverviewSelOut) property is 0 which indicates no effect. The Background(exOverviewSelUnit) property specifies the background color / visual appearance to display the selected unit within the control's overview. The Chart.OverviewSelBackColor property specifies background color or the visual appearance for the selection in the chart's overview 
exOverviewSelUnit	201	Specifies the background color / visual appearance to display the selected unit within the control's overview. By default, the Background(exOverviewSelUnit) property is 0 which indicates that the Chart.OverviewSelBackColor property specifies the background color of the selected unit-scale 

exVSUp	256	 align="top"> The up button in normal state.
exVSUpP	257	The up button when it is pressed.
exVSUpD	258	The up button when it is disabled.
exVSUpH	259	The up button when the cursor hovers it.
exVSThumb	260	 align="top"> The thumb part (exThumbPart) in normal state.
exVSThumbP	261	The thumb part (exThumbPart) when it is pressed.
exVSThumbD	262	The thumb part (exThumbPart) when it is disabled.
exVSThumbH	263	The thumb part (exThumbPart) when cursor hovers it.
exVSDown	264	 align="top"> The down button in normal state.
exVSDownP	265	The down button when it is pressed.
exVSDownD	266	The down button when it is disabled.
exVSDownH	267	The down button when the cursor hovers it.
exVSLower	268	 align="top"> The lower part (exLowerBackPart) in normal state.
exVSLowerP	269	The lower part (exLowerBackPart) when it is pressed.
		The lower part (exLowerBackPart) when it is

exVSLowerD	270	disabled.
exVSLowerH	271	The lower part (exLowerBackPart) when the cursor hovers it.
exVSUpper	272	 align="top"> The upper part (exUpperBackPart) in normal state.
exVSUpperP	273	The upper part (exUpperBackPart) when it is pressed.
exVSUpperD	274	The upper part (exUpperBackPart) when it is disabled.
exVSUpperH	275	The upper part (exUpperBackPart) when the cursor hovers it.
exVSBack	276	 align="top"> The background part (exLowerBackPart and exUpperBackPart) in normal state.
exVSBackP	277	The background part (exLowerBackPart and exUpperBackPart) when it is pressed.
exVSBackD	278	The background part (exLowerBackPart and exUpperBackPart) when it is disabled.
exVSBackH	279	The background part (exLowerBackPart and exUpperBackPart) when the cursor hovers it.
exHSLeft	384	 The left button in normal state.
exHSLeftP	385	The left button when it is pressed.
exHSLeftD	386	The left button when it is disabled.
exHSLeftH	387	The left button when the cursor hovers it.
		 The thumb part (exThumbPart) in

exHSThumb	388	normal state.
exHSThumbP	389	The thumb part (exThumbPart) when it is pressed.
exHSThumbD	390	The thumb part (exThumbPart) when it is disabled.
exHSThumbH	391	The thumb part (exThumbPart) when the cursor hovers it.
exHSRight	392	 The right button in normal state.
exHSRightP	393	The right button when it is pressed.
exHSRightD	394	The right button when it is disabled.
exHSRightH	395	The right button when the cursor hovers it.
exHSLower	396	 The lower part (exLowerBackPart) in normal state.
exHSLowerP	397	The lower part (exLowerBackPart) when it is pressed.
exHSLowerD	398	The lower part (exLowerBackPart) when it is disabled.
exHSLowerH	399	The lower part (exLowerBackPart) when the cursor hovers it.
exHSUpper	400	 The upper part (exUpperBackPart) in normal state.
exHSUpperP	401	The upper part (exUpperBackPart) when it is pressed.
exHSUpperD	402	The upper part (exUpperBackPart) when it is disabled.
exHSUpperH	403	The upper part (exUpperBackPart) when the cursor hovers it.
exHSBack	404	 The background part (exLowerBackPart and exUpperBackPart) in normal state.
exHSBackP	405	The background part (exLowerBackPart and exUpperBackPart) when it is pressed.
		The background part (exLowerBackPart and

exHSBackD	406	exUpperBackPart) when it is disabled.
exHSBackH	407	The background part (exLowerBackPart and exUpperBackPart) when the cursor hovers it.
exSBtn	324	 All button parts (L1-L5, LButton, exThumbPart, RButton, R1-R6), in normal state.
exSBtnP	325	All button parts (L1-L5, LButton, exThumbPart, RButton, R1-R6), when it is pressed.
exSBtnD	326	All button parts (L1-L5, LButton, exThumbPart, RButton, R1-R6), when it is disabled.
exSBtnH	327	All button parts (L1-L5, LBUtton, exThumbPart, RButton, R1-R6), when the cursor hovers it .
exScrollHoverAll	500	Enables or disables the hover-all feature. By default (Background(exScrollHoverAll) = 0), the left/top, right/bottom and thumb parts of the control' scrollbars are displayed in hover state while the cursor hovers any part of the scroll bar (hover-all feature). The hover-all feature is available on Windows 11 or greater, if only left/top, right/bottom, thumb, lower and upper-background parts of the scrollbar are visible, no custom visual-appearance is applied to any visible part. The hover-all feature is always on If Background(exScrollHoverAll) = -1. The Background(exScrollHoverAll) = 1 disables the hover-all feature.
exVSThumbExt	503	The thumb-extension part in normal state. The ScrollPartVisible property indicates whether the specified scroll part is visible or hidden. The exExtentThumbPart part indicates the thumb-extension part.. 

exVSThumbExtP	504	The thumb-extension part when it is pressed.
exVSThumbExtD	505	The thumb-extension part when it is disabled.
exVSThumbExtH	506	The thumb-extension when the cursor hovers it.
exHSThumbExt	507	The thumb-extension in normal state. 
exHSThumbExtP	508	The thumb-extension when it is pressed.
exHSThumbExtD	509	The thumb-extension when it is disabled.
exHSThumbExtH	510	The thumb-extension when the cursor hovers it.
exScrollSizeGrip	511	Specifies the visual appearance of the control's size grip when both scrollbars are shown. 

If you refer a part of the scroll bar please notice the following:

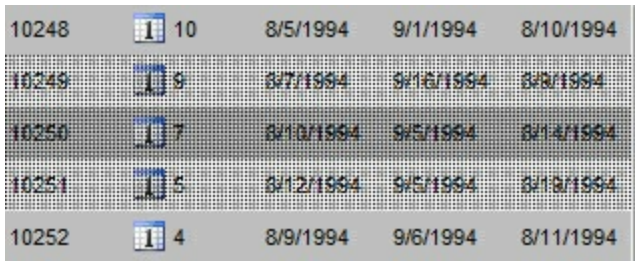
- All BackgroundPartEnum expressions that starts with **exVS** changes a part in a vertical scroll bar
- All BackgroundPartEnum expressions that starts with **exHS** changes a part in the horizontal scroll bar
- Any BackgroundPartEnum expression that ends with **P** (and starts with exVS or exHS) specifies a part of the scrollbar when it is pressed.
- Any BackgroundPartEnum expression that ends with **D** (and starts with exVS or exHS) specifies a part of the scrollbar when it is disabled.
- Any BackgroundPartEnum expression that ends with **H** (and starts with exVS or exHS) specifies a part of the scrollbar when the cursor hovers it.
- Any BackgroundPartEnum expression that ends with no **H**, **P** or **D** (and starts with

exVS or exHS) specifies a part of the scrollbar on normal state

constants BackModeEnum

Specifies how the control displays the selection. The SelBackMode property specifies the way the selected items are shown in the control. Use the SelBackColor property to specify the background color for selected items in the component. The SelForeColor property specifies the foreground color of the selected item. Use the SelectItem property to selects an item by code. The BackModeEnum type supports the following values:

Name	Value	Description
exOpaque	0	(Default) The selection is opaque. The selected items overrides any background color or picture. The following screen shot shows the selected items (10249, 10250, 10251), when the SelBackMode on exOpaque: 
exTransparent	1	The selection is transparent. The selected items is combined with background color or picture. The following screen shot shows the selected items (10249, 10250, 10251), when the SelBackMode on exTransparent: 
		The control paints a grid selection. The selected items is shown in an array of dots. The following screen shot shows the selected items

exGrid	2	(10249, 10250, 10251), when the SelBackMode on exTransparent: 

constants BarOperationEnum

The BarOperationEnum type specifies an operation being notified in the chart area.

Name	Value	Description
exMoveBar	1	A bar is resized or moved. The ItemBar(exBarStart) and ItemBar(exBarEnd) indicates the start and ending points of the bar.
exResizeStartBar	2	A bar is resized to the left. The ItemBar(exBarStart) specifies the starting date-time of the bar.
exResizeEndBar	3	A bar is resized to the right. The ItemBar(exBarEnd) specifies the ending date-time of the bar.
exAddLink	4	Adds a new link. The AddLink method adds a link between two bars. Use the Link(exLinkGroupBars) to group two linked bars, preventing their length, interval and so on.
exResizePercentBar	5	Resizes the percent value of the bar. The ItemBar(exBarPercent) specifies the value of the percent being displays in the percent bar.
exCreateBar	6	Creates a new bar. The AddBar method adds a new bar.
exResizeBaseLevel	7	Resizes a time scale unit in the base level area. The AllowInsizeZoom property specifies whether the user may resize different time-scale units, while other start unchanged. The Width property specifies the width of the inside zoom unit.
exBaseLevelDblClk	8	Magnifies a time scale unit by double clicking the base level area. The AllowInsizeZoom property specifies whether the user may resize different time-scale units, while other start unchanged.
exSelectDate	9	The user selects or unselects a date. The SelectDate property selects a date in the chart area. The SelectedDates property specifies the list of selected date-time units. The SelectLevel property specifies the index of the level in the chart that shows the selecting area in the chart.
		The user resizes the list/chart area using the control's vertical splitter. The vertical splitter may

exVSplitterChange	10	change the width of the items/chart area, so the PaneWidth property specifies the width of the columns / chart panel. The OnResizeControl property (exDisableSplitter) to specify whether the user can resize chart at runtime. The exVSplitterChange notification occurs only when the user start dragging the vertical splitter bar.
exHSplitterChange	11	The user resizes the list/chart area using the control's horizontal splitter (histogram). The HistogramVisible property specifies whether the chart's histogram is visible or hidden. The chart's horizontal splitter bar is visible only, if the HistogramVisible property is True. The OnResizeControl property (exDisableHistogram) to specify whether the user can resize the histogram at runtime. The exHSplitterChange notification occurs only when the user start dragging the horizontal splitter bar.
exPDM	12	Scheduling PDM operation is performed. The PDM or Precedence Diagramming Method is a tool for scheduling activities in a project plan. The SchedulePDM method schedules the bars using the PDM.
exResizeLevel	13	The user resizes the chart. The AllowResizeChart property of the Chart object specifies whether the user can enlarge (zoom-in,zoom-out) or resize the chart using the control's header, middle mouse button.
exResizeSelect	14	The user zooms the chart by right-selecting the overview-zoom part. The AllowOverviewZoom property of the Chart gets or sets a value that indicates whether the user can zoom and scale the chart at runtime.
exOSplitterChange	15	The user resizes the list/chart area using the control's horizontal splitter (overview). The OverviewVisible property specifies whether the control's overview is visible or hidden. Include the exOverviewSplitter flag into the OverviewVisible property so the user can resize the chart's overview at runtime by clicking and dragging.
		An Undo operation is performed, or the Undo

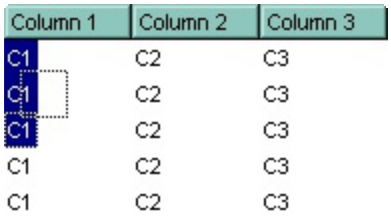
exUndo	17	method has been called. Use the UndoListAction property to list undo operations that can be performed. For instance, if the user presses the CTRL + Z and an Undo operation is available, this event is fired.
exRedo	18	An Undo operation is performed, or the Redo method has been called. Use the RedoListAction property to list redo operations that can be performed. For instance, if the user presses the CTRL + Y and a Redo operation is available, this event is fired.
exUndoRedoUpdate	16	The Undo/Redo queue is updated. Use the UndoListAction/RedoListAction property to list undo/redo operations that can be performed. The CanUndo property specifies whether an Undo operation is available. The CanRedo property specifies whether a Redo operation is available.
exSplitPaneChange	19	The user splits/resizes the chart's panel into multiple views.
exNoteChange	20	The user changes / moves a note.

constants CellSelectEnum

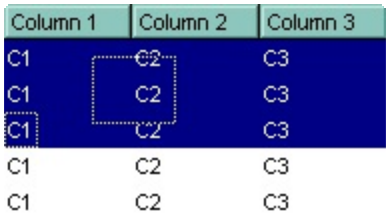
Specifies how the control selects cells or items within the control. Use the FullRowSelect property to enables full-row selection.

Name	Value	Description
exColumnSel	0	(**False**) Enables single-cell selection in the control.
exItemSel	-1	**(True**) Enables full-row selection in the control.
exRectSel	1	Enables rectangle selection in the control.

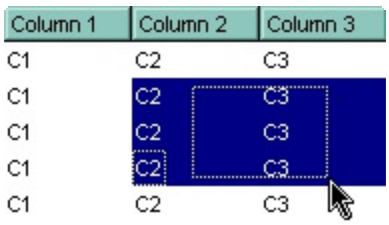
When the FullRowSelect property is **exColumnSel** the selection looks like:



When the FullRowSelect property is **exItemSel** the selection looks like:

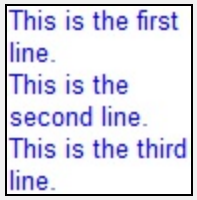
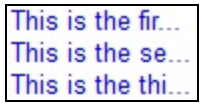


When the FullRowSelect property is **exRectSel** the selection looks like:



constants CellSingleLineEnum

The CellSingleLineEnum type defines whether the cell's caption is displayed on a single or multiple lines. The CellSingleLine property retrieves or sets a value indicating whether the cell is displayed using one line, or more than one line. The Def(exCellSingleLine) property specifies that all cells in the column display their content using multiple lines. The CellSingleLineEnum type supports the following values:

Name	Value	Description
exCaptionSingleLine	-1	Indicates that the cell's caption is displayed on a single line. In this case any \r\n or HTML tags is ignored. For instance the "This is the first line.\r\nThis is the second line.\r\nThis is the third line." shows as: 
exCaptionWordWrap	0	Specifies that the cell's caption is displayed on multiple lines, by wrapping the words. Any \r\n or HTML tag breaks the line. For instance the "This is the first line.\r\nThis is the second line.\r\nThis is the third line." shows as: 
exCaptionBreakWrap	1	Specifies that the cell's caption is displayed on multiple lines, by wrapping the breaks only. Only The \r\n or HTML tag breaks the line. For instance the "This is the first line.\r\nThis is the second line.\r\nThis is the third line." shows as: 

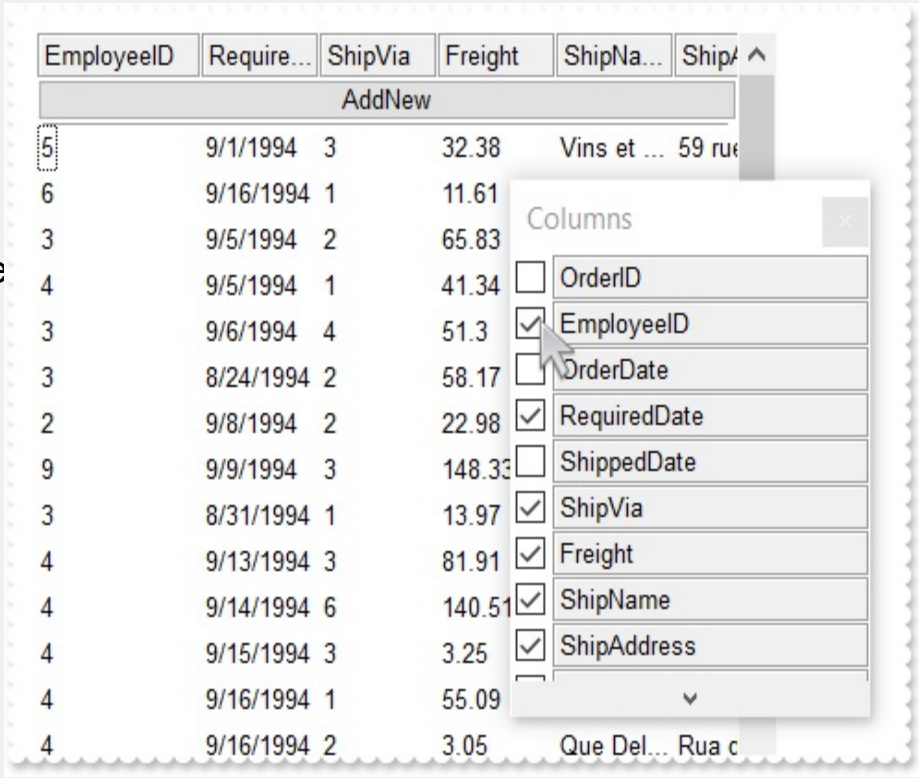
constants CheckStateEnum

Specifies the cell's state if CellHasCheckBox or CellHasRadioButton property is True.

Name	Value	Description
Unchecked	0	The cell is not checked.
Checked	1	The cell is checked.
PartialChecked	2	The cell is partially checked. To allow partially checks for a cell, the PartialCheck property should be True.

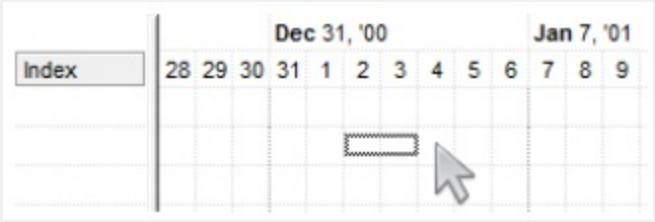
constants ColumnsFloatBarVisibleEnum

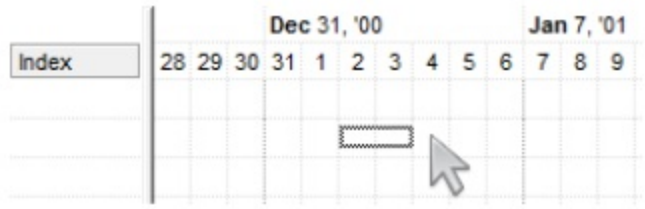
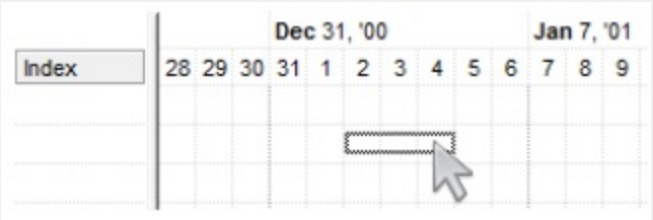
The ColumnsFloatBarVisibleEnum type specifies whether the control's Columns float-bar is visible or hidden. The ColumnsFloatBarVisibleEnum type supports the following values:

Name	Value	Description
exColumnsFloatBarHidden	0	Indicates that the control's Columns float-panel is not visible (hidden)
exColumnsFloatBarVisibleIncludeHiddenColumns	1	Specifies that the control's Columns float-panel shows only hidden-columns (dragable-columns only). The Visible property specifies whether the column is visible or hidden.
exColumnsFloatBarVisibleIncludeGroupableColumns	2	Specifies that the control's Columns float-panel shows only columns that can be group- by (dragable-columns only). The AllowGroupBy property specifies whether the column can be group-by.
exColumnsFloatBarVisibleIncludeCheckBoxes	4	Indicates that the control's Columns float-panel shows visible and hidden columns with a check-box associated (dragable-columns only), The Visible property specifies whether the column is visible or hidden. 

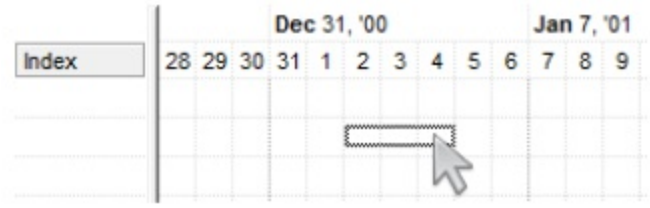
constants CreateBarEnum

The CreateBarEnum type specifies if a new bar is added automatically or manually during the CreateBar event. Use the AllowCreateBar property to let user creates new bars by selecting the bar's area at runtime. By default, the AllowCreateBar is exCreateBarManual.

Name	Value	Description
exNoCreateBar	0	The user can not create bars by drag an drop. Instead, if the AllowSelectObjects property is not-zero the user can select objects of the chart by drag and drop. No CreateBar event is fired.
exCreateBarManual	-1	The user can create new bars by drag and drop. If the user clicks the empty portion of the control (anywhere below the last visible item) the AddItem event occurs where the Item parameter of the event is a negative value and indicates the number of items to add to fill the empty space. The CreateBar event is fired and no bar is added, so you need to call AddBar method to add a new bar. 
exCreateBarAuto	1	The user can create automatically new bars by drag and drop. If the user clicks the empty portion of the control (anywhere below the last visible item) the control automatically adds new items to fill the empty space and so prior to CreateBar event, several AddItem event may occur. The CreateBar event is fired and a 'newbar' of 'Task' type is added. The exCreateBarAuto option creates empty bars (bars with zero-length) when user simple clicks the control (no drag). During the CreateEvent you can decide what to do with the newly created bar: keep, change or remove. Use the ItemBar property to change the key, the name or any other property of the newly created bar whose exBarKey property is "newbar" and it's exBarName is "Task". In other words, when this value is set, the control automatically adds a new bar to selected position,

	with the key 'newbar' that looks like this: . If the ChangeBar event is not handled, only a single bar is added to the same item, as the key is never changed. If you handle the CreateBar event and assign a different key for the newly created bar, multiple bars can be added at runtime. 
exCreateBarAutoEndInclusive 2	The user can create automatically new bars by drag and drop. The exCreateBarAutoEndInclusive value is similar with exCreateBarAuto, excepts that the end-margin of the bar to create, is defined by the end-date of the date-time unit from the cursor. If the user clicks and drags the bar over the empty portion of the control (anywhere below the last visible item) the control automatically adds new items to fill the empty space and so prior to CreateBar event, several AddItem event may occur. The CreateBar event is fired and a 'newbar' of 'Task' type is added. No items or bar are automatically created if the user just simple clicks the chart (no drag). 
	The user can create new bars by drag and drop. The exCreateBarManualEndInclusive value is similar with exCreateBarManual, excepts that the end-margin of the bar to draw, is defined by the end-date of the date-time unit from the cursor. If the user clicks and drags the bar over the empty portion of the control (anywhere below the last visible item) the AddItem event occurs where the Item parameter of the event is a negative value and indicates the number of items to add to fill the

exCreateBarManualEndInclusive2	empty space. The CreateBar event is fired and no bar is added, so you need to call AddBar method to add a new bar. No AddItem or CreateBar event is called if the user just simple clicks the chart (no drag).



constants DefColumnEnum

The Def property retrieves or sets a value that indicates the default value of given properties for all cells in the same column.

Name	Value	Description
exCellHasCheckBox	0	Assigns check boxes to all cells in the column, if it is True. Similar with the CellHasCheckBox property. By default, the exCellHasCheckBox property is False (0). *(Boolean expression)*
exCellHasRadioButton	1	Assigns radio buttons to all cells in the column, if it is True. Similar with the CellHasRadioButton property. By default, the exCellHasRadioButton property is False (0). *(Boolean expression)*
exCellHasButton	2	Specifies that all cells in the column are buttons, if it is True. Similar with the CellHasButton property. By default, the exCellHasButton property is False (0). *(Boolean expression)*
exCellButtonAutoWidth	3	Similar with the CellButtonAutoWidth property. By default, the exCellButtonAutoWidth property is False (0). The exCellButtonAutoWidth has effect only if the exCellHasButton option is True. *(Boolean expression)*
exCellBackColor	4	Specifies the background color for all cells in the column. Use the CellBackColor property to assign a background color for a specific cell. The property has effect only if the property is different than zero (default value). *(Color expression)*
		Specifies the foreground color for all cells in the

exCellForeColor	5	column. Use the CellForeColor property to assign a foreground color for a specific cell. The property has effect only if the property is different than zero (default value). *(Color expression)*
exCellVAlignment	6	Specifies the column's vertical alignment. By default, the Def(exCellVAlignment) property is exMiddle. Use the CellVAlignment property to specify the vertical alignment for a particular cell. By default, the exCellVAlignment property is MiddleAlignment (1). *(VAlignmentEnum expression)*
exHeaderBackColor	7	Specifies the column's header background color. Use this option to change the background color for a column in the header area. The exHeaderBackColor option supports skinning, so the last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. The property has effect only if the property is different than zero (default value). *(Color expression)*
exHeaderForeColor	8	Specifies the column's header background color. The property has effect only if the property is different than zero (default value). *(Color expression)*
exCellSingleLine	16	Specifies that all cells in the column displays its content into single or multiple lines. Similar with the CellSingleLine property. If using the CellSingleLine / Def(exCellSingleLine) property, we recommend to set the ScrollBySingleLine property on True so all items can be scrolled. *(CellSingleLineEnum type, previously Boolean expression)*

exCellValueFormat	17	The exCellValueFormat indicates that format to display all cells in the column such as text or HTML text. The CellValueFormat property specifies whether a particular cells displays text or HTML text. By default, the exCellValueFormat property is exText (0). *(ValueFormatEnum expression)*
exCellValueToItemBarProperty	18	Specifies the ItemBarPropertyEnum property of the bars being shown in the column. By default, the exCellValueToItemBarProperty is -1, so the cells are no associated with any bar. *The exCellValueToItemBarProperty has effect only if the AllowCellValueToItemBar property is True.* For instance, you can use the exCellValueToItemBarProperty and exCellValueToItemBarKey options to display the start and ending date of any bar in the item, in the entire column. The values in the cells are automatically changed once the bar is resized or moved and reverse, if the cell's value is changed the bar is moved or resized. The CellValueToItemBar method associates the cell's value with a property of the bar in the item. By default, the exCellValueToItemBarProperty is -1, which indicates that the column is not affiliated with any property of the bar. *(ItemBarPropertyEnum expression or -1)*
exCellValueToItemBarKey	19	Indicates the key of the bars whose property is shown in the column. *The exCellValueToItemBarKey has effect only if the AllowCellValueToItemBar property is True.* The exCellValueToItemBarProperty option indicates the property being displayed in the column, while the exCellValueToItemBarKey indicates the key of the bar in the item. The Key parameter of the AddBar indicates the key of the bar being added. For instance, you can use the exCellValueToItemBarProperty and

		exCellValueToItemBarKey options to display the start and ending date of any bar in the item, in the entire column. The values in the cells are automatically changed once the bar is resized or moved and reverse, if the cell's value is changed the bar is moved or resized. The CellValueToItemBar method associates the cell's value with a property of the bar in the item. *(Variant expression)*
exCellFormatLevel	32	Specifies the format layout for the cells. The CellFormatLevel property indicates the format layout for a specified cell. Use the FormatLevel property to specify the layout of the column in the control's header bar. *(CRD string expression)*
exCellDrawPartsOrder	34	Specifies the order of the drawing parts for the entire column. By default, this option is "check,icon,icons,picture,caption", which means that the cell displays its parts in the following order: check box/ radio buttons (CellHasCheckBox/CellRadioButton), single icon (CellImage), multiple icons (CellImages), custom size picture (CellPicture), and the cell's caption. Use the exCellDrawPartsOrder option to specify a new order for the drawing parts in the cells of the column. The RightToLeft property automatically flips the order of the columns. By default, the exCellDrawPartsOrder property is "check,icon,icons,picture,caption". *(String expression)*
exCellPaddingLeft	48	The padding defines the space between the element border and the element content. Gets or sets the left padding (space) of the cells within the column. This option applies a padding to all cells in the column. Use the exHeaderPaddingLeft option to apply the padding to the column's caption in the control's header. The padding does not affect the

		element's background color. By default, the exCellPaddingLeft property is 0. *(Long expression)*
exCellPaddingRight	49	Gets or sets the right padding (space) of the cells within the column. This option applies a padding to all cells in the column. Use the exHeaderPaddingRight option to apply the padding to the column's caption in the control's header. The padding does not affect the element's background color. By default, the exCellPaddingRight property is 0. *(Long expression)*
exCellPaddingTop	50	Gets or sets the top padding (space) of the cells within the column. This option applies a padding to all cells in the column. Use the exHeaderPaddingTop option to apply the padding to the column's caption in the control's header. The padding does not affect the element's background color. By default, the exCellPaddingTop property is 0. *(Long expression)*
exCellPaddingBottom	51	Gets or sets the bottom padding (space) of the cells within the column. This option applies a padding to all cells in the column. Use the exHeaderPaddingBottom option to apply the padding to the column's caption in the control's header. The padding does not affect the element's background color. By default, the exCellPaddingBottom property is 0. *(Long expression)*
exHeaderPaddingLeft	52	Gets or sets the left padding (space) of the column's header. This option applies the padding to the column's caption in the control's header. Use the exCellPaddingLeft option to apply the padding to all cells in the column. The padding does not affect the element's background color. By default, the

		exHeaderPaddingLeft property is 0. *(Long expression)*
exHeaderPaddingRight	53	Gets or sets the right padding (space) of the column's header. This option applies the padding to the column's caption in the control's header. Use the exCellPaddingRight option to apply the padding to all cells in the column. The padding does not affect the element's background color. By default, the exHeaderPaddingRight property is 0. *(Long expression)*
exHeaderPaddingTop	54	Gets or sets the top padding (space) of the column's header. This option applies the padding to the column's caption in the control's header. Use the exCellPaddingTop option to apply the padding to all cells in the column. The padding does not affect the element's background color. By default, the exHeaderPaddingTop property is 0. *(Long expression)*
exHeaderPaddingBottom	55	Gets or sets the bottom padding (space) of the column's header. This option applies the padding to the column's caption in the control's header. Use the exCellPaddingBottom option to apply the padding to all cells in the column. The padding does not affect the element's background color. By default, the exHeaderPaddingBottom property is 0. *(Long expression)*
exColumnResizeContiguously	64	exColumnResizeContiguously. Gets or sets a value that indicates whether the control's content is updated while the user is resizing the column.

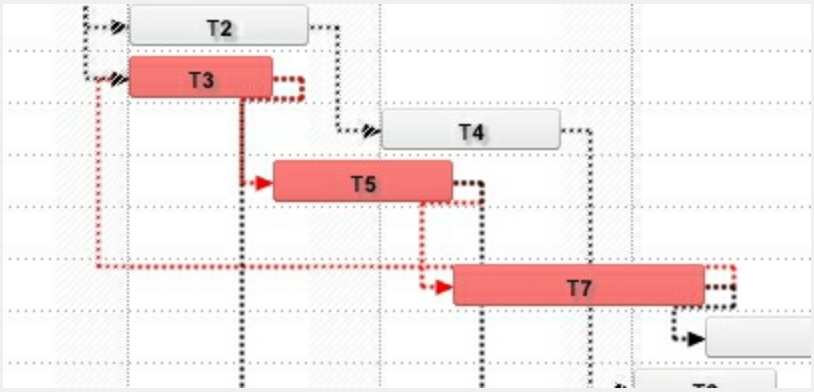
constants DefSchedulePDMEnum

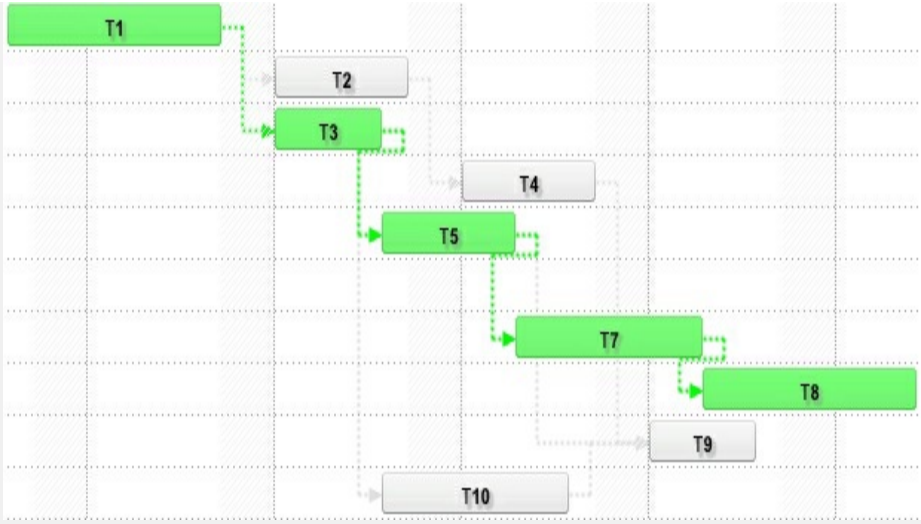
The DefSchedulePDMEnum type defines options to be used by the SchedulePDM method. The DefSchedulePDM property indicates the default options to be used by the next call of the SchedulePDM method. In other words, we will recommend calling/setting the DefSchedulePDM property before calling the SchedulePDM method.

The DefSchedulePDMEnum type defines the following values:

Name	Value	Description
exPDMScheduleType	0	Specifies the type of scheduling the next SchedulePDM call executes. The valid values are 0, 1, 2 as explained bellow: • **0**, (default). exPDMScheduleDate has no effect. The start and end of the project is not fixed, so once a bar is moved the starting or ending point of the project may be variable. • **1**, exPDMScheduleDate indicates the start of the project. If this option is set, the next call of the SchedulePDM consider this value as being the start of the project, and so all the bars starts scheduling using this date. If DefSchedulePDM(exPDMScheduleType) property is 1, the DefSchedulePDM(exPDMScheduleDate) property indicates the date to start the project. • **2**, exPDMScheduleDate indicates the end of the project. If this option is set, the next call of the SchedulePDM consider this value as being the end of the project, and so all the bars starts scheduling using this date as the end of the project, so where the project should end. If DefSchedulePDM(exPDMScheduleType) property is 2, the DefSchedulePDM(exPDMScheduleDate) property indicates the date to end the project. *(long expression)*
		Specifies the date to start or end the project. This option is valid only if the exPDMScheduleType is 1 or 2.

exPDMScheduleDate	1	For instance the following sample specifies the start of the project to be 1/8/2001, With G2antt1 With .Items **.DefSchedulePDM(exPDMScheduleType) = 1** **.DefSchedulePDM(exPDMScheduleDate) = #1/8/2001#** .SchedulePDM 0,"K1" End With End With and the following sample specifies the end of the project to be 1/8/2001 With G2antt1 With .Items **.DefSchedulePDM(exPDMScheduleType) = 2** **.DefSchedulePDM(exPDMScheduleDate) = #1/8/2001#** .SchedulePDM 0,"K1" End With End With *(date expression)*
exPDMErrorColor	2	Specifies the color to show the error nodes. This property can be used to highlight the nodes that makes the layout impossible to complete. This property is ignored if the exPDMErrorColor is 0. *(color expression)*
		Specifies the color to show the cycling nodes. This property can be used to highlight the nodes that makes the layout impossible to complete. This property is ignored if the exPDMCycleColor is 0.

exPDMCycleColor	3	Set the exPDMCallHasCycle property on True, so the chart displays the entire cycle, not just the node that fails to be arranged in the layout. The following screen shot shows in red the cycle that fails to layout the chart based on the links:  *(color expression)*
exPDMCallHasCycle	4	Indicates whether the control shows the entire cycle, so the layout can not be completed. By default, the exPDMCallHasCycle property is False, which indicates that the control shows no cycles, if the layout fails to complete. *(boolean expression)*
exPDMCriticalPathBarColor	5	Indicates the color to show the activities on the critical path. By default, the exPDMCriticalPathBarColor is 0. This property is ignored if the exPDMCriticalPathBarColor is 0. The exPDMCriticalPathBarColor color is applied to the node, only if it has no color associated using the Items.ItemBar(exBarColor) (Items.ItemBar(exBarColor) property is 0). The Items.ItemBar(exBarCriticalPath) property on True specifies whether the bar makes part of the critical path. The following screen shot shows in green the critical path (T1, T3, T5, T7 and T8) for the layout:

	 *(color expression)*
exPDMCriticalPathOffBarColor 6	Indicates the color to show the activities off the critical path. By default, the exPDMCriticalPathOffBarColor is 0. This property is ignored if the exPDMCriticalPathOffBarColor is 0. The exPDMCriticalPathOffBarColor color is applied to the node, only if it has no color associated using the Items.ItemBar(exBarColor) (Items.ItemBar(exBarColor) property is 0). The Items.ItemBar(exBarCriticalPath) property on False specifies whether the bar is off of the critical path. *(color expression)*
exPDMCriticalPathLinkColor 7	Indicates the color to show the links on the critical path. By default, the exPDMCriticalPathLinkColor is 0. This property is ignored if the exPDMCriticalPathLinkColor is 0. *(color expression)*
exPDMCriticalPathOffLinkColor 8	Indicates the color to show the links off of the critical path. By default, the exPDMCriticalPathOffLinkColor is 0. This property is ignored if the exPDMCriticalPathOffLinkColor is 0. *(color expression)*

exPDMCriticalPathLenMethod 9	Determines the way how the length of the path is computed. *(long expression)*

constants DescriptionTypeEnum

The control's Description property defines descriptions for few control parts.

Name	Value	Description
exFilterBarAll	0	Defines the caption of (All) in the drop down filter window. If the Description(exFilterBarAll) property is empty, the (All) predefined item is not shown in the drop down filter window.
exFilterBarBlanks	1	Defines the caption of (Blanks) in the drop down filter window. If the Description(exFilterBarBlanks) property is empty, the (Blanks) predefined item is not shown in the drop down filter window. The (Blanks) option is displayed in the drop down filter window only if the FilterList property includes the exShowBlanks flag.
exFilterBarNonBlanks	2	Defines the caption of (NonBlanks) in the drop down filter window. If the Description(exFilterBarNonBlanks) property is empty, the (NonBlanks) predefined item is not shown in the drop down filter window. The (Blanks) option is displayed in the drop down filter window only if the FilterList property includes the exShowBlanks flag.
exFilterBarFilterForCaption	3	Defines the caption of "Filter For:" in the drop down filter window
exFilterBarFilterTitle	4	Defines the title for the filter tooltip. The tooltip is shown only if the FilterList property includes the exEnableToolTip flag.
exFilterBarPatternFilterTitle	5	Defines the title for the filter pattern tooltip. The tooltip is shown only if the FilterList property includes the exEnableToolTip flag.
exFilterBarTooltip	6	Defines the tooltip for filter window. The tooltip is shown only if the FilterList property includes the exEnableToolTip flag.
exFilterBarPatternTooltip	7	Defines the tooltip for filter pattern window. The tooltip is shown only if the FilterList property includes the exEnableToolTip flag.
exFilterBarFilterForTooltip	8	Defines the tooltip for "Filter For:" window. The tooltip is shown only if the FilterList property

		includes the exEnableToolTip flag.
exFilterBarIsBlank	9	Defines the caption of the function 'IsBlank' in the control's filter bar.
exFilterBarIsNonBlank	10	Defines the caption of the function 'not IsBlank' in the control's filter bar.
exFilterBarAnd	11	Customizes the ' and ' text in the control's filter bar when multiple columns are used to filter the items in the control.
exFilterBarDate	12	Specifies the "Date:" caption being displayed in the drop down filter window when DisplayFilterPattern property is True, and DisplayFilterDate property is True.
exFilterBarDateTo	13	Specifies the "to" sequence being used to split the from date and to date in the Date field of the drop down filter window. For instance, the "to 12/13/2004" specifies the items before 12/13/2004, "12/23/2004 to 12/24/2004" filters the items between 12/23/2004 and 12/24/2004, or "Feb 12 2004 to" specifies all items after a date.
exFilterBarDateTooltip	14	Describes the tooltip that shows up when cursor is over the Date field. "You can filter the items into a given interval of dates. For instance, you can filter all items dated before a specified date (**to 2/13/2004**), or all items dated after a date (**Feb 13 2004 to**) or all items that are in a given interval (**2/13/2004 to 2/13/2005**)." The tooltip is shown only if the FilterList property includes the exEnableToolTip flag.
exFilterBarDateTitle	15	Describes the title of the tooltip that shows up when the cursor is over the Date field. By default, the exFilterBarDateTitle is "Date". The tooltip is shown only if the FilterList property includes the exEnableToolTip flag.
exFilterBarDateTodayCaption	16	Specifies the caption for the 'Today' button in a date filter window. By default, the exFilterBarDateTodayCaption property is "Today".
exFilterBarDateMonths	17	Specifies the name for months to be displayed in a date filter window. The list of months should be delimitated by space characters. By default, the

		exFilterBarDateMonths is "January February March April May June July August September October November December".
exFilterBarDateWeekDays	18	Specifies the shortcut for the weekdays to be displayed in a date filter window. The list of shortcut for the weekdays should be separated by space characters. By default, the exFilterBarDateWeekDays is "S M T W T F S". The first shortcut in the list indicates the shortcut for the Sunday, the second shortcut indicates the shortcut for Monday, and so on.
exFilterBarChecked	19	Defines the caption of (Checked) in the filter bar window. The exFilterBarChecked option is displayed only if the FilterType property is exCheck. If the Description(exFilterBarChecked) property is empty, the (Checked) predefined item is not shown in the drop down filter window. If the user selects the (Checked) item the control filter checked items. The CellState property indicates the state of the cell's checkbox.
exFilterBarUnchecked	20	Defines the caption of (Unchecked) in the filter bar window. The exFilterBarUnchecked option is displayed only if the FilterType property is exCheck. If the Description(exFilterBarUnchecked) property is empty, the (Unchecked) predefined item is not shown in the drop down filter window. If the user selects the (Unchecked) item the control filter unchecked items. The CellState property indicates the state of the cell's checkbox.
exFilterBarIsChecked	21	Defines the caption of the 'IsChecked' function in the control's filter bar. The 'IsChecked' function may appear only if the user selects (Checked) item in the drop down filter window, when the FilterType property is exCheck.
exFilterBarIsUnchecked	22	Defines the caption of the 'not IsChecked' function in the control's filter bar. The 'not IsChecked' function may appear only if the user selects (Unchecked) item in the drop down filter window, when the FilterType property is exCheck.
exFilterBarOr	23	Customizes the 'or' operator in the control's filter bar when multiple columns are used to filter the

		items in the control.
exFilterBarNot	24	Customizes the 'not' operator in the control's filter bar.
exFilterBarExclude	25	Specifies the 'Exclude' caption being displayed in the drop down filter. The Exclude option is displayed in the drop down filter window only if the FilterList property includes the exShowExlcude flag.
exColumnsFloatBar	26	Specifies the caption to be shown on control's Columns float bar.

constants DividerAlignmentEnum

Defines the alignment for a divider line into a divider item. Use the ItemDividerLineAlignment property to align the line in a divider item. Use the ItemDivider property to add a divider item

Name	Value	Description
DividerBottom	0	The divider line is displayed on bottom side of the item.
DividerCenter	1	The divider line is displayed on center of the item.
DividerTop	2	The divider line is displayed at the top of the item.
DividerBoth	3	The divider line is displayed at the top and bottom of the item.

constants DividerLineEnum

Defines the type of divider line. The ItemDividerLine property uses the DividerLineEnum type.

Name	Value	Description
EmptyLine	0	No line.
SingleLine	1	Single line
DoubleLine	2	Double line
DotLine	3	Dotted line
DoubleDotLine	4	DoubleDottted line
ThinLine	5	Thin line
DoubleThinLine	6	Double thin line

constants DrawPartEnum

The DrawPartEnum type specifies the identifier of parts that can be overridden using the BeforeDrawPart and AfterDrawPart events. Currently, the supported values are:

Name	Value	Description
exOwnerDrawBar	0	Specifies that the "OwnerDraw" bar is drawing. Use the Add or Copy method to add a "OwnerDraw" bar. The DrawPartItem property specifies the handle of the item that hosts the "OwnerDraw" bar. The DrawPartKey property specifies the key of the bar to be painted. Use the ItemBar(DrawPartItem,DrawPartKey) property to access properties of the drawing bar.
exDrawLeftHistogram	1	Specifies the left part of the histogram. The area is shown in the histogram part just bellow the list/items area. Use the HistogramRulerLinesColor property to automatically show the ruler in the left part of the histogram. Use the HistogramCumulativeShowLegend property to automatically show the legend for items being included in the histogram when cumulative colors are shown. The control fires the HistogramBoundsChanged event when the bound of the left histogram part is changed, so you can automatically updates the position for inside controls if you display them on the histogram part.
exDrawRightHistogram	2	Specifies the right part of the histogram. The area is shown in the histogram part just bellow the chart area. The right part of the histogram shows the chart's histogram when HistogramVisible property is True and the **HistogramPattern or/and HistogramColor** property is set.

constants DropDownWidthType

The DropDownWidthType expression specifies the width of the drop down portion of an editor. The DropDownAutoWidth property specifies the width of the drop down portion of the editor. The DropDownMinWidth property specifies the minimum width for the drop down portion.

Name	Value	Description
exDropDownAutoWidth	-1	The drop down width is automatically computed to let all predefined items in the editor fi the drop down portion.
exDropDownEditorWidth	0	The width of the drop down portion of the editor is specified by the width of the cell that holds the editor.
exDropDownAutoEditorWidth	1	The width of the drop down portion of the editor is specified by the width of the cell that holds the editor. The width of the drop down can't be less than the width required to let all predefined items being visible. The width of the drop down portion is always greater than the DropDownMinWidth value.

constants EditorOptionEnum

Specifies different options for a built-in editor. The Option property specifies the editor's options. Use the DefaultEditorOption property to specify default option for the editors of a specified type. The following options are supported:

Name	Value	Description
exMemoHScrollBar	1	Adds the horizontal scroll bar to a MemoType or MemoDropDownType editor. *(boolean expression, by default it is false)*
exMemoVScrollBar	2	Adds the vertical scroll bar to a MemoType or MemoDropDownType editor. *(boolean expression, by default it is false)*
exMemoAutoSize	3	Specifies whether the MemoType editor is resized when user alters the text. *(boolean expression, by default it is true)*
exColorListShowName	4	Specifies whether a ColorListType editor displays the name of the color. *(boolean expression, by default it is false)*
exColorShowPalette	5	Specifies whether the ColorList editor displays the palette colors list. *(boolean expression, by default it is true)*
exColorShowSystem	6	Specifies whether the ColorType editor shows the system colors list. *(boolean expression, by default it is true)*
exMemoDropDownWidth	7	Specifies the width for a MemoDropDownType editor. *(long expression, by default it is 128)*

exMemoDropDownHeight	8	Specifies the height for a MemoDropDownType editor. *(long expression, by default it is 116)*
exMemoDropDownAcceptReturn	9	Specifies whether the Return key is used to add new lines into a MemoDropDownType editor. *(boolean expression, by default it is true)*
exEditRight	10	Right-aligns text in a single-line or multiline edit control. *(boolean expression, by default it is false)*
exProgressBarBackColor	11	Specifies the background color for a progress bar editor. Use the exProgressBarMarkTicker option to specify the background color or visual appearance of the progress bar. *(color expression, by default it is 0x80000000 | COLOR_HIGHLIGHT)*
exProgressBarAlignment	12	Specifies the alignment of the caption inside of a progress bar editor. *(AlignmentEnum expression, by default it is LeftAlignment)*
exProgressBarMarkTicker	13	Retrieves or sets a value that indicates whether the ticker of a progress bar editor is visible or hidden. If value is 0 (false), no progress's background is shown. If -1(true), the progress's background is shown using the current visual theme, else the solid color or the EBN object is applied on the progress's background. *(color expression, by default it is -1)*
exDateAllowNullDate	14	Allows you to specify an empty date to a DateType editor. *(boolean expression, by default it is true)*

exCheckValue0	15	Specifies the check box state being displayed for unchecked state. *(long expression, valid values are 0, 1 or 2, by default it is 0)*
exCheckValue1	16	Specifies the check box state being displayed for checked state. *(long expression, valid values are 0, 1 or 2, by default it is 1)*
exCheckValue2	17	Specifies the check box state being displayed for partial checked state. (long expression, valid values are 0, 1 or 2). For instance, if your cells load boolean values (True is -1, False is 0), the control displays the partial-check icon for True values. You can call G2antt1.DefaultEditorOption(exCheckValue2) = 1 before loading the CheckValueType editor, and so the partial-check cells show as check icons. *(long expression, valid values are 0, 1 or 2, by default it is 2)*
exEditPassword	18	Specifies a value that indicates whether an edit control displays all characters as an asterisk (*) as they are typed (passwords). *(boolean expression, by default it is false)*
exEditPasswordChar	19	Specifies a value that indicates the password character. *(character expression, by default it is '*')*
		(VK_LEFT) Specifies whether the left arrow key is handled by the control or by the editor. By default, the Option(exLeftArrow) property is exHandleControl. Use the exLeftArrow option to disable focusing a new cell if the user presses the

exLeftArrow	20	left arrow key while editing. The option is valid for all editors. *(ArrowHandleEnum expression, by default it is exHandleControl)*
exRightArrow	21	(VK_RIGHT) Specifies whether the right arrow key is handled by the control or by the editor. By default, the Option(exRightArrow) property is exHandleControl. Use the exRightArrow option to disable focusing a new cell if the user presses the right arrow key while editing. The option is valid for all editors. *(ArrowHandleEnum expression, by default it is exHandleControl)*
exUpArrow	22	(VK_UP) Specifies whether the up arrow key is handled by the control or by the editor. By default, the Option(exUpArrow) property is exHandleControl. Use the exUpArrow option to disable focusing a new cell if the user presses the up arrow key while editing. The option is valid for all editors. *(ArrowHandleEnum expression, by default it is exHandleControl)*
exDownArrow	23	(VK_DOWN) Specifies whether the down arrow key is handled by the control or by the editor. By default, the Option(exDownArrow) property is exHandleControl. Use the exDownArrow option to disable focusing a new cell if the user presses the down arrow key while editing. The option is valid for all editors. *(ArrowHandleEnum expression, by default it is exHandleControl)*
		(VK_HOME) Specifies whether the home key is handled by the control or by the current editor. By default, the Option(exHomeKey) property is True.

exHomeKey	24	Use the exHomeKey option to disable focusing a new cell if the user presses the home key while editing. The option is valid for all editors. *(boolean expression, by default it is true)*
exEndKey	25	(VK_END) Specifies whether the end key is handled by the control or by the current editor. By default, the Option(exEndKey) property is True. Use the exEndKey option to disable focusing a new cell if the user presses the end key while editing. The option is valid for all editors. *(boolean expression, by default it is true)*
exPageUpKey	26	(VK_PRIOR) Specifies whether the page up key is handled by the control or by the current editor. By default, the Option(exPageUpKey) property is True. Use the exPageUpKey option to disable focusing a new cell if the user presses the page up key while editing. The option is valid for all editors. *(boolean expression, by default it is true)*
exPageDownKey	27	(VK_NEXT) Specifies whether the page down key is handled by the control or by the current editor. By default, the Option(exPageDownKey) property is True. Use the exPageDownKey option to disable focusing a new cell if the user presses the page down key while editing. The option is valid for all editors. *(boolean expression, by default it is true)*
exDropDownImage	28	Displays the predefined icon in the control's cell, if the user selects an item from a drop down editor. By default, the exDropDownImage property is True. The option is valid for DropDownListType, PickEdit and ColorListType editors. *(boolean expression, by default it is true)*
		Specifies the caption for the 'Today' button in a

exDateTodayCaption	29	DateType editor. *(string expression, by default it is "Today")*
exDateMonths	30	Specifies the name for months to be displayed in a DateType editor. The list of months should be delimitated by spaces. *(string expression, by default it is "January February March April May June July August September October November December")*
exDateWeekDays	31	Specifies the shortcut for the weekdays to be displayed in a DateType editor. The list of shortcut for the weekdays should be separated by spaces. The first shortcut in the list indicates the shortcut for the Sunday, the second shortcut indicates the shortcut for Monday, and so on. *(string expression, by default it is ""S M T W T F S")*
exDateFirstWeekDay	32	Specifies the first day of the week in a DateType editor. The valid values for the Editor.Option(exDateFirstWeekDay) property are like follows: 0 - Sunday, 1 - Monday, 2 - Tuesday, 3 - Wednesday, 4 - Thursday, 5 - Friday and 6 - Saturday. *(long expression, valid values are 0 to 6, by default it is 0)*
exDateShowTodayButton	33	Specifies whether the 'Today' button is visible or hidden in a DateType editor. *(boolean expression, by default it is true)*
exDateMarkToday	34	Gets or sets a value that indicates whether the today date is marked in a DateType editor. *(boolean expression, by default it is false)*

exDateShowScroll	35	Specifies whether the years scroll bar is visible or hidden in a DateType editor. *(boolean expression, by default it is true)*
exEditLimitText	36	Limits the length of the text that the user may enter into an edit control. By default, the Editor.Option(exEditLimitText) is zero, and so no limit is applied to the edit control. *(long expression, by default it is 0)*
exAutoDropDownList	37	The exAutoDropDownList has no effect Editor.Option(exAutoDropDownList) property is 0 (default). Automatically shows the drop down list when user starts typing characters into a DropDownList editor, if the Editor.Option(exAutoDropDownList) property is -1. If the Editor.Option(exAutoDropDownList) property is +1, the control selects a new item that matches typed characters without opening the drop down portion of the editor. *(long expression, valid values are -1, 0 and +1, by default it is 0)*
exExpandOnSearch	38	Expands items while user types characters into a drop down editor. The exExpandOnSearch type has effect for drop down type editors. *(boolean expression, by default it is false)*
exAutoSearch	39	Specifies the kind of searching while user types characters within the drop down editor. The exExpandOnSearch type has effect for drop down type editors. *(AutoSearchEnum expression, valid values are 0 and 1, by default it is exStartWith)*
		Specifies the proposed change when user clicks a spin control. The exSpinStep should be a positive number, else clicking the spin has no effect. Integer

exSpinStep	40	or floating points allowed as well. For instance, if the exSpinStep is 0.01, the proposed change when user clicks the spin is 0.01. If the exSpinStep property is 0, the spin control is hidden (useful if you have a slider control). *(positive numeric expression, by default it is 1)*
exSliderWidth	41	Specifies the width in pixels of the slider control. The exSliderWidth value could be 0, when the slider control is hidden, a positive value that indicates the width in pixels of the slider in the control, a negative number when its absolute value indicates the percent of the cell's size being used by the slider. For instance, Option(exSliderWidth) = 0, hides the slider, Option(exSliderWidth) = 100, shows a slider of 100 pixels width, Option(exSliderWidth) = -50, uses half of the cell's client area to display a slider control. By default the Option(exSliderWidth) property is 64 pixels. Use the exSpinStep to hide the spin control. *(long expression, by default it is 64)*
exSliderStep	42	Specifies a value that represents the proposed change in the slider control's position. The exSliderTickFrequency property specifies the frequency to display ticks on a slider control. The exSliderMin and exSliderMax determines the range of values for the slider control. *(numeric expression , by default it is 1)*
exSliderMin	43	Specifies the slider's minimum value. *(numeric expression, by default it is 0)*
exSliderMax	44	Specifies the slider's maximum value. *(numeric expression, by default it is 100)*
		Keeps the selection background color while the

exKeepSelBackColor	45	editor is visible. The exKeepSelBackColor option is valid for all editors. Use the exKeepSelBackColor to let the editor to display the control's selection background color when it is visible. *(boolean expression, by default it is false)*
exEditDecimalSymbol	46	Specifies the symbol that indicates the decimal values while editing a floating point number. Use the exEditDecimaSymbol option to assign a different symbol for floating point numbers, when Numeric property is exFloat. *(long expression, that indicates the ASCII code for the character being used as decimal symbol, by default, it is the "Decimal symbol" settings as in the Regional Options, in your control panel)*
exDateWeeksHeader	47	Sets or gets a value that indicates whether the weeks header is visible or hidden in a DateType editor. *(boolean expression, by default it is false)*
exEditSelStart	48	Sets the starting point of text selected, when an EditType editor is opened. If the exEditSelStart property is 0, the text gets selected from the first character. If the exEditSelStart property is -1, the cursor is placed at the end of the text. *(long expression, by default it is 0)*
exEditSelLength	49	Sets the number of characters selected, when an EditType editor is opened. If the exEditSelLength is 0, no text is selected, instead the exEditSelStart changes the position of the cursor. If the exEditSelLength property is -1, the text from the exEditSelStart position to the end gets selected. *(long expression, by default it is -1)*
		Specifies the background color for a locked edit

exEditLockedBackColor	50	control. By default, the exEditLockedBackColor property is a system color that indicates the face color for three-dimensional display elements and for dialog box backgrounds. *(color expression, by default it is 0x80000000 | COLOR_3DFACE)*
exEditLockedForeColor	51	Specifies the foreground color for a locked edit control. *(color expression, by default it is 0)*
exShowPictureType	52	Specifies whether a PictureType editor displays the type of the picture. *(boolean expression, by default it is true)*
exSliderTickFrequency	53	Gets or sets the interval between tick marks slider types. By default, the exSliderTickFrequency property is 0 which makes the slider to display no ticks. The exSliderTickFrequency property specifies the frequency to display ticks on a slider control. The exSliderStep proposed change in the slider control's position. The exSliderMin and exSliderMax determines the range of values for the slider control. The exSliderWidth option specifies the width of the slider within the cell. *(numeric expression, by default it is 0)*
exPickAllowEmpty	54	Specifies whether the editor of PickEditType supports empty value. *(boolean expression, by default it is false)*
exDropDownBackColor	55	Specifies the drop down's background color. If 0 the exDropDownBackColor has no effect. *(color expression, by default it is 0)*
		Specifies the drop down's foreground color. If 0 the

exDropDownForeColor	56	exDropDownBackColor has no effect. *(color expression, by default it is 0)*
exDropDownColumnCaption	57	Specifies the HTML caption for each column within the drop down list, separated by Ś character (vertical broken bar, ALT + 221). For instance, "<sha ;;0>Name</sha>Ś<sha ;;0>ID</sha>" defines two columns for the drop down editor. The header of the drop down list is visible, if the exDropDownColumnCaption is not empty. *(string expression, by default it is "")*
exDropDownColumnWidth	58	Specifies the width for each column within the drop down list, separated by Ś character (vertical broken bar, ALT + 221). For instance, "Ś32" defines the width of the second column to 32 pixels, within a drop down multiple columns editor. *(string expression, by default it is "")*
exDropDownColumnPosition	59	Specifies the position for each column within the drop down list, separated by Ś character (vertical broken bar, ALT + 221). For instance, "Ś0" defines sets the second column to be first visible-column, within a drop down multiple columns editor. *(string expression, by default it is "")*
exDropDownColumnAutoResize	60	Specifies whether the drop down list resizes automatically its visible columns to fit the drop down width. Specifies whether the drop down multiple columns editor displays horizontal-scroll bar. *(boolean expression, by default it is true)*
exSliderTickStyle	63	exSliderTickStyle. Gets or sets the style to display the slider' ticks.
exCalcExecuteKeys	100	Specifies whether the calculator editor executes the keys while focused and the drop down portion is hidden.

		(boolean expression, by default it is true)
exCalcCannotDivideByZero	101	Specifies the message whether a division by zero occurs in a calendar editor. *(string expression, by default it is "Cannot divide by zero.")*
exCalcButtonWidth	102	Specifies the width in pixels of the buttons in the calculator editor. *(long expression, by default it is 24)*
exCalcButtonHeight	103	Specifies the height in pixels of the buttons in the calculator editor. *(long expression, by default it is 24)*
exCalcButtons	104	Specifies buttons in a calendar editor. The property specifies the buttons and the layout of the buttons in the control. A string expression that indicates the list of buttons being displayed. The rows are separated by chr(13)+chr(10) (vbCrLf) sequence, and the buttons inside the row are separated by ';' character. *(string expression)*
exCalcPictureUp	105	Specifies the picture when the button is up in a drop down calendar editor. A Picture object that indicates the node's picture. *(A Picture object that implements IPicture interface, a string expression that indicates the base64 encoded string that holds a picture object (use the eximages tool to save your picture as base64 encoded format, by default it is "")*
		Specifies the picture when the button is down in a drop down calendar editor. A Picture object that indicates the node's picture.

exCalcPictureDown	106	*(A Picture object that implements IPicture interface, a string expression that indicates the base64 encoded string that holds a picture object (use the eximages tool to save your picture as base64 encoded format, by default it is "")*
exEditAllowOverType	200	Specifies whether the editor supports overtype mode. The option is valid for EditType and MemoType editors. *(boolean expression, by default it is false)*
exEditOverType	201	Retrieves or sets a value that indicates whether the editor is in insert or overtype mode. The option is valid for EditType and MemoType editors. *(boolean expression, by default it is false)*
exEditAllowContextMenu	202	Specifies whether the editor displays the edit's default context menu when the user right clicks the field. *(boolean expression, by default it is true)*

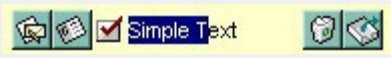
constants EditorVisibleEnum

The EditorVisibleEnum type specifies the way the editor is shown/hidden on the cell. The CellEditorVisible property specifies whether the cell's editor is visible or hidden. The EditorVisibleEnum type supports the following values:

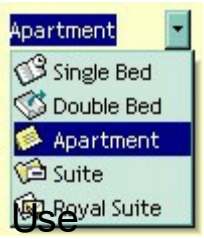
Name	Value	Description
exEditorHidden	0	The editor is hidden.
exEditorVisible	1	The editor is always visible.
exEditorVisibleOnFocus	-1	The editor is visible when the cell receives the focus.

constants EditTypeEnum

Use the EditType property to specify the editor for a cell or a column. Any editor can have a check box (use CellHasCheckBox property) , radio button (use CellHasRadioButton property) associated, or multiple buttons to the left or right side (use AddButton method). The Mask property is applied to most of all editors that has associated a standard edit control. Use the Option property to assign different options for a given editor. Use the DefaultEditorOption property to specify default option for the editors of a specified type. The CellValue property indicates the value for the editor. A cell or a column supports the following type of editors:

Name	Value	Description
ReadOnly	0	The column or the cell has no editor associated.
EditType	1	A standard text edit field.  The editor supports the following options: • exEditRight, Right-aligns text in a single-line or multiline edit control. • exEditPassword, Specifies a value that indicates whether an edit control displays all characters as an asterisk (*) as they are typed (passwords). • exEditPasswordChar, Specifies a value that indicates the password character. • exEditLimitText, Limits the length of the text that the user may enter into an edit control. • exEditDecimalSymbol, Specifies the symbol that indicates the decimal values while editing a floating point number. The Numeric property should be on exFloat. • exEditSelStart, Sets the starting point of text selected, when an EditType editor is opened. • exEditSelLength, Sets the number of characters selected, when an EditType editor is opened. • exEditLockedBackColor property. Specifies the background color for a locked edit control. • exEditLockedForeColor property. Specifies the foreground color for a locked edit control.

It provides an intuitive interface for your



users to select values from pre-defined lists presented in a drop-down window, but it accepts new values at runtime too. The DropDownType editor has associated a standard text edit field too. Use AddItem or InsertItem method to add predefined values to the drop down list. The DropDownRows property specifies the maximum number of visible rows into the drop-down list. The editor displays the CellValue value, not the identifier of the selected item. The EditType options are supported too.

The following sample adds a column with a DropDownType editor:

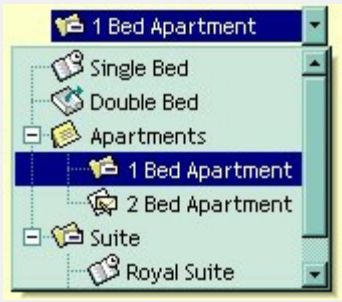
DropDownType	2	With .Columns.Add("Editor").Editor .EditType = DropDownType .AddItem 0, "Single Bed", 1 .AddItem 1, "Double Bed", 2 .AddItem 2, "Apartment", 3 .AddItem 3, "Suite", 4 .AddItem 4, "Royal Suite", 5 End With .Items.CellValue(.Items(0), "Editor") = "Apartment"

The editor supports the following options:

- exDropDownBackColor, specifies the drop down's background color
- exDropDownForeColor, specifies the drop down's foreground color
- exDropDownColumnCaption, specifies the HTML caption for each column within the drop down list, separated by Ś character (vertical broken bar, ALT + 221)
- exDropDownColumnWidth, specifies the width for each column within the drop down list, separated by Ś character (vertical broken bar, ALT + 221).
- exDropDownColumnPosition, specifies the position for each column within the drop down list, separated by Ś character (vertical broken

bar, ALT + 221).
- exDropDownColumnAutoResize, specifies whether the drop down list resizes automatically its visible columns to fit the drop down width

DropDownListType | 3

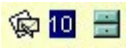
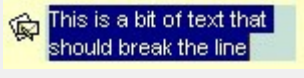


It provides an intuitive interface for your users to select values from predefined lists presented in a drop-down window. The DropDownListType editor has no standard edit field associated. Use the AddItem or InsertItem method to add predefined values to the drop down list. The DropDownRows property specifies the maximum number of visible rows into the drop-down list. The editor displays the caption of the item that matches the CellValue value. The item's icon is also displayed if it exists.

The following sample adds a column with a DropDownListType editor:

```
With .Columns.Add("Editor").Editor
    .DropDownAutoWidth = False
    .EditType = DropDownListType
    .AddItem 0, "Single Bed", 1
    .AddItem 1, "Double Bed", 2
    .AddItem 2, "Apartments", 3
    .InsertItem 3, "1 Bed Apartment", 4, 2
    .InsertItem 4, "2 Bed Apartment", 5, 2
    .AddItem 5, "Suite", 4
    .InsertItem 6, "Royal Suite", 1, 5
    .InsertItem 7, "Deluxe Suite", 2, 5
    .ExpandAll
End With
.Items.CellValue(.Items(0), "Editor") = 3
```

The editor supports the following options:

		• exDropDownImage, displays the predefined icon in the control's cell, if the user selects an item from a drop down editor. • exDropDownBackColor, specifies the drop down's background color • exDropDownForeColor, specifies the drop down's foreground color • exDropDownColumnCaption, specifies the HTML caption for each column within the drop down list, separated by Ś character (vertical broken bar, ALT + 221) • exDropDownColumnWidth, specifies the width for each column within the drop down list, separated by Ś character (vertical broken bar, ALT + 221). • exDropDownColumnPosition, specifies the position for each column within the drop down list, separated by Ś character (vertical broken bar, ALT + 221). • exDropDownColumnAutoResize, specifies whether the drop down list resizes automatically its visible columns to fit the drop down width
SpinType	4	 The SpinType allows your users to view and change numeric values using a familiar up/down button (spin control) combination. The AddItem or InsertItem method has no effect, if the EditType is SpinType. Use the exSpinStep option to specify the proposed change when user clicks the spin. Use the Numeric property to specify whether the edit control allows only numeric values only. Use the exSpinUpButtonUp, exSpinUpButtonDown, exSpinDownButtonUp and exSpinDownButtonDown to change the visual appearance for the spin control.
		 The MemoType is designed to provide an unique and intuitive interface, which you can implement within your application to assist users in working with textual

MemoType	5	information. If all information does not fit within the edit box, the window of the editor is enlarged. The AddItem or InsertItem method has no effect, if the EditType is MemoType. You can use options like exMemoHScrollBar, exMemoVScrollBar and so on.
CheckListType	6	 It provides an intuitive interface for your users to check values from predefined lists presented in a drop-down window. Each item has a check box associated. The editor displays the list of item captions, separated by comma, that is OR combination of CellValue value. Use the The AddItem or InsertItem method to add new predefined values to the drop down list. The DropDownRows property specifies the maximum number of visible rows into the drop-down list. Use the CheckImage property to change the check box appearance. The following sample adds a column with a CheckListType editor: `With .Columns.Add("Editor").Editor` `  .EditType = CheckListType` `  .AddItem 1, "Single Bed", 1` `  .AddItem 2, "Double Bed", 2` `  .AddItem 4, "Apartment", 3` `  .AddItem 8, "Suite", 4` `  .AddItem 16, "Royal Suite", 5` `End With` `.Items.CellValue(.Items(0), "Editor") = 5` The editor supports the following options: • exDropDownBackColor, specifies the drop down's background color • exDropDownForeColor, specifies the drop down's foreground color

DateType	7	 The DateType is a date/calendar control (not the Microsoft Calendar Control). The dropdown calendar provides an efficient and appealing way to edit dates at runtime. The DateType editor has a standard edit control associated. The user can easy select a date by selecting a date from the drop down calendar, or by typing directly the date. The editor displays the CellValue value as date. To change how the way how the control displays the date you can use FormatColumn event. The AddItem or InsertItem method has no effect, if the EditType is DateType. The following sample adds a column with a DateType editor: `With .Columns.Add("Editor").Editor` `  .EditType = DateType` `End With` `.Items.CellValue(.Items(0), "Editor") = Date`
MaskType	8	 You can use the MaskType to enter any data that includes literals and requires a mask to filter characters during data input. You can use this control to control the entry of many types of formatted information such as telephone numbers, social security numbers, IP addresses, license keys etc. The Mask property specifies the editor's mask. The MaskChar property specifies the masking character. The AddItem or InsertItem method has no effect, if the EditType is MaskType. The Mask property can use one or more literals: #,x,X,A,? <,>,*,\,{nMin,nMax},[...]. The following sample shows how to mask a column for input phone numbers: `With .Columns.Add("Editor").Editor`

```
    .EditType = MaskType
    .Mask = "(###) ### - ####"
End With
.Items.CellValue(.Items(0), "Editor") = "(214) 345 - 789"
```

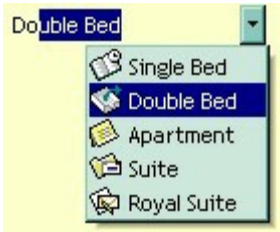
ColorType	9	 You can include a color selection control in your applications via the ColorType editor. Check the ColorListType also. The editor has a standard edit control and a color drop-down window. The color drop-down window contains two tabs that can be used to select colors, the "Pallette" tab shows a grid of colors, while the "System" tab shows the current windows color constants. The AddItem or InsertItem methodhas no effect, if the EditType is ColorType. You can use options like exColorShowPalette or exColorShowSystem. The following sample adds a column with a ColorType editor: `With .Columns.Add("Editor").Editor` `  .EditType = ColorType` `End With` `.Items.CellValue(.Items(0), "Editor") = vbRed`
FontType	10	 Provides an intuitive way for selecting fonts. The FontType editor contains a standard edit control and a font drop-down window. The font drop-down window contains a list with all system fonts. The AddItem or InsertItem method has no effect, if the EditType is FontType. The DropDownRows property specifies the maximum number of visible rows into the drop=down list.

The following sample adds a column with a FontType editor:

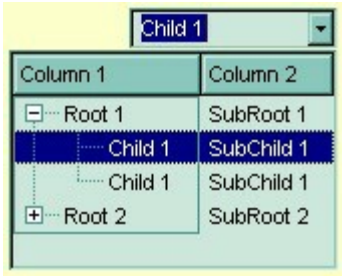
```
With .Columns.Add("Editor").Editor
   .EditType = FontType
End With
.Items.CellValue(.Items(0), "Editor") = "Times New Roman"
```

PictureType	11	 The PictureType provides an elegant way for displaying the fields of OLE Object type and cells that have a reference to an IPicture interface. An OLE Object field can contain a picture, a Microsoft Clip Gallery, a package, a chart, PowerPoint slide, a word document, a WordPad document, a wave file, an so on. In MS Access you can specify the field type to OLE Object. The DropDownMinWidth property specifies the minimum width for the drop=down window. The drop-down window is scaled based on the picture size. The AddItem or InsertItem method has no effect, if the EditType is PictureType. If your control is bounded to a ADO recordset, it automatically detects the OLE Object fields, so setting the editor's type to PictureType is not necessary. If your control is not bounded to an ADO recordset you can use the following sample to view OLE objects in the column "OLEObject" (the sample uses the NWIND database installed in your VB folder. Change the path if necessary, in the following sample: ' Creates an ADO Recordset Dim rs As Object Set rs = CreateObject("ADODB.Recordset") rs.Open "Employees",

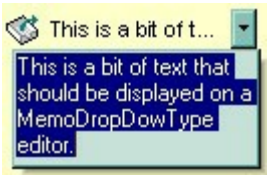
```
"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=
D:\Program Files\Microsoft Visual
Studio\VB98\NWIND.MDB", 3

' Adds a column of PictureType edit
Dim c As Column
Set c = .Columns.Add("OLEObject")
With c.Editor
.EditType = PictureType
End With
.Items.CellValue(.Items(0), "OLEObject") =
rs("Photo").Value
```

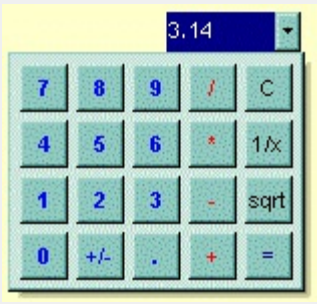
ButtonType	12	 The ButtonType editor consists into a standard edit field and a "..." button. The ButtonClick event is fired if the user has clicked the button. The AddItem or InsertItem method has no effect, if the EditType is ButtonType. Of course, you can apply for multiple buttons using the AddButton method, for any types.
ProgressBarType	13	 Uses the CellValue property to specify the percent being displayed in the ProgressBarTpe editor. The CellValue property should be between 0 and 100.
		 It provides an intuitive interface for your users to select values from pre-defined lists presented in a drop-down window. The PickEditType editor has a standard edit field associated, that useful for searching items while typing. The DropDownRows property specifies the maximum number of visible rows into the drop=down list. Use AddItem or InsertItem method to add new predefined values to the drop

PickEditType	14	down list. The editor displays the caption of the item that matches the CellValue value. The item's icon is also displayed if it exists. The following sample shows how to add values to a drop down list: ``` With .Columns.Add("Editor").Editor .EditType = PickEditType .AddItem 0, "Single Bed", 1 .AddItem 1, "Double Bed", 2 .AddItem 2, "Apartment", 3 .AddItem 3, "Suite", 4 .AddItem 4, "Royal Suite", 5 End With .Items.CellValue(.Items(0), "Editor") = "Apartment" ``` The editor supports the following options: • exDropDownBackColor, specifies the drop down's background color • exDropDownForeColor, specifies the drop down's foreground color • exDropDownColumnCaption, specifies the HTML caption for each column within the drop down list, separated by Ś character (vertical broken bar, ALT + 221) • exDropDownColumnWidth, specifies the width for each column within the drop down list, separated by Ś character (vertical broken bar, ALT + 221). • exDropDownColumnPosition, specifies the position for each column within the drop down list, separated by Ś character (vertical broken bar, ALT + 221). • exDropDownColumnAutoResize, specifies whether the drop down list resizes automatically its visible columns to fit the drop down width
		The LinkEditType control allows your application to edit and display hyperlink

LinkEditType	15	addresses.
UserEditorType	16	 The control is able to use ActiveX controls as a built-in editor. The control uses the UserEditor property to define the user control. If it succeeded the UserEditorObject property retrieves the newly created object. Events like: UserEditOpen, UserEditClose and UserEditorOleEvent are fired when the control uses custom editors. The setup installs the VB\UserEdit, VC\User.Edit samples that uses Exontrol's ExComboBox component as a new editor into the ExG2antt component (a multiple columns combobox control).
ColorListType	17	 You can include a color selection control in your application via the ColorListType editor, also. The editor hosts a predefined list of colors. By default. the following colors are added: Black, White, Dark Red, Dark Green, Dark Yellow, Dark Blue, Dark Magenta, Dark Cyan, Light Grey, Dark Grey, Red, Green, Yellow, Blue, Magenta, Cyan. The AddItem method adds a new color to your color list editor. You can use the exColorListShowName option to display the color's name. The following sample adds few custom colors to the ColorListType editor: `With .Columns.Add("Editor").Editor` `  .EditType = ColorListType` `  .AddItem 128, "Dark Red"`

```
    .AddItem RGB(0, 128, 0), "Dark Green"
    .AddItem RGB(0, 0, 128), "Dark Blue"
End With
.Items.CellValue(.Items(0), "Editor") = 128
```

MemoDropDownType	18	It provides a multiple lines edit control that's displayed into a drop down window.  • The Editor.Option(exMemoDropDownWidth) specifies the width (in pixels) of the MemoDropDownType editor when it is dropped. • The Editor.Option(exMemoDropDownHeight) specifies the height (in pixels) of the MemoDropDownType editor when it is dropped. • The Editor.Option(exMemoDropDownAcceptReturn) specifies whether the user closes the MemoDropDownType editor by pressing the ENTER key. If the Editor.Option(exMemoDropDownAcceptReturn) is True, the user inserts new lines by pressing the ENTER key. The user can close the editor by pressing the CTRL + ENTER key. If the Editor.Option(exMemoDropDownAcceptReturn) is False, the user inserts new lines by pressing the CTRL + ENTER key. The user can close the editor by pressing the ENTER key. • The Editor.Option(exMemoHScrollBar) adds the horizontal scroll bar to a MemoType or MemoDropDownType editor. • The Editor.Option(exMemoVScrollBar) adds the vertical scroll bar to a MemoType or MemoDropDownType editor • Use the Items.CellSingleLine property to specify whether the cell displays multiple lines The AddItem or InsertItem method has no effect, if the EditType is MemoDropDownType.

CheckValueType	19	Displays check boxes in the column or cell. The CellValue property indicates the state of the cell's check box. See also: CellHasCheckBox property. The CheckValueType editor supports the following options: • exCheckValue0. Specifies the check box state being displayed for unchecked state • exCheckValue1. Specifies the check box state being displayed for checked state • exCheckValue2. Specifies the check box state being displayed for partial-check state For instance, if your cells load boolean values (True is -1, False is 0), the control displays the partial-check icon for True values. You can call the following code before loading the CheckValueType editor: `G2antt1.DefaultEditorOption(exCheckValue2) = 1` in order to replace the partial-check appearance, to check state appearance.
SliderType	20	 Adds a slider control to a cell. Use the exSliderWidth, exSliderStep, exSliderMin, exSliderMax options to control the slider properties. Use the exSpinStep option to hide the spin control. Use the exSpinUpButtonUp, exSpinUpButtonDown, exSpinDownButtonUp and exSpinDownButtonDown to change the visual appearance for the spin control. Use the exSliderRange and exSliderThumb to change the visual appearance for the slider control.
CalculatorType	21	 Adds a drop down calculator to a node. Use the exCalcExecuteKeys, exCalcCannotDivideByZero, exCalcButtonWidth, exCalcButtonHeight, exCalcButtons, exCalcPictureUp, exCalcPictureDown to specify different options for calculator editor.

All editors support the following options:

- exLeftArrow, Disables focusing a new cell if the user presses the left arrow key while editing.
- exRightArrow, Disables focusing a new cell if the user presses the right arrow key while editing.
- exUpArrow, Disable focusing a new cell if the user presses the up arrow key while editing.
- exDownArrow, Disable focusing a new cell if the user presses the down arrow key while editing.
- exHomeKey, Disable focusing a new cell if the user presses the home key while editing.
- exEndKey, Disables focusing a new cell if the user presses the end key while editing.
- exPageUpKey, Disable focusing a new cell if the user presses the page up key while editing.
- exKeepSelBackColor. Keeps the selection background color while editor is visible.

constants exClipboardFormatEnum

Defines the clipboard format constants. Use GetFormat property to check whether the clipboard data is of given type

Name	Value	Description
exCFText	1	Null-terminated, plain ANSI text in a global memory bloc.
exCFBitmap	2	A bitmap compatible with Windows 2.x.
exCFMetafile	3	A Windows metafile with some additional information about how the metafile should be displayed.
exCFDIB	8	A global memory block containing a Windows device-independent bitmap (DIB).
exCFPalette	9	A color-palette handle.
exCFEMetafile	14	A Windows enhanced metafile.
exCFFiles	15	A collection of files. Use Files property to get or set the collection of files.
exCFRTF	-16639	A RTF document.

constants exOLEDragOverEnum

State transition constants for the OLEDragOver event

Name	Value	Description
exOLEDragEnter	0	Source component is being dragged within the range of a target.
exOLEDragLeave	1	Source component is being dragged out of the range of a target.
exOLEDragOver	2	Source component has moved from one position in the target to another.

constants exOLEDropEffectEnum

Drop effect constants for OLE drag and drop events.

Name	Value	Description
exOLEDropEffectNone	0	Drop target cannot accept the data, or the drop operation was cancelled.
exOLEDropEffectCopy	1	Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
exOLEDropEffectMove	2	Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.
exOLEDropEffectScroll	-2147483648	This value is not implemented.

constants exOLEDropModeEnum

Constants for the OLEDropMode property, that defines how the control accepts OLE drag and drop operations. Use the OLEDropMode property to set how the component handles drop operations.

Name	Value	Description
exOLEDropNone	0	The control is not used OLE drag and drop functionality.
exOLEDropManual	1	The control triggers the OLE drop events, allowing the programmer to handle the OLE drop operation in code.

Here's the list of events related to OLE drag and drop: OLECompleteDrag, OLEDragDrop, OLEDragOver, OLEGiveFeedback, OLESetData, OLEStartDrag.

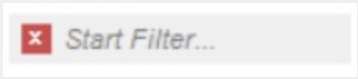
constants ExpandButtonEnum

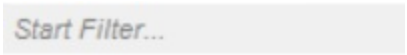
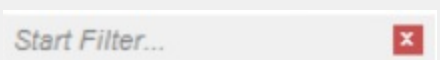
Defines how the control displays the expanding/collapsing buttons.

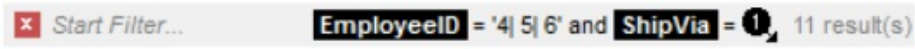
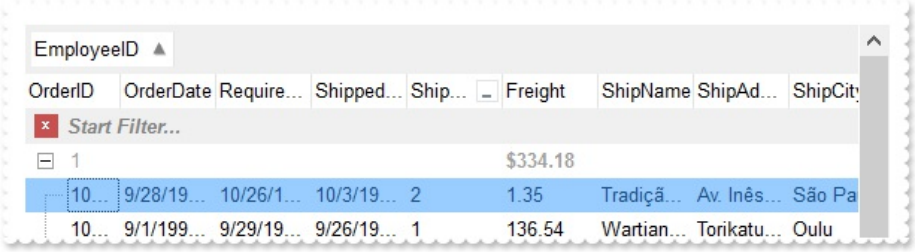
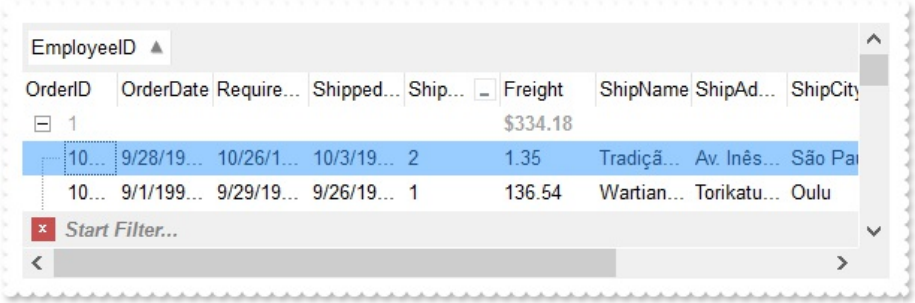
Name	Value	Description
exNoButtons	0	The control displays no expand buttons.
exPlus	-1	A plus sign is displayed for collapsed items, and a minus sign for expanded items.(⊞ ⊟)
exArrow	1	The control uses icons to display the expand buttons.(▸ ▾)
exCircle	2	The control uses icons to display the expand buttons. (⊕ ⊖)
exWPlus	3	The control uses icons to display the expand buttons. (✚ ▬)
exCustom	4	The HasButtonsCustom property specifies the index of icons being used for +/- signs on parent items.

constants FilterBarVisibleEnum

The FilterBarVisibleEnum type defines the flags you can use on FilterBarPromptVisible property. The FilterBarCaption property defines the caption to be displayed on the control's filter bar. The FilterBarPromptVisible property , specifies how the control's filter bar is displayed and behave. The FilterBarVisibleEnum type includes several flags that can be combined together, as described bellow:

Name	Value	Description
exFilterBarHidden	0	No filter bar is shown while there is no filter applied. The control's filter bar is automatically displayed as soon a a filter is applied.
exFilterBarPromptVisible	1	The exFilterBarPromptVisible flag specifies that the control's filter bar displays the filter prompt. The exFilterBarPromptVisible, exFilterBarVisible, exFilterBarCaptionVisible flag , forces the control's filter-prompt, filter bar or filter bar description (even empty) to be shown. If missing, no filter prompt is displayed. The FilterBarPrompt property to specify the HTML caption being displayed in the filter bar when the filter pattern is missing. 
exFilterBarVisible	2	The exFilterBarVisible flag forces the control's filter bar to be shown, no matter if any filter is applied. If missing, no filter bar is displayed while the control has no filter applied.  or combined with exFilterBarPromptVisible 
exFilterBarCaptionVisible	4	The exFilterBarVisible flag forces the control's filter bar to display the FilterBarCaption property. 

exFilterBarSingleLine	16	The exFilterBarVisible flag specifies that the caption on the control's filter bar id displayed on a single line. The exFilterBarSingleLine flag , specifies that the filter bar's caption is shown on a single line, so HTML tag or \r\n are not handled. By default, the control's filter description applies word wrapping. Can be combined to exFilterBarCompact to display a single-line filter bar. If missing, the caption on the control's filter bar is displayed on multiple lines. You can change the height of the control's filter bar using the FilterBarHeight property.
exFilterBarToggle	256	The exFilterBarToggle flag specifies that the user can close the control's filter bar (removes the control's filter) by clicking the close button of the filter bar or by pressing the CTRL + F, while the control's filter bar is visible. If no filter bar is displayed, the user can display the control's filter bar by pressing the CTRL + F key. While the control's filter bar is visible the user can navigate though the list or control's filter bar using the ALT + Up/Down keys. If missing, the control's filter bar is always shown if any of the following flags is present exFilterBarPromptVisible, exFilterBarVisible, exFilterBarCaptionVisible.
exFilterBarShowCloseIfRequired	512	The exFilterBarShowCloseIfRequired flag indicates that the close button of the control's filter bar is displayed only if the control has any currently filter applied. The Background(exFooterFilterBarButton) property on -1 hides permanently the close button of the control's filter bar. 
exFilterBarShowCloseOnRight	1024	The exFilterBarShowCloseOnRight flag specifies that the close button of the control's filter bar should be displayed on the right side. If the control's RightToLeft property is True, the close button of the control's filter bar would be automatically displayed on the left side. 

exFilterBarCompact	2048	The exFilterBarCompact flag compacts the control's filter bar, so the filter-prompt will be displayed to the left, while the control's filter bar caption will be displayed to the right. This flag has effect only if combined with the exFilterBarPromptVisible. This flag can be combined with the exFilterBarSingleLine flag, so all filter bar will be displayed compact and on a single line. 
exFilterBarShort	4096	exFilterBarShort
		The exFilterBarTop flag displays the filter-bar on top (between control's header and items section as shown: 
exFilterBarTop	8192	By default, the filter-bar is shown aligned to the bottom (between items and horizontal-scroll bar) as shown: 

constants FilterIncludeEnum

The FilterIncludeEnum type defines the items to include when control's filter is applied. The FilterInclude property specifies the items being included, when the list is filtered. The FilterIncludeEnum type supports the following values:

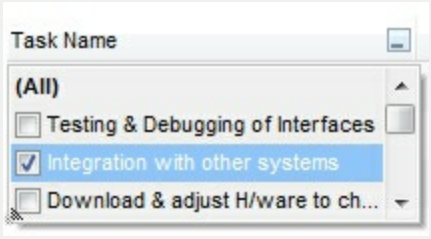
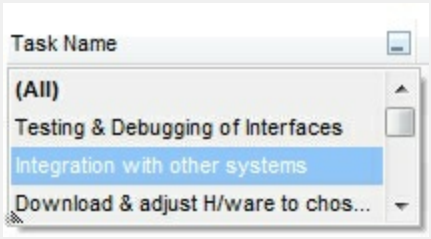
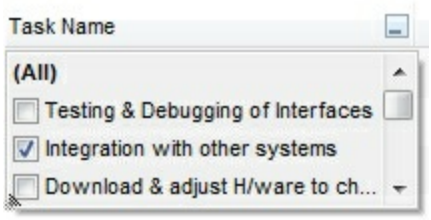
Name	Value	Description
exItemsWithoutChilds	0	Items (and parent-items) that match the filter are shown (no child-items are included)
exItemsWithChilds	1	Items (parent and child-items) that match the filter are shown
exRootsWithoutChilds	2	Only root-items (excludes child-items) that match the filter are displayed
exRootsWithChilds	3	Root-items (and child-items) that match the filter are displayed
exMatchingItemsOnly	4	Shows only the items that matches the filter (no parent or child-items are included)
exMatchIncludeParent	240	Specifies that the item matches the filter if any of its parent-item matches the filter. The exMatchIncludeParent flag can be combined with any other value.

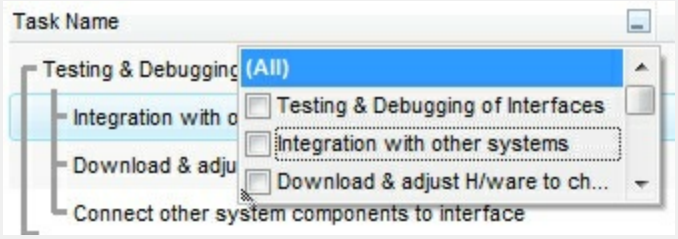
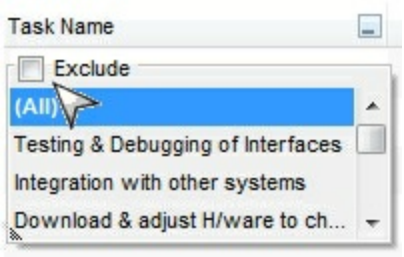
constants FilterListEnum

The FilterListEnum type specifies the type of items being included in the column's drop down list filter. The FilterList property specifies the items being included to the column's drop down filter-list, including other options for filtering. Use the DisplayFilterPattern and/or DisplayFilterDate property to display the pattern field, a date pattern or a calendar control inside the drop down filter window.

The FilterList can be a bit-combination of exAllItems, exVisibleItems or exNoItems with any other flags being described bellow:

Name	Value	Description
exAllItems	0	The filter's list includes all items in the column.
exVisibleItems	1	The filter's list includes only visible (filtered) items from the column. The visible items include child items of collapsed items.
exNoItems	2	The filter's list does not include any item from the column. Use this option if the drop down filter displays a calendar control for instance.
exLeafItems	3	The filter's list includes the leaf items only. A leaf item is an item with no child items.
exRootItems	4	The filter's list includes the root items only.
exSortItemsDesc	16	If the exSortItemsDesc flag is set the values in the drop down filter's list gets listed descending. If none of the exSortItemsAsc or exSortItemsDesc is present, the list is built as the items are displayed in the control.
exSortItemsAsc	32	If the exSortItemsAsc flag is set the values in the drop down filter's list gets listed ascending. If none of the exSortItemsAsc or exSortItemsDesc is present, the list is built as the items are displayed in the control.
exIncludeInnerCells	64	The exIncludeInnerCells flag specifies whether the inner cells values are included in the drop down filter's list. The SplitCell method adds an inner cell, on in other words splits a cell.
exSingleSel	128	If this flag is present, the filter's list supports single selection. By default, (If missing), the user can select multiple items using the CTRL key. Use the exSingleSel property to prevent multiple items

		selection in the drop down filter list.
exShowCheckBox	256	The filter's list displays a check box for each included item. Clicking the checkbox, makes the item to be include din the filter. If this flag is present, the filter is closed once the user presses ENTER or clicks outside of the drop down filter window. By default, (this flag is missing), clicking an item closes the drop down filter, if the CTRL key is not pressed. This flag can be combined with exHideCheckSelect. The following screen shot shows the drop down filter **with** or **with no** exShowCheckBox flag:  or 
exHideCheckSelect	512	The selection background is not shown for checked items in the filter's list. This flag can be combined with exShowCheckBox. The following screen shot shows no selection background for the checked items: 
		This flag allows highlighting the focus cell value in the filter's list. The focus cell value is the cell's content at the moment the drop down filter window is shown. For instance, click an item so a new item is selected, and click the drop down filter button. A

exShowFocusItem	1024	item being focused in the drop down filter list is the one you have in the control's selection. This flag has effect also, if displaying a calendar control in the drop down filter list. The following screen shot shows the focused item in the filter's list (The Integration ... item in the background is the focused item, and the same is in the filter's list) : 
exShowPrevSelectOpaque	2048	By default, the previously selection in the drop down filter's list is shown using a semi-transparent color. Use this flag to show the previously selection using an opaque color. The exSelFilterForeColor and exSelFilterBackColor options defines the filter's list selection foreground and background colors.
exEnableToolTip	4096	This flag indicates whether the filter's tooltip is shown. The Description(exFilterBarTooltip,exFilterBarPatternTooltip ...) properties defines the filter's tooltips.
exShowExclude	8192	This flag indicates whether the Exclude option is shown in the drop down filter window. This option has effect also if the drop down filter window shows a calendar control. The exFilterExclude flag excludes programmatically the selected items in the drop down filter panel. The following screen shot shows the Exclude field in the drop down filter window: 

exShowBlanks	16384	This flag indicates whether the (Blanks) and (NonBlanks) items are shown in the filter's list

constants FilterPromptEnum

The FilterPromptEnum type specifies the type of prompt filtering. Use the FilterBarPromptType property to specify the type of filtering when using the prompt. The FilterBarPromptColumns specifies the list of columns to be used when filtering. The FilterBarPromptPattern property specifies the pattern for filtering. The pattern may contain one or more words being delimited by space characters.

The filter prompt feature supports the following values:

Name	Value	Description
exFilterPromptContainsAll	1	The list includes the items that contains all specified sequences in the filter. Can be combined with exFilterPromptCaseSensitive, exFilterPromptStartWords, exFilterPromptEndWords or exFilterPromptWords
exFilterPromptContainsAny	2	The list includes the items that contains any of specified sequences in the filter. Can be combined with exFilterPromptCaseSensitive, exFilterPromptStartWords, exFilterPromptEndWords or exFilterPromptWords
exFilterPromptStartWith	3	The list includes the items that starts with any specified sequences in the filter. Can be combined with exFilterPromptCaseSensitive, exFilterPromptStartWords, exFilterPromptEndWords or exFilterPromptWords
exFilterPromptEndWith	4	The list includes the items that ends with any specified sequences in the filter. Can be combined with exFilterPromptCaseSensitive, exFilterPromptStartWords, exFilterPromptEndWords or exFilterPromptWords
exFilterPromptPattern	16	The filter indicates a pattern that may include wild characters to be used to filter the items in the list. Can be combined with exFilterPromptCaseSensitive. The FilterBarPromptPattern property may include wild characters as follows: • '?' for any single character • '*' for zero or more occurrences of any character • '#' for any digit character

- ' ' space delimits the patterns inside the filter

exFilterPromptCaseSensitive	256	Filtering the list is case sensitive. Can be combined with exFilterPromptContainsAll, exFilterPromptContainsAny, exFilterPromptStartWith, exFilterPromptEndWith or exFilterPromptPattern
exFilterPromptStartWords	4608	The list includes the items that starts with specified words, in any position. Can be combined with exFilterPromptContainsAll, exFilterPromptContainsAny, exFilterPromptStartWith or exFilterPromptEndWith.
exFilterPromptEndWords	8704	The list includes the items that ends with specified words, in any position. Can be combined with exFilterPromptContainsAll, exFilterPromptContainsAny, exFilterPromptStartWith or exFilterPromptEndWith.
exFilterPromptWords	12800	The filter indicates a list of words. Can be combined with exFilterPromptContainsAll, exFilterPromptContainsAny, exFilterPromptStartWith or exFilterPromptEndWith.

constants FilterTypeEnum

Defines the type of filter applies to a column. Use the FilterType property of the Column object to specify the type of filter being used. Use the Filter property of Column object to specify the filter being used. The value for Filter property depends on the FilterType property.

Name	Value	Description
exAll	0	No filter applied
exBlanks	1	Only blank items are included
exNonBlanks	2	Only non blanks items are included
exPattern	3	Only items that match the pattern are included. The Filter property defines the pattern. A pattern may contain the wild card characters '?' for any single character, '*' for zero or more occurrences of any character, '#' for any digit character, and [chars] indicates a group of characters. If any of the *, ?, # or | characters are preceded by a \ (escape character) it masks the character itself.
exDate	4	Use the exDate type to filter items into a given interval. The Filter property of the Column object defines the interval of dates being used to filter items. The interval of dates should be as [dateFrom] to [dateTo]. Use the Description property to changes the "to" conjunction used to split the dates in the interval. If the dateFrom value is missing, the control includes only the items before the dateTo date, if the dateTo value is missing, the control includes the items after the dateFrom date. If both dates (dateFrom and dateTo) are present, the control includes the items between this interval of dates. For instance, the "2/13/2004 to" includes all items after 2/13/2004 inclusive, or "2/13/2004 to Feb 14 2005" includes all items between 2/13/2004 and 2/14/2004.
exNumeric	5	If the FilterType property is exNumeric, the Filter property may include operators like <, <=, =, <>, >= or > and numbers to define rules to include numbers in the control's list. For instance, the "> 10 < 100" filter indicates all numbers greater than 10 and less than 100. If the FilterType property is

		exNumeric, the drop down filter window doesn't display the filter list that includes items "(All)", "(Blanks)", ... and so on.
exCheck	6	Only checked or unchecked items are included. The CellState property indicates the state of the cell's checkbox. The control filters for checked items, if the Filter property is "1". The control filters for unchecked items, if the Filter property is "0". A checked item has the the CellState property different than zero. An unchecked item has the CellState property on zero.
exImage	10	Filters items by icons. The CellImage property indicates the cell's icon
exFilter	240	Only the items that are in the Filter property are included.
exFilterDoCaseSensitive	256	If this flag is present, the filtering on the column is case-sensitive. If this flag is missing, the filtering is case-insensitive (by default). You can use the exFilterDoCaseSensitive flag to perform case-sensitive filtering within the column. This flag is not applied to filter prompt feature.
exFilterExclude	512	The flag indicates that the Exclude field of the column is checked. The flag indicates that the items that match the filter are excluded from the list.

constants FormatApplyToEnum

The FormatApplyToEnum expression indicates whether a format is applied to an item or to a column. Any value that's greater than 0 indicates that the conditional format is applied to the column with the value as index. A value less than zero indicates that the conditional format object is applied to items. Use the ApplyTo property to specify whether the conditional format is applied to items or to columns.

Name	Value	Description
exFormatToItems	-1	Specifies whether the condition is applied to items.
exFormatToColumns	0	Specifies whether the condition is applied to columns. The 0 value indicates that the conditional format is applied to the first column. The 1 value indicates the conditional format is applied to the second column. The 2 value indicates the conditional format is applied to the third column, and so on.

constants GridLinesEnum

Defines how the control paints the grid lines.

Name	Value	Description
exNoLines	0	The control displays no grid lines.
exAllLines	-1	The control displays vertical and horizontal grid lines.
exRowLines	-2	The control paints grid lines only for current rows.
exHLines	1	Only horizontal grid lines are shown.
exVLines	2	Only vertical grid lines are shown.

constants GridLinesStyleEnum

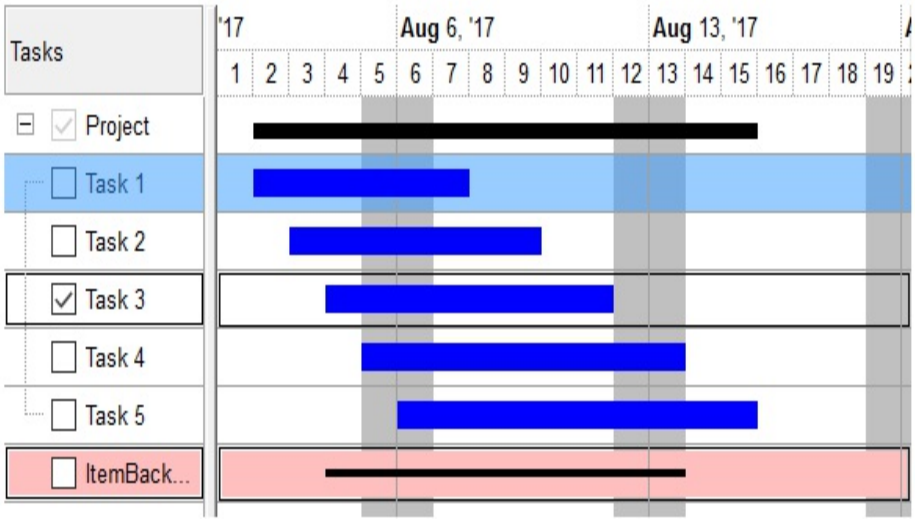
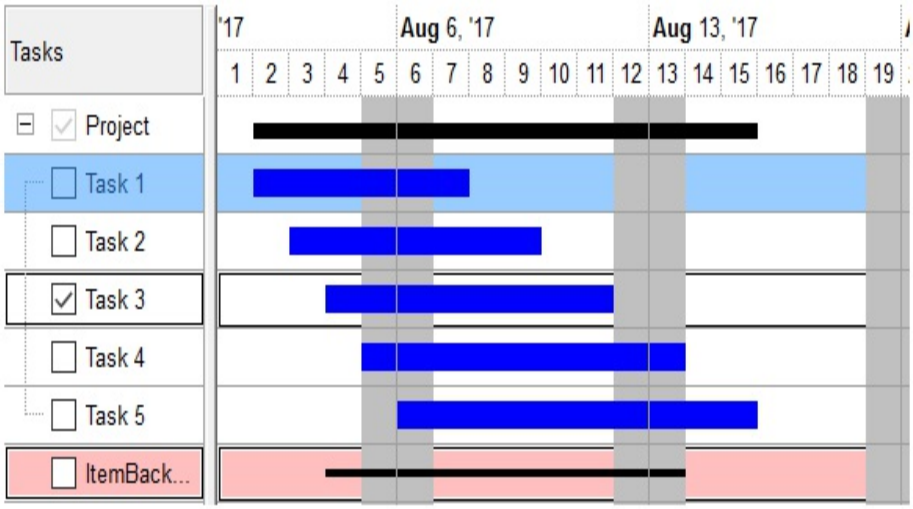
The GridLinesStyle type specifies the style to show the control's grid lines. The GridLineStyle property indicates the style of the gridlines being displayed in the view if the DrawGridLines property is not zero. The GridLinesStyle enumeration specifies the style for horizontal or/and vertical gridlines in the control. The DrawGridLines property of the Chart object specifies whether the grid lines are shown in the chart part of the control.

Name	Value	Description
exGridLinesDot	0	The control's gridlines are shown as dotted.
exGridLinesHDot4	1	The horizontal control's gridlines are shown as dotted.
exGridLinesVDot4	2	The vertical control's gridlines are shown as dotted.
exGridLinesDot4	3	The control's gridlines are shown as solid.
exGridLinesHDash	4	The horizontal control's gridlines are shown as dashed.
exGridLinesVDash	8	The vertical control's gridlines are shown as dashed.
exGridLinesDash	12	The control's gridlines are shown as dashed.
exGridLinesHSolid	16	The horizontal control's gridlines are shown as solid.
exGridLinesVSolid	32	The vertical control's gridlines are shown as solid.
exGridLinesSolid	48	The control's gridlines are shown as solid.

The exGridLinesBehind flag specifies whether the:

- chart's vertical gridlines are shown behind bars. For instance, Chart.GridLineStyle = GridLinesStyleEnum.exGridLinesHSolid Or GridLinesStyleEnum.exGridLinesBehind shows horizontal gridlines as solid, and the vertical gridlines shows behind the bars
- non-working part of the item is shown behind the item's background

The following screen shot shows the non-working part (gray section) behind the item's background (Chart.GridLineStyle property includes the exGridLinesBehind flag)

exGridLinesBehind	256	 The following screen shot shows the non-working part (gray section) in front of the item's background (Chart.GridLineStyle property without the exGridLinesBehind flag)  The exGridLinesBehind flag has effect for the chart area only, so it has to be used with the Chart.GridLineStyle property.
exGridLinesGeometric	512	The control's gridlines are drawn using a geometric pen. The exGridLinesGeometric flag can be combined with any other flag. A geometric pen can have any width and can have any of the attributes of a brush, such as dithers and patterns. A cosmetic pen can only be a single pixel wide and must be a solid color, but cosmetic pens are generally faster than geometric pens. The width of a geometric pen is always specified in world units. The width of a cosmetic pen is always 1.

constants GroupBarsOptionsEnum

The GroupBarsOptionEnum type specifies the way two bars gets grouped together. It may specify to prevent changing the lengths of the bars, or limit the interval between them when grouping. The GroupBars method groups two bars, so they are moving or resizing together, when any bar in the group changes.

The length of the bar and interval between two bars are defined as follows:

- The **length** of the bar is the same as its duration, in other words it is the difference between ending date of the bar and starting date of the bar. If the ItemBar(exBarKeepWorkingCount) property of the bar is True, the **length** indicates the bar's working count (ItemBar(exBarWorkingCount) property)
- The **interval** between bars is the same as the distance between the starting and ending points of the grouping bars. For instance, if you have linked the end of the bar A with the start of the bar B, the interval is defined as difference between the starting date of the bar B and ending date of the bar A. In other sample, you may have linked the start of the bar A with start of the bar B, in this case the interval is defined as being the difference between the start of the bar A and starting date of the bar B, nothing else.

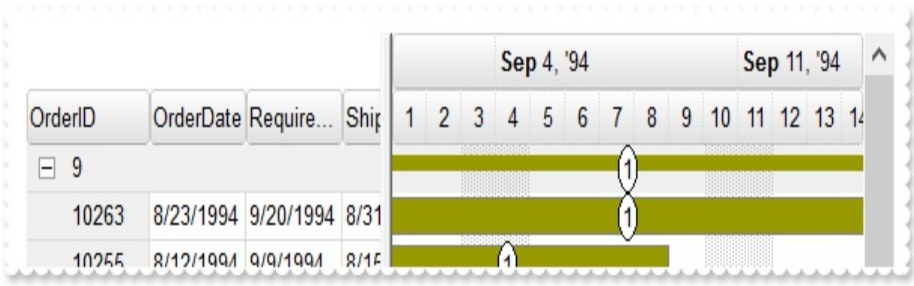
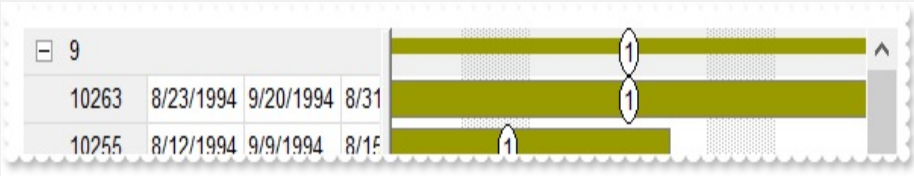
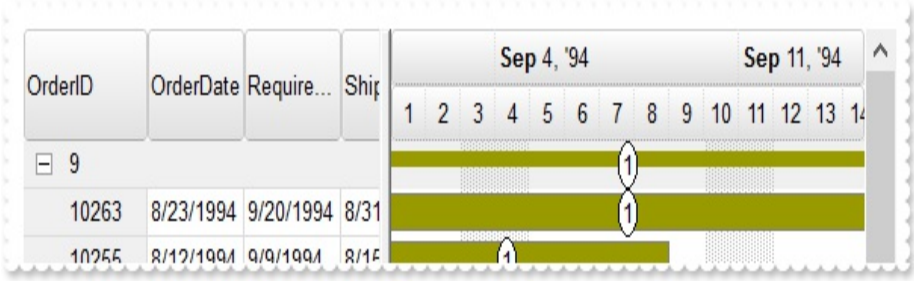
The GroupBarsOptionEnum value mat be a combination of any of the following values:

Name	Value	Description
exGroupBarsNone	-1	Performs ungrouping the bars or specifies that the bars are not grouped. The exGroupBarsNone can't be used with any other options. Set the Link(exLinkGroupBars) property on exGroupBarsNone and if the linked bars were grouped they are ungrouped.
exGroupBarsOptionNone	0	Specifies no options when grouping bars. Default option.
exPreserveBarLengthA	1	Preserves the length of the bar A when grouping. The length of the bar A is not changed if another bar in the same group may change it by grouping. Use the exPreserveBarLength value to specify that both bars should preserve their lengths.
exPreserveBarLengthB	2	Preserves the length of the bar B when grouping. The length of the bar B is not changed if another bar in the same group may change it by grouping. Use the exPreserveBarLength value to specify that both bars should preserve their lengths.

exPreserveBarLength	3	Preserves the length of both bars when grouping. The length of both bars is not changed when another bar in the same group may change the length of bars A and B.
exIgnoreOriginalInterval	4	Ignores the original interval between bars when grouping. At the moment GroupBars method is called, the control keeps the original interval between bars, so this option will specify whether to handle or not. For instance, you can have the exIgnoreOriginalInterval, and you can specify a different interval between bars using the first parameter (Fixed Interval) in the Options parameter of the GroupBars method.
exLimitIntervalMin	8	Limits the interval between bars so it can't be less than a specified value.
exLimitIntervalMax	16	Limits the interval between bars so it can't be greater than a specified value.
exLimitInterval	24	Limits the interval between bars so it fits a specified range.
exFlexibleInterval	32	The interval between bars is not limited and the bar B can be moved anywhere to the right of the bar A.
exLimitIntervalTreatAsWorking	64	The interval between bars is specified in working units. The NonworkingDays property specifies the days to be non-working. The exLimitIntervalTreatAsWorking can be combined with exLimitIntervalMin, exLimitIntervalMax or exLimitInterval. If the exLimitIntervalTreatAsWorking is set, interval value indicates working days, else it indicates days.

constants HeaderVisibleEnum

YThe HeaderVisibleEnum type specifies whether the control's header bar is visible or hidden. The HeaderVisible property retrieves or sets a value that indicates whether the control's header is visible or hidden. Use the HeaderHeight property to specify the height of the control's header bar. The HeaderVisibleEnum type supports the following values:

Name	Value	Description
exHeaderVisible	-1	The control's header is visible. 
exHeaderHidden	0	The control's header is hidden. 
exHeaderVisibleExtendLevels	1	The control's header is visible, and each column's height is extended to cover all levels of the control's chart (LevelCount property). 

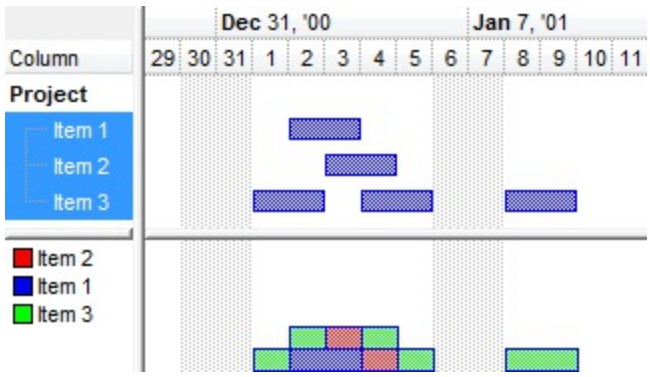
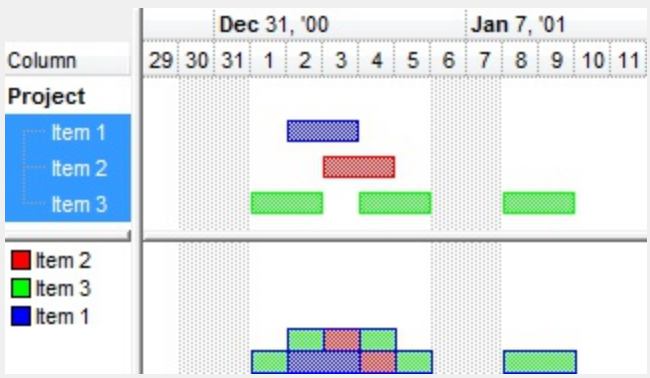
constants HierarchyLineEnum

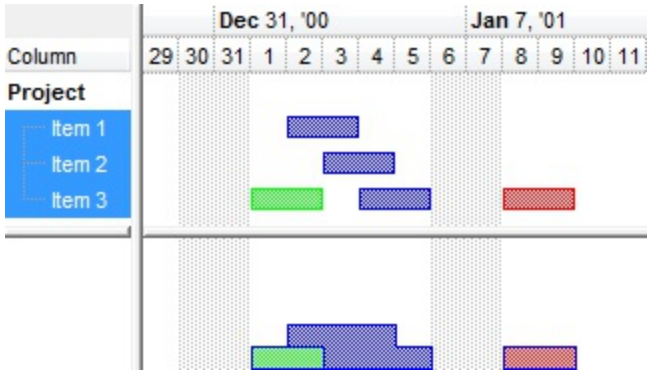
Defines how the control paints the hierarchy lines. Use the TreeColumnIndex property to define the index of the column that displays the hierarchy. Use the LinesAtRoot property to connect root items. Use the HasLines property to connect a child items to their correspondent parent item.

Name	Value	Description
exNoLine	0	The control displays no lines when painting the hierarchy.
exDotLine	-1	The control uses a dotted line to paint the hierarchy.
exSolidLine	1	The control uses a solid line to paint the hierarchy.
exThinLine	2	The control uses a thin line to paint the hierarchy.

constants HistogramCumulativeOriginalColorBarsEnum

The HistogramCumulativeOriginalColorBarsEnum type indicates whether the color for the bars being represented in the histogram is changed. The Bar.HistogramCumulativeOriginalColorBars property indicates whether the bar's color is changed while representing them in the histogram.

Name	Value	Description
exShowCumulativeColor	-1	The color for bar is not changed, but its reflection in the histogram shows the corresponding cumulative color.  The HistogramCumulativeShowLegend property specifies the index of the column to show the legend for the items being displayed in the cumulative histogram. The HistogramCumulativeColors property defines the number of colors being displayed in the cumulative histogram. The HistogramCumulativeColor property specifies a cumulative color based on its index.
exChangeColor	0	The color for bar and its reflection in the histogram is showing the corresponding cumulative color.  The HistogramCumulativeColors property defines the number of colors being displayed in the

		cumulative histogram. The HistogramCumulativeColor property specifies a cumulative color based on its index. The HistogramCumulativeShowLegend property specifies the index of the column to show the legend for the items being displayed in the cumulative histogram.
exKeepOriginalColor	1	The color for bar and its reflection in the histogram is not changed.  Use the ItemBar(exBarColor) property to specify a different color for a specified bar. The ItemBar(exBarHistLegend) property specifies the description to show within the histogram's legend for the bar in the control's histogram (exKeepOriginalColor only). For exKeepOriginalColor, the HistogramCumulativeShowLegend property specifies whether the bar's legend (ItemBar(exBarHistLegend) property) is shown or hidden.

VBA Is it possible to define the bar colors, and have the cumulative histogram showing the same colors?

```
With G2antt1
   .BeginUpdate
   .SingleSel = False
   With .Chart
      .LevelCount = 2
      .AllowLinkBars = False
      .DrawGridLines = -1
      .FirstVisibleDate = #12/29/2000#
      .HistogramVisible = True
```

```
      .HistogramHeight = 72
      .PaneWidth(0) = 64
      .HistogramView = 1298
      With .Bars.Item("Task")
         .HistogramType = 256
         .HistogramItems = 6
         .HistogramPattern = .Pattern
         .HistogramCumulativeOriginalColorBars = 1
      End With
   End With
   .Columns.Add "Column"
   With .Items
      h = .AddItem("Project")
      .ItemBold(h) = True
      .SelectableItem(h) = False
      h1 = .InsertItem(h,0,"Item 1")
      .AddBar h1,"Task",#1/2/2001#,#1/4/2001#
      h1 = .InsertItem(h,0,"Item 2")
      .AddBar h1,"Task",#1/3/2001#,#1/5/2001#
      h1 = .InsertItem(h,0,"Item 3")
      .AddBar h1,"Task",#1/4/2001#,#1/6/2001#
      .AddBar h1,"Task",#1/1/2001#,#1/3/2001#,"green"
      .ItemBar(h1,"green",33) = 65280
      .AddBar h1,"Task",#1/8/2001#,#1/10/2001#,"red"
      .ItemBar(h1,"red",33) = 255
      .ExpandItem(h) = True
      .SelectAll
   End With
   .EndUpdate
End With
```

VB6 Is it possible to define the bar colors, and have the cumulative histogram showing the same colors?

```
With G2antt1
   .BeginUpdate
   .SingleSel = False
```

```
    With .Chart
        .LevelCount = 2
        .AllowLinkBars = False
        .DrawGridLines = exAllLines
        .FirstVisibleDate = #12/29/2000#
        .HistogramVisible = True
        .HistogramHeight = 72
        .PaneWidth(0) = 64
        .HistogramView = HistogramViewEnum.exHistogramSelectedItems Or
HistogramViewEnum.exHistogramUnlockedItems Or
HistogramViewEnum.exHistogramLeafItems Or
HistogramViewEnum.exHistogramNoGrouping
        With .Bars.Item("Task")
            .HistogramType = exHistCumulative
            .HistogramItems = 6
            .HistogramPattern = .Pattern
            .HistogramCumulativeOriginalColorBars = exKeepOriginalColor
        End With
    End With
    .Columns.Add "Column"
    With .Items
        h = .AddItem("Project")
        .ItemBold(h) = True
        .SelectableItem(h) = False
        h1 = .InsertItem(h,0,"Item 1")
        .AddBar h1,"Task",#1/2/2001#,#1/4/2001#
        h1 = .InsertItem(h,0,"Item 2")
        .AddBar h1,"Task",#1/3/2001#,#1/5/2001#
        h1 = .InsertItem(h,0,"Item 3")
        .AddBar h1,"Task",#1/4/2001#,#1/6/2001#
        .AddBar h1,"Task",#1/1/2001#,#1/3/2001#,"green"
        .ItemBar(h1,"green",exBarColor) = 65280
        .AddBar h1,"Task",#1/8/2001#,#1/10/2001#,"red"
        .ItemBar(h1,"red",exBarColor) = 255
        .ExpandItem(h) = True
        .SelectAll
    End With
```

```
   .EndUpdate
End With
```

VB.NET Is it possible to define the bar colors, and have the cumulative histogram showing the same colors?

```
Dim h,h1
With Exg2antt1
   .BeginUpdate()
   .SingleSel = False
   With .Chart
      .LevelCount = 2
      .AllowLinkBars = False
      .DrawGridLines = exontrol.EXG2ANTTLib.GridLinesEnum.exAllLines
      .FirstVisibleDate = #12/29/2000#
      .HistogramVisible = True
      .HistogramHeight = 72
      .set_PaneWidth(False,64)
      .HistogramView =
exontrol.EXG2ANTTLib.HistogramViewEnum.exHistogramSelectedItems Or
exontrol.EXG2ANTTLib.HistogramViewEnum.exHistogramUnlockedItems Or
exontrol.EXG2ANTTLib.HistogramViewEnum.exHistogramLeafItems Or
exontrol.EXG2ANTTLib.HistogramViewEnum.exHistogramNoGrouping
      With .Bars.Item("Task")
         .HistogramType = exontrol.EXG2ANTTLib.HistogramTypeEnum.exHistCumulative
         .HistogramItems = 6
         .HistogramPattern = .Pattern
         .HistogramCumulativeOriginalColorBars =
exontrol.EXG2ANTTLib.HistogramCumulativeOriginalColorBarsEnum.exKeepOriginalColor
      End With
   End With
   .Columns.Add("Column")
   With .Items
      h = .AddItem("Project")
      .set_ItemBold(h,True)
      .set_SelectableItem(h,False)
      h1 = .InsertItem(h,0,"Item 1")
```

```
      .AddBar(h1,"Task",#1/2/2001#,#1/4/2001#)
      h1 = .InsertItem(h,0,"Item 2")
      .AddBar(h1,"Task",#1/3/2001#,#1/5/2001#)
      h1 = .InsertItem(h,0,"Item 3")
      .AddBar(h1,"Task",#1/4/2001#,#1/6/2001#)
      .AddBar(h1,"Task",#1/1/2001#,#1/3/2001#,"green")

.set_ItemBar(h1,"green",exontrol.EXG2ANTTLib.ItemBarPropertyEnum.exBarColor,65280)
      .AddBar(h1,"Task",#1/8/2001#,#1/10/2001#,"red")
      .set_ItemBar(h1,"red",exontrol.EXG2ANTTLib.ItemBarPropertyEnum.exBarColor,255)
      .set_ExpandItem(h,True)
      .SelectAll()
   End With
   .EndUpdate()
End With
```

VB.NET for /COM Is it possible to define the bar colors, and have the cumulative histogram showing the same colors?

```
Dim h,h1
With AxG2antt1
   .BeginUpdate()
   .SingleSel = False
   With .Chart
      .LevelCount = 2
      .AllowLinkBars = False
      .DrawGridLines = EXG2ANTTLib.GridLinesEnum.exAllLines
      .FirstVisibleDate = #12/29/2000#
      .HistogramVisible = True
      .HistogramHeight = 72
      .PaneWidth(False) = 64
      .HistogramView = EXG2ANTTLib.HistogramViewEnum.exHistogramSelectedItems Or
EXG2ANTTLib.HistogramViewEnum.exHistogramUnlockedItems Or
EXG2ANTTLib.HistogramViewEnum.exHistogramLeafItems Or
EXG2ANTTLib.HistogramViewEnum.exHistogramNoGrouping
      With .Bars.Item("Task")
         .HistogramType = EXG2ANTTLib.HistogramTypeEnum.exHistCumulative
```

```
        .HistogramItems = 6
        .HistogramPattern = .Pattern
        .HistogramCumulativeOriginalColorBars =
EXG2ANTTLib.HistogramCumulativeOriginalColorBarsEnum.exKeepOriginalColor
      End With
   End With
   .Columns.Add("Column")
   With .Items
      h = .AddItem("Project")
      .ItemBold(h) = True
      .SelectableItem(h) = False
      h1 = .InsertItem(h,0,"Item 1")
      .AddBar(h1,"Task",#1/2/2001#,#1/4/2001#)
      h1 = .InsertItem(h,0,"Item 2")
      .AddBar(h1,"Task",#1/3/2001#,#1/5/2001#)
      h1 = .InsertItem(h,0,"Item 3")
      .AddBar(h1,"Task",#1/4/2001#,#1/6/2001#)
      .AddBar(h1,"Task",#1/1/2001#,#1/3/2001#,"green")
      .ItemBar(h1,"green",EXG2ANTTLib.ItemBarPropertyEnum.exBarColor) = 65280
      .AddBar(h1,"Task",#1/8/2001#,#1/10/2001#,"red")
      .ItemBar(h1,"red",EXG2ANTTLib.ItemBarPropertyEnum.exBarColor) = 255
      .ExpandItem(h) = True
      .SelectAll()
   End With
   .EndUpdate()
End With
```

C++ Is it possible to define the bar colors, and have the cumulative histogram showing the same colors?

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
```

```
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
spG2antt1->PutSingleSel(VARIANT_FALSE);
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutLevelCount(2);
   var_Chart->PutAllowLinkBars(VARIANT_FALSE);
   var_Chart->PutDrawGridLines(EXG2ANTTLib::exAllLines);
   var_Chart->PutFirstVisibleDate("12/29/2000");
   var_Chart->PutHistogramVisible(VARIANT_TRUE);
   var_Chart->PutHistogramHeight(72);
   var_Chart->PutPaneWidth(VARIANT_FALSE,64);
   var_Chart->PutHistogramView(EXG2ANTTLib::exHistogramSelectedItems |
EXG2ANTTLib::exHistogramUnlockedItems | EXG2ANTTLib::exHistogramLeafItems |
EXG2ANTTLib::exHistogramNoGrouping);
   EXG2ANTTLib::IBarPtr var_Bar = var_Chart->GetBars()->GetItem("Task");
      var_Bar->PutHistogramType(EXG2ANTTLib::exHistCumulative);
      var_Bar->PutHistogramItems(6);
      var_Bar->PutHistogramPattern(var_Bar->GetPattern());
      var_Bar-
>PutHistogramCumulativeOriginalColorBars(EXG2ANTTLib::exKeepOriginalColor);
spG2antt1->GetColumns()->Add(L"Column");
EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems();
   long h = var_Items->AddItem("Project");
   var_Items->PutItemBold(h,VARIANT_TRUE);
   var_Items->PutSelectableItem(h,VARIANT_FALSE);
   long h1 = var_Items->InsertItem(h,long(0),"Item 1");
   var_Items->AddBar(h1,"Task","1/2/2001","1/4/2001",vtMissing,vtMissing);
   h1 = var_Items->InsertItem(h,long(0),"Item 2");
   var_Items->AddBar(h1,"Task","1/3/2001","1/5/2001",vtMissing,vtMissing);
   h1 = var_Items->InsertItem(h,long(0),"Item 3");
   var_Items->AddBar(h1,"Task","1/4/2001","1/6/2001",vtMissing,vtMissing);
   var_Items->AddBar(h1,"Task","1/1/2001","1/3/2001","green",vtMissing);
   var_Items->PutItemBar(h1,"green",EXG2ANTTLib::exBarColor,long(65280));
   var_Items->AddBar(h1,"Task","1/8/2001","1/10/2001","red",vtMissing);
   var_Items->PutItemBar(h1,"red",EXG2ANTTLib::exBarColor,long(255));
   var_Items->PutExpandItem(h,VARIANT_TRUE);
```

```
    var_Items->SelectAll();
spG2antt1->EndUpdate();
```

C++ Builder Is it possible to define the bar colors, and have the cumulative histogram showing the same colors?

```
G2antt1->BeginUpdate();
G2antt1->SingleSel = false;
Exg2anttlib_tlb::IChartPtr var_Chart = G2antt1->Chart;
    var_Chart->LevelCount = 2;
    var_Chart->AllowLinkBars = false;
    var_Chart->DrawGridLines = Exg2anttlib_tlb::GridLinesEnum::exAllLines;
    var_Chart->set_FirstVisibleDate(TVariant(TDateTime(2000,12,29).operator double()));
    var_Chart->HistogramVisible = true;
    var_Chart->HistogramHeight = 72;
    var_Chart->set_PaneWidth(false,64);
    var_Chart->HistogramView =
Exg2anttlib_tlb::HistogramViewEnum::exHistogramSelectedItems |
Exg2anttlib_tlb::HistogramViewEnum::exHistogramUnlockedItems |
Exg2anttlib_tlb::HistogramViewEnum::exHistogramLeafItems |
Exg2anttlib_tlb::HistogramViewEnum::exHistogramNoGrouping;
    Exg2anttlib_tlb::IBarPtr var_Bar = var_Chart->Bars->get_Item(TVariant("Task"));
        var_Bar->HistogramType = Exg2anttlib_tlb::HistogramTypeEnum::exHistCumulative;
        var_Bar->HistogramItems = 6;
        var_Bar->HistogramPattern = var_Bar->Pattern;
        var_Bar->HistogramCumulativeOriginalColorBars =
Exg2anttlib_tlb::HistogramCumulativeOriginalColorBarsEnum::exKeepOriginalColor;
G2antt1->Columns->Add(L"Column");
Exg2anttlib_tlb::IItemsPtr var_Items = G2antt1->Items;
    long h = var_Items->AddItem(TVariant("Project"));
    var_Items->set_ItemBold(h,true);
    var_Items->set_SelectableItem(h,false);
    long h1 = var_Items->InsertItem(h,TVariant(0),TVariant("Item 1"));
    var_Items->AddBar(h1,TVariant("Task"),TVariant(TDateTime(2001,1,2).operator
double()),TVariant(TDateTime(2001,1,4).operator double()),TNoParam(),TNoParam());
    h1 = var_Items->InsertItem(h,TVariant(0),TVariant("Item 2"));
    var_Items->AddBar(h1,TVariant("Task"),TVariant(TDateTime(2001,1,3).operator
```

```
double()),TVariant(TDateTime(2001,1,5).operator double()),TNoParam(),TNoParam());
   h1 = var_Items->InsertItem(h,TVariant(0),TVariant("Item 3"));
   var_Items->AddBar(h1,TVariant("Task"),TVariant(TDateTime(2001,1,4).operator
double()),TVariant(TDateTime(2001,1,6).operator double()),TNoParam(),TNoParam());
   var_Items->AddBar(h1,TVariant("Task"),TVariant(TDateTime(2001,1,1).operator
double()),TVariant(TDateTime(2001,1,3).operator double()),TVariant("green"),TNoParam());
   var_Items-
>set_ItemBar(h1,TVariant("green"),Exg2anttlib_tlb::ItemBarPropertyEnum::exBarColor,TVaria

   var_Items->AddBar(h1,TVariant("Task"),TVariant(TDateTime(2001,1,8).operator
double()),TVariant(TDateTime(2001,1,10).operator double()),TVariant("red"),TNoParam());
   var_Items-
>set_ItemBar(h1,TVariant("red"),Exg2anttlib_tlb::ItemBarPropertyEnum::exBarColor,TVariant(

   var_Items->set_ExpandItem(h,true);
   var_Items->SelectAll();
G2antt1->EndUpdate();
```

C# Is it possible to define the bar colors, and have the cumulative histogram showing the same colors?

```
exg2antt1.BeginUpdate();
exg2antt1.SingleSel = false;
exontrol.EXG2ANTTLib.Chart var_Chart = exg2antt1.Chart;
   var_Chart.LevelCount = 2;
   var_Chart.AllowLinkBars = false;
   var_Chart.DrawGridLines = exontrol.EXG2ANTTLib.GridLinesEnum.exAllLines;
   var_Chart.FirstVisibleDate = Convert.ToDateTime("12/29/2000");
   var_Chart.HistogramVisible = true;
   var_Chart.HistogramHeight = 72;
   var_Chart.set_PaneWidth(false,64);
   var_Chart.HistogramView =
exontrol.EXG2ANTTLib.HistogramViewEnum.exHistogramSelectedItems |
exontrol.EXG2ANTTLib.HistogramViewEnum.exHistogramUnlockedItems |
exontrol.EXG2ANTTLib.HistogramViewEnum.exHistogramLeafItems |
exontrol.EXG2ANTTLib.HistogramViewEnum.exHistogramNoGrouping;
   exontrol.EXG2ANTTLib.Bar var_Bar = var_Chart.Bars["Task"];
```

```
      var_Bar.HistogramType =
exontrol.EXG2ANTTLib.HistogramTypeEnum.exHistCumulative;
      var_Bar.HistogramItems = 6;
      var_Bar.HistogramPattern = var_Bar.Pattern;
      var_Bar.HistogramCumulativeOriginalColorBars =
exontrol.EXG2ANTTLib.HistogramCumulativeOriginalColorBarsEnum.exKeepOriginalColor;

exg2antt1.Columns.Add("Column");
exontrol.EXG2ANTTLib.Items var_Items = exg2antt1.Items;
   int h = var_Items.AddItem("Project");
   var_Items.set_ItemBold(h,true);
   var_Items.set_SelectableItem(h,false);
   int h1 = var_Items.InsertItem(h,0,"Item 1");

var_Items.AddBar(h1,"Task",Convert.ToDateTime("1/2/2001"),Convert.ToDateTime("1/4/200

   h1 = var_Items.InsertItem(h,0,"Item 2");

var_Items.AddBar(h1,"Task",Convert.ToDateTime("1/3/2001"),Convert.ToDateTime("1/5/200

   h1 = var_Items.InsertItem(h,0,"Item 3");

var_Items.AddBar(h1,"Task",Convert.ToDateTime("1/4/2001"),Convert.ToDateTime("1/6/200


var_Items.AddBar(h1,"Task",Convert.ToDateTime("1/1/2001"),Convert.ToDateTime("1/3/200


var_Items.set_ItemBar(h1,"green",exontrol.EXG2ANTTLib.ItemBarPropertyEnum.exBarColor,6


var_Items.AddBar(h1,"Task",Convert.ToDateTime("1/8/2001"),Convert.ToDateTime("1/10/20


var_Items.set_ItemBar(h1,"red",exontrol.EXG2ANTTLib.ItemBarPropertyEnum.exBarColor,255

   var_Items.set_ExpandItem(h,true);
```

```
   var_Items.SelectAll();
exg2antt1.EndUpdate();
```

JavaScript Is it possible to define the bar colors, and have the cumulative histogram showing the same colors?

```
<OBJECT classid="clsid:CD481F4D-2D25-4759-803F-752C568F53B7" id="G2antt1">
</OBJECT>

<SCRIPT LANGUAGE="JScript">
   G2antt1.BeginUpdate()

   G2antt1.SingleSel = false

   var var_Chart = G2antt1.Chart

      var_Chart.LevelCount = 2

      var_Chart.AllowLinkBars = false

      var_Chart.DrawGridLines = -1

      var_Chart.FirstVisibleDate = "12/29/2000"

      var_Chart.HistogramVisible = true

      var_Chart.HistogramHeight = 72

      var_Chart.PaneWidth(0) = 64

      var_Chart.HistogramView = 1298

      var var_Bar = var_Chart.Bars.Item("Task")

         var_Bar.HistogramType = 256

         var_Bar.HistogramItems = 6
```

```
        var_Bar.HistogramPattern = var_Bar.Pattern
        var_Bar.HistogramCumulativeOriginalColorBars = 1
G2antt1.Columns.Add("Column")
var var_Items = G2antt1.Items
    var h = var_Items.AddItem("Project")
    var_Items.ItemBold(h) = true
    var_Items.SelectableItem(h) = false
    var h1 = var_Items.InsertItem(h,0,"Item 1")
    var_Items.AddBar(h1,"Task","1/2/2001","1/4/2001",null,null)
    h1 = var_Items.InsertItem(h,0,"Item 2")
    var_Items.AddBar(h1,"Task","1/3/2001","1/5/2001",null,null)
    h1 = var_Items.InsertItem(h,0,"Item 3")
    var_Items.AddBar(h1,"Task","1/4/2001","1/6/2001",null,null)
    var_Items.AddBar(h1,"Task","1/1/2001","1/3/2001","green",null)
    var_Items.ItemBar(h1,"green",33) = 65280
    var_Items.AddBar(h1,"Task","1/8/2001","1/10/2001","red",null)
    var_Items.ItemBar(h1,"red",33) = 255
    var_Items.ExpandItem(h) = true
```

```
    var_Items.SelectAll()

  G2antt1.EndUpdate()

</SCRIPT>
```

C# for /COM Is it possible to define the bar colors, and have the cumulative histogram showing the same colors?

```
axG2antt1.BeginUpdate();
axG2antt1.SingleSel = false;
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
  var_Chart.LevelCount = 2;
  var_Chart.AllowLinkBars = false;
  var_Chart.DrawGridLines = EXG2ANTTLib.GridLinesEnum.exAllLines;
  var_Chart.FirstVisibleDate = Convert.ToDateTime("12/29/2000");
  var_Chart.HistogramVisible = true;
  var_Chart.HistogramHeight = 72;
  var_Chart.set_PaneWidth(false,64);
  var_Chart.HistogramView =
EXG2ANTTLib.HistogramViewEnum.exHistogramSelectedItems |
EXG2ANTTLib.HistogramViewEnum.exHistogramUnlockedItems |
EXG2ANTTLib.HistogramViewEnum.exHistogramLeafItems |
EXG2ANTTLib.HistogramViewEnum.exHistogramNoGrouping;
  EXG2ANTTLib.Bar var_Bar = var_Chart.Bars["Task"];
    var_Bar.HistogramType = EXG2ANTTLib.HistogramTypeEnum.exHistCumulative;
    var_Bar.HistogramItems = 6;
    var_Bar.HistogramPattern = var_Bar.Pattern;
    var_Bar.HistogramCumulativeOriginalColorBars =
EXG2ANTTLib.HistogramCumulativeOriginalColorBarsEnum.exKeepOriginalColor;
axG2antt1.Columns.Add("Column");
EXG2ANTTLib.Items var_Items = axG2antt1.Items;
  int h = var_Items.AddItem("Project");
  var_Items.set_ItemBold(h,true);
  var_Items.set_SelectableItem(h,false);
  int h1 = var_Items.InsertItem(h,0,"Item 1");
```

```
var_Items.AddBar(h1,"Task",Convert.ToDateTime("1/2/2001"),Convert.ToDateTime("1/4/200

   h1 = var_Items.InsertItem(h,0,"Item 2");

var_Items.AddBar(h1,"Task",Convert.ToDateTime("1/3/2001"),Convert.ToDateTime("1/5/200

   h1 = var_Items.InsertItem(h,0,"Item 3");

var_Items.AddBar(h1,"Task",Convert.ToDateTime("1/4/2001"),Convert.ToDateTime("1/6/200


var_Items.AddBar(h1,"Task",Convert.ToDateTime("1/1/2001"),Convert.ToDateTime("1/3/200


var_Items.set_ItemBar(h1,"green",EXG2ANTTLib.ItemBarPropertyEnum.exBarColor,65280);

var_Items.AddBar(h1,"Task",Convert.ToDateTime("1/8/2001"),Convert.ToDateTime("1/10/20

   var_Items.set_ItemBar(h1,"red",EXG2ANTTLib.ItemBarPropertyEnum.exBarColor,255);
   var_Items.set_ExpandItem(h,true);
   var_Items.SelectAll();
axG2antt1.EndUpdate();
```

X++ (Dynamics Ax 2009) Is it possible to define the bar colors, and have the cumulative histogram showing the same colors?

```
public void init()
{
   COM com_Bar,com_Chart,com_Items

   anytype var_Bar,var_Chart,var_Items

   int h,h1
```

```
    super()


    exg2antt1.BeginUpdate()

    exg2antt1.SingleSel(false)

    var_Chart = exg2antt1.Chart()
     com_Chart = var_Chart

        com_Chart.LevelCount(2)

        com_Chart.AllowLinkBars(false)

        com_Chart.DrawGridLines(-1/*exAllLines*/)


com_Chart.FirstVisibleDate(COMVariant::createFromDate(str2Date("12/29/2000",213)))

        com_Chart.HistogramVisible(true)

        com_Chart.HistogramHeight(72)

    /*should be called during the form's activate method*/     com_Chart.PaneWidth(0,64);
        com_Chart.HistogramView(1298/*exHistogramSelectedItems |
exHistogramUnlockedItems | exHistogramLeafItems | exHistogramNoGrouping*/)

        var_Bar = COM::createFromObject(com_Chart.Bars()).Item("Task")
     com_Bar = var_Bar

            com_Bar.HistogramType(256/*exHistCumulative*/)

            com_Bar.HistogramItems(6)

            com_Bar.HistogramPattern(com_Bar.Pattern())

            com_Bar.HistogramCumulativeOriginalColorBars(1/*exKeepOriginalColor*/)
```

```
exg2antt1.Columns().Add("Column")

var_Items = exg2antt1.Items()
 com_Items = var_Items

    h = com_Items.AddItem("Project")

    com_Items.ItemBold(h,true)

    com_Items.SelectableItem(h,false)

    h1 = com_Items.InsertItem(h,COMVariant::createFromInt(0),"Item 1")

com_Items.AddBar(h1,"Task",COMVariant::createFromDate(str2Date("1/2/2001",213)),COMV

    h1 = com_Items.InsertItem(h,COMVariant::createFromInt(0),"Item 2")

com_Items.AddBar(h1,"Task",COMVariant::createFromDate(str2Date("1/3/2001",213)),COMV

    h1 = com_Items.InsertItem(h,COMVariant::createFromInt(0),"Item 3")

com_Items.AddBar(h1,"Task",COMVariant::createFromDate(str2Date("1/4/2001",213)),COMV

com_Items.AddBar(h1,"Task",COMVariant::createFromDate(str2Date("1/1/2001",213)),COMV

    com_Items.ItemBar(h1,"green",33/*exBarColor*/,COMVariant::createFromInt(65280))
```

```
com_Items.AddBar(h1,"Task",COMVariant::createFromDate(str2Date("1/8/2001",213)),COMV


        com_Items.ItemBar(h1,"red",33/*exBarColor*/,COMVariant::createFromInt(255))

        com_Items.ExpandItem(h,true)

        com_Items.SelectAll()

    exg2antt1.EndUpdate()

}

/*
public void activate(boolean _active)
{
    super(_active)


    exg2antt1.Chart().PaneWidth(0,64)

}
*/
```

VFP Is it possible to define the bar colors, and have the cumulative histogram showing the same colors?

```
with thisform.G2antt1
    .BeginUpdate
    .SingleSel = .F.
    with .Chart
        .LevelCount = 2
        .AllowLinkBars = .F.
        .DrawGridLines = -1
        .FirstVisibleDate = {^2000-12-29}
        .HistogramVisible = .T.
        .HistogramHeight = 72
```

```
      .PaneWidth(0) = 64
      .HistogramView = 1298
      with .Bars.Item("Task")
         .HistogramType = 256
         .HistogramItems = 6
         .HistogramPattern = .Pattern
         .HistogramCumulativeOriginalColorBars = 1
      endwith
   endwith
   .Columns.Add("Column")
   with .Items
      h = .AddItem("Project")
      .ItemBold(h) = .T.
      .SelectableItem(h) = .F.
      h1 = .InsertItem(h,0,"Item 1")
      .AddBar(h1,"Task",{^2001-1-2},{^2001-1-4})
      h1 = .InsertItem(h,0,"Item 2")
      .AddBar(h1,"Task",{^2001-1-3},{^2001-1-5})
      h1 = .InsertItem(h,0,"Item 3")
      .AddBar(h1,"Task",{^2001-1-4},{^2001-1-6})
      .AddBar(h1,"Task",{^2001-1-1},{^2001-1-3},"green")
      .ItemBar(h1,"green",33) = 65280
      .AddBar(h1,"Task",{^2001-1-8},{^2001-1-10},"red")
      .ItemBar(h1,"red",33) = 255
      .ExpandItem(h) = .T.
      .SelectAll
   endwith
   .EndUpdate
endwith
```

dBASE Plus Is it possible to define the bar colors, and have the cumulative histogram showing the same colors?

```
local h,h1,oG2antt,var_Bar,var_Chart,var_Items

oG2antt = form.Activex1.nativeObject
oG2antt.BeginUpdate()
```

```
oG2antt.SingleSel = false
var_Chart = oG2antt.Chart
   var_Chart.LevelCount = 2
   var_Chart.AllowLinkBars = false
   var_Chart.DrawGridLines = -1
   var_Chart.FirstVisibleDate = "12/29/2000"
   var_Chart.HistogramVisible = true
   var_Chart.HistogramHeight = 72
   // var_Chart.PaneWidth(false) = 64
   with (oG2antt)
      TemplateDef = [Dim var_Chart]
      TemplateDef = var_Chart
      Template = [var_Chart.PaneWidth(false) = 64]
   endwith
   var_Chart.HistogramView = 1298 /*exHistogramSelectedItems |
exHistogramUnlockedItems | exHistogramLeafItems | exHistogramNoGrouping*/
   var_Bar = var_Chart.Bars.Item("Task")
      var_Bar.HistogramType = 256
      var_Bar.HistogramItems = 6
      var_Bar.HistogramPattern = var_Bar.Pattern
      var_Bar.HistogramCumulativeOriginalColorBars = 1
oG2antt.Columns.Add("Column")
var_Items = oG2antt.Items
   h = var_Items.AddItem("Project")
   // var_Items.ItemBold(h) = true
   with (oG2antt)
      TemplateDef = [Dim var_Items,h]
      TemplateDef = var_Items
      TemplateDef = h
      Template = [var_Items.ItemBold(h) = true]
   endwith
   // var_Items.SelectableItem(h) = false
   with (oG2antt)
      TemplateDef = [Dim var_Items,h]
      TemplateDef = var_Items
      TemplateDef = h
      Template = [var_Items.SelectableItem(h) = false]
```

```
  endwith
  h1 = var_Items.InsertItem(h,0,"Item 1")
  var_Items.AddBar(h1,"Task","01/02/2001","01/04/2001")
  h1 = var_Items.InsertItem(h,0,"Item 2")
  var_Items.AddBar(h1,"Task","01/03/2001","01/05/2001")
  h1 = var_Items.InsertItem(h,0,"Item 3")
  var_Items.AddBar(h1,"Task","01/04/2001","01/06/2001")
  var_Items.AddBar(h1,"Task","01/01/2001","01/03/2001","green")
  // var_Items.ItemBar(h1,"green",33) = 65280
  with (oG2antt)
    TemplateDef = [Dim var_Items,h1]
    TemplateDef = var_Items
    TemplateDef = h1
    Template = [var_Items.ItemBar(h1,"green",33) = 65280]
  endwith
  var_Items.AddBar(h1,"Task","01/08/2001","01/10/2001","red")
  // var_Items.ItemBar(h1,"red",33) = 255
  with (oG2antt)
    TemplateDef = [Dim var_Items,h1]
    TemplateDef = var_Items
    TemplateDef = h1
    Template = [var_Items.ItemBar(h1,"red",33) = 255]
  endwith
  // var_Items.ExpandItem(h) = true
  with (oG2antt)
    TemplateDef = [Dim var_Items,h]
    TemplateDef = var_Items
    TemplateDef = h
    Template = [var_Items.ExpandItem(h) = true]
  endwith
  var_Items.SelectAll()
oG2antt.EndUpdate()
```

XBasic (Alpha Five) Is it possible to define the bar colors, and have the cumulative histogram showing the same colors?

```
Dim h as N
```

```
Dim h1 as N
Dim oG2antt as P
Dim var_Bar as P
Dim var_Chart as P
Dim var_Items as P

oG2antt = topparent:CONTROL_ACTIVEX1.activex
oG2antt.BeginUpdate()
oG2antt.SingleSel = .f.
var_Chart = oG2antt.Chart
   var_Chart.LevelCount = 2
   var_Chart.AllowLinkBars = .f.
   var_Chart.DrawGridLines = -1
   var_Chart.FirstVisibleDate = {12/29/2000}
   var_Chart.HistogramVisible = .t.
   var_Chart.HistogramHeight = 72
   ' var_Chart.PaneWidth(.f.) = 64
   oG2antt.TemplateDef = "Dim var_Chart"
   oG2antt.TemplateDef = var_Chart
   oG2antt.Template = "var_Chart.PaneWidth(False) = 64"

   var_Chart.HistogramView = 1298 'exHistogramSelectedItems +
exHistogramUnlockedItems + exHistogramLeafItems + exHistogramNoGrouping
   var_Bar = var_Chart.Bars.Item("Task")
      var_Bar.HistogramType = 256
      var_Bar.HistogramItems = 6
      var_Bar.HistogramPattern = var_Bar.Pattern
      var_Bar.HistogramCumulativeOriginalColorBars = 1
oG2antt.Columns.Add("Column")
var_Items = oG2antt.Items
   h = var_Items.AddItem("Project")
   ' var_Items.ItemBold(h) = .t.
   oG2antt.TemplateDef = "Dim var_Items,h"
   oG2antt.TemplateDef = var_Items
   oG2antt.TemplateDef = h
   oG2antt.Template = "var_Items.ItemBold(h) = True"
```

```
  ' var_Items.SelectableItem(h) = .f.
  oG2antt.TemplateDef = "Dim var_Items,h"
  oG2antt.TemplateDef = var_Items
  oG2antt.TemplateDef = h
  oG2antt.Template = "var_Items.SelectableItem(h) = False"

  h1 = var_Items.InsertItem(h,0,"Item 1")
  var_Items.AddBar(h1,"Task",{01/02/2001},{01/04/2001})
  h1 = var_Items.InsertItem(h,0,"Item 2")
  var_Items.AddBar(h1,"Task",{01/03/2001},{01/05/2001})
  h1 = var_Items.InsertItem(h,0,"Item 3")
  var_Items.AddBar(h1,"Task",{01/04/2001},{01/06/2001})
  var_Items.AddBar(h1,"Task",{01/01/2001},{01/03/2001},"green")
  ' var_Items.ItemBar(h1,"green",33) = 65280
  oG2antt.TemplateDef = "Dim var_Items,h1"
  oG2antt.TemplateDef = var_Items
  oG2antt.TemplateDef = h1
  oG2antt.Template = "var_Items.ItemBar(h1,\"green\",33) = 65280"

  var_Items.AddBar(h1,"Task",{01/08/2001},{01/10/2001},"red")
  ' var_Items.ItemBar(h1,"red",33) = 255
  oG2antt.TemplateDef = "Dim var_Items,h1"
  oG2antt.TemplateDef = var_Items
  oG2antt.TemplateDef = h1
  oG2antt.Template = "var_Items.ItemBar(h1,\"red\",33) = 255"

  ' var_Items.ExpandItem(h) = .t.
  oG2antt.TemplateDef = "Dim var_Items,h"
  oG2antt.TemplateDef = var_Items
  oG2antt.TemplateDef = h
  oG2antt.Template = "var_Items.ExpandItem(h) = True"

  var_Items.SelectAll()
oG2antt.EndUpdate()
```

Delphi 8 (.NET only) Is it possible to define the bar colors, and have the cumulative histogram showing the same colors?

```
with AxG2antt1 do
begin
  BeginUpdate();
  SingleSel := False;
  with Chart do
  begin
    LevelCount := 2;
    AllowLinkBars := False;
    DrawGridLines := EXG2ANTTLib.GridLinesEnum.exAllLines;
    FirstVisibleDate := '12/29/2000';
    HistogramVisible := True;
    HistogramHeight := 72;
    PaneWidth[False] := 64;
    HistogramView :=
Integer(EXG2ANTTLib.HistogramViewEnum.exHistogramSelectedItems) Or
Integer(EXG2ANTTLib.HistogramViewEnum.exHistogramUnlockedItems) Or
Integer(EXG2ANTTLib.HistogramViewEnum.exHistogramLeafItems) Or
Integer(EXG2ANTTLib.HistogramViewEnum.exHistogramNoGrouping);
    with Bars.Item['Task'] do
    begin
      HistogramType := EXG2ANTTLib.HistogramTypeEnum.exHistCumulative;
      HistogramItems := 6;
      HistogramPattern := Pattern;
      HistogramCumulativeOriginalColorBars :=
EXG2ANTTLib.HistogramCumulativeOriginalColorBarsEnum.exKeepOriginalColor;
    end;
  end;
  Columns.Add('Column');
  with Items do
  begin
    h := AddItem('Project');
    ItemBold[h] := True;
    SelectableItem[h] := False;
    h1 := InsertItem(h,TObject(0),'Item 1');
    AddBar(h1,'Task','1/2/2001','1/4/2001',Nil,Nil);
    h1 := InsertItem(h,TObject(0),'Item 2');
    AddBar(h1,'Task','1/3/2001','1/5/2001',Nil,Nil);
```

```
      h1 := InsertItem(h,TObject(0),'Item 3');
      AddBar(h1,'Task','1/4/2001','1/6/2001',Nil,Nil);
      AddBar(h1,'Task','1/1/2001','1/3/2001','green',Nil);
      ItemBar[h1,'green',EXG2ANTTLib.ItemBarPropertyEnum.exBarColor] :=
TObject(65280);
      AddBar(h1,'Task','1/8/2001','1/10/2001','red',Nil);
      ItemBar[h1,'red',EXG2ANTTLib.ItemBarPropertyEnum.exBarColor] := TObject(255);
      ExpandItem[h] := True;
      SelectAll();
   end;
   EndUpdate();
end
```

Delphi (standard) Is it possible to define the bar colors, and have the cumulative histogram showing the same colors?

```
with G2antt1 do
begin
   BeginUpdate();
   SingleSel := False;
   with Chart do
   begin
      LevelCount := 2;
      AllowLinkBars := False;
      DrawGridLines := EXG2ANTTLib_TLB.exAllLines;
      FirstVisibleDate := '12/29/2000';
      HistogramVisible := True;
      HistogramHeight := 72;
      PaneWidth[False] := 64;
      HistogramView := Integer(EXG2ANTTLib_TLB.exHistogramSelectedItems) Or
Integer(EXG2ANTTLib_TLB.exHistogramUnlockedItems) Or
Integer(EXG2ANTTLib_TLB.exHistogramLeafItems) Or
Integer(EXG2ANTTLib_TLB.exHistogramNoGrouping);
      with Bars.Item['Task'] do
      begin
         HistogramType := EXG2ANTTLib_TLB.exHistCumulative;
         HistogramItems := 6;
```

```
        HistogramPattern := Pattern;
        HistogramCumulativeOriginalColorBars :=
EXG2ANTTLib_TLB.exKeepOriginalColor;
      end;
   end;
   Columns.Add('Column');
   with Items do
   begin
      h := AddItem('Project');
      ItemBold[h] := True;
      SelectableItem[h] := False;
      h1 := InsertItem(h,OleVariant(0),'Item 1');
      AddBar(h1,'Task','1/2/2001','1/4/2001',Null,Null);
      h1 := InsertItem(h,OleVariant(0),'Item 2');
      AddBar(h1,'Task','1/3/2001','1/5/2001',Null,Null);
      h1 := InsertItem(h,OleVariant(0),'Item 3');
      AddBar(h1,'Task','1/4/2001','1/6/2001',Null,Null);
      AddBar(h1,'Task','1/1/2001','1/3/2001','green',Null);
      ItemBar[h1,'green',EXG2ANTTLib_TLB.exBarColor] := OleVariant(65280);
      AddBar(h1,'Task','1/8/2001','1/10/2001','red',Null);
      ItemBar[h1,'red',EXG2ANTTLib_TLB.exBarColor] := OleVariant(255);
      ExpandItem[h] := True;
      SelectAll();
   end;
   EndUpdate();
end
```

Visual Objects Is it possible to define the bar colors, and have the cumulative histogram showing the same colors?

```
local var_Bar as IBar
local var_Chart as IChart
local var_Items as IItems
local h,h1 as USUAL

oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:SingleSel := false
```

```
var_Chart := oDCOCX_Exontrol1:Chart
   var_Chart:LevelCount := 2
   var_Chart:AllowLinkBars := false
   var_Chart:DrawGridLines := exAllLines
   var_Chart:FirstVisibleDate := SToD("20001229")
   var_Chart:HistogramVisible := true
   var_Chart:HistogramHeight := 72
   var_Chart:[PaneWidth,false] := 64
   var_Chart:HistogramView := exHistogramSelectedItems | exHistogramUnlockedItems |
exHistogramLeafItems | exHistogramNoGrouping
   var_Bar := var_Chart:Bars:[Item,"Task"]
      var_Bar:HistogramType := exHistCumulative
      var_Bar:HistogramItems := 6
      var_Bar:HistogramPattern := var_Bar:Pattern
      var_Bar:HistogramCumulativeOriginalColorBars := exKeepOriginalColor
oDCOCX_Exontrol1:Columns:Add("Column")
var_Items := oDCOCX_Exontrol1:Items
   h := var_Items:AddItem("Project")
   var_Items:[ItemBold,h] := true
   var_Items:[SelectableItem,h] := false
   h1 := var_Items:InsertItem(h,0,"Item 1")
   var_Items:AddBar(h1,"Task",SToD("20010102"),SToD("20010104"),nil,nil)
   h1 := var_Items:InsertItem(h,0,"Item 2")
   var_Items:AddBar(h1,"Task",SToD("20010103"),SToD("20010105"),nil,nil)
   h1 := var_Items:InsertItem(h,0,"Item 3")
   var_Items:AddBar(h1,"Task",SToD("20010104"),SToD("20010106"),nil,nil)
   var_Items:AddBar(h1,"Task",SToD("20010101"),SToD("20010103"),"green",nil)
   var_Items:[ItemBar,h1,"green",exBarColor] := 65280
   var_Items:AddBar(h1,"Task",SToD("20010108"),SToD("20010110"),"red",nil)
   var_Items:[ItemBar,h1,"red",exBarColor] := 255
   var_Items:[ExpandItem,h] := true
   var_Items:SelectAll()
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder Is it possible to define the bar colors, and have the cumulative histogram showing the same colors?

```
OleObject oG2antt,var_Bar,var_Chart,var_Items
any h,h1

oG2antt = ole_1.Object
oG2antt.BeginUpdate()
oG2antt.SingleSel = false
var_Chart = oG2antt.Chart
   var_Chart.LevelCount = 2
   var_Chart.AllowLinkBars = false
   var_Chart.DrawGridLines = -1
   var_Chart.FirstVisibleDate = 2000-12-29
   var_Chart.HistogramVisible = true
   var_Chart.HistogramHeight = 72
   var_Chart.PaneWidth(false,64)
   var_Chart.HistogramView = 1298 /*exHistogramSelectedItems |
exHistogramUnlockedItems | exHistogramLeafItems | exHistogramNoGrouping*/
   var_Bar = var_Chart.Bars.Item("Task")
      var_Bar.HistogramType = 256
      var_Bar.HistogramItems = 6
      var_Bar.HistogramPattern = var_Bar.Pattern
      var_Bar.HistogramCumulativeOriginalColorBars = 1
oG2antt.Columns.Add("Column")
var_Items = oG2antt.Items
   h = var_Items.AddItem("Project")
   var_Items.ItemBold(h,true)
   var_Items.SelectableItem(h,false)
   h1 = var_Items.InsertItem(h,0,"Item 1")
   var_Items.AddBar(h1,"Task",2001-01-02,2001-01-04)
   h1 = var_Items.InsertItem(h,0,"Item 2")
   var_Items.AddBar(h1,"Task",2001-01-03,2001-01-05)
   h1 = var_Items.InsertItem(h,0,"Item 3")
   var_Items.AddBar(h1,"Task",2001-01-04,2001-01-06)
   var_Items.AddBar(h1,"Task",2001-01-01,2001-01-03,"green")
   var_Items.ItemBar(h1,"green",33,65280)
   var_Items.AddBar(h1,"Task",2001-01-08,2001-01-10,"red")
   var_Items.ItemBar(h1,"red",33,255)
   var_Items.ExpandItem(h,true)
```

```
    var_Items.SelectAll()
oG2antt.EndUpdate()
```

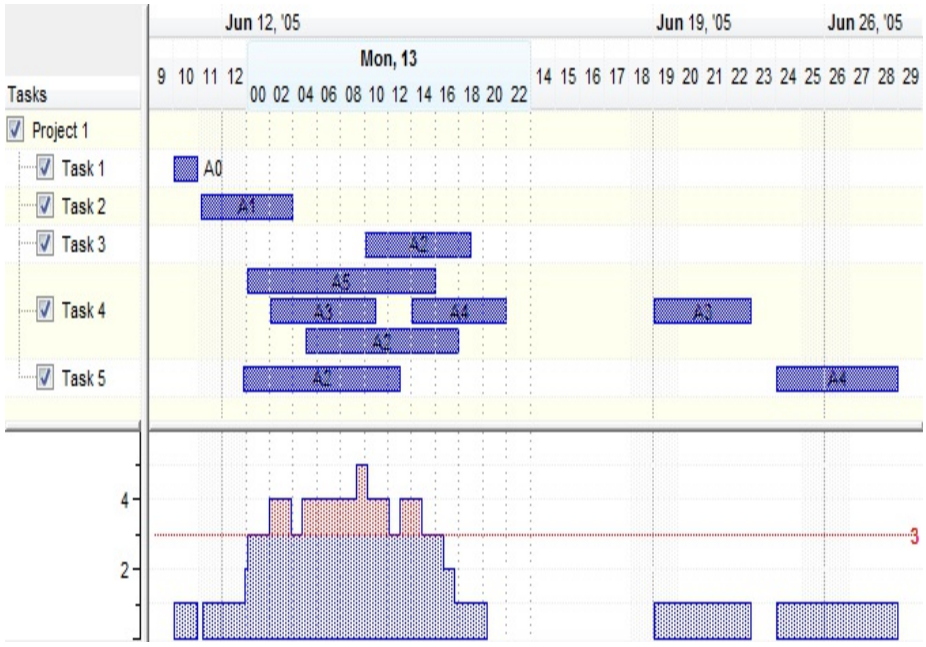
constants HistogramTypeEnum

The HistogramTypeEnum type specifies the types of the histogram that currently the control supports. Use the HistogramType property to specify the histogram-graph to be displayed for a specified type of bar.

Changes the HistogramPattern or/and HistogramColor property, else no bars will be shown in the histogram.

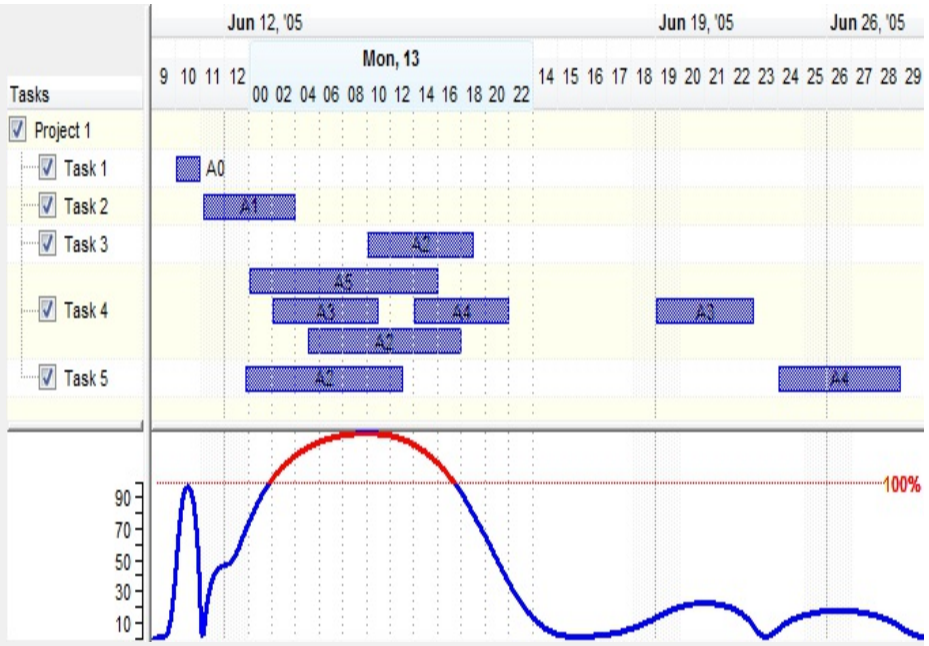
Use the HistogramBorderColor property to define the color to draw the frame arround the histogram from rectangular patterns, or the color to show the curve, when non-rectangular values are used for HistogramPattern property. The HistogramBorderSize property defines the size of the curve when showing in the histogram. Use the ResizeScaleUnit property to refine the histogram based on the resizing unit. Use the HistogramCriticalValue property to define a critical value. Use the HistogramCriticalColor property to define the color to show the values in the histogram greater than critical values. Use the HistogramRulerLinesColor property to specify the color to show the ruler in the left part of the histogram.

Name	Value	Description
exHistOverload	0	The histogram shows the overloads and subloads of your current planning situation. The histogram-graph shows the count of specified tasks day by day, or unit by unit. Use the HistogramItems property to specify the number of items being displayed in the histogram. Use the HistogramGridLinesColor property to specify the color to show the grid lines in the histogram.



exHistOverAllocation	1

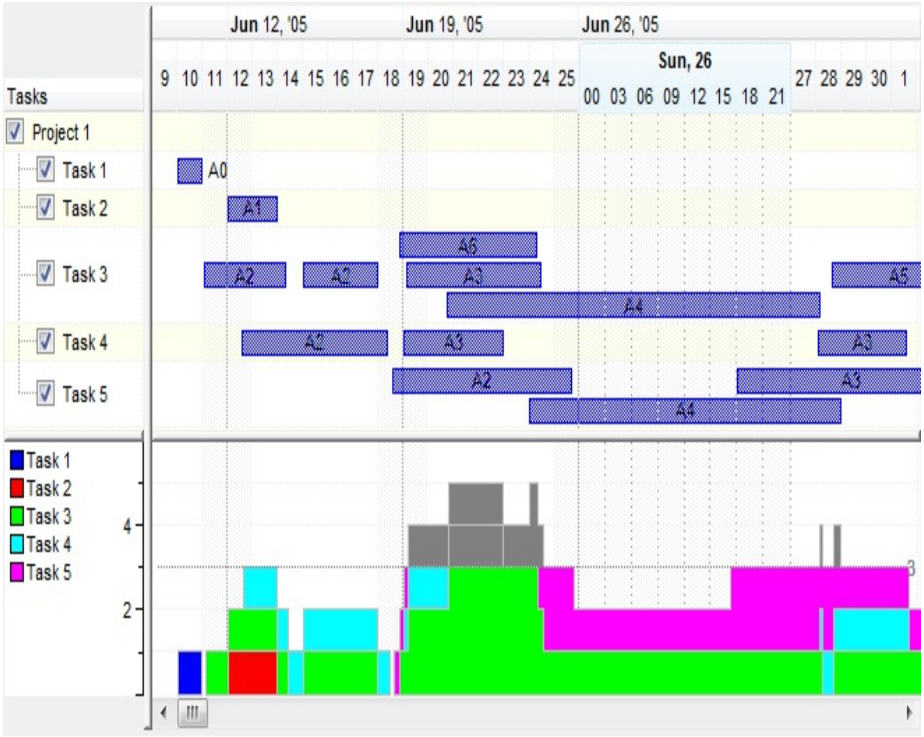
The histogram shows in percents, the over-allocations of your current planning situation using the effort of the task divided by the length of the task (effort/length). The exHistOverAllocation flag can be combined with exHistOverAllocationFixed or exHistOverAllocationMultiply. The work-load for a task is computed as exBarEffort / length of the bar. The work-load for the task is the work effort / task duration. (i.e. If item.exBarEffort = 1 and bar's length is 10 days, then the work-load = 0.1 or 10%). The histogram- graph shows the sum of the work-loads (the work-load of each task item is added, unit by unit).



The bars in the histogram shows cumulative colors. The exHistCumulative can be applied to exHistOverload and exHistOverAllocation values. For instance, the exHistOverAllocation + exHistCumulative defines a cumulative histogram for exHistOverAllocation type. The HistogramCumulativeColors property defines the number of colors being displayed in the cumulative histogram. The HistogramCumulativeColor property specifies a cumulative color based on its index. Use the HistogramCumulativeShowLegend property to specify whether the index of the column being shown in the left side of the histogram to show the legend of the colors being used for cumulative bars. The HistogramPattern property should not be a curve, in order to show a cumulative histogram, in other words should be a predefined pattern. You can change the original color of the bars that generates the cumulative histogram using the HistogramCumulativeOriginalColorBars property.

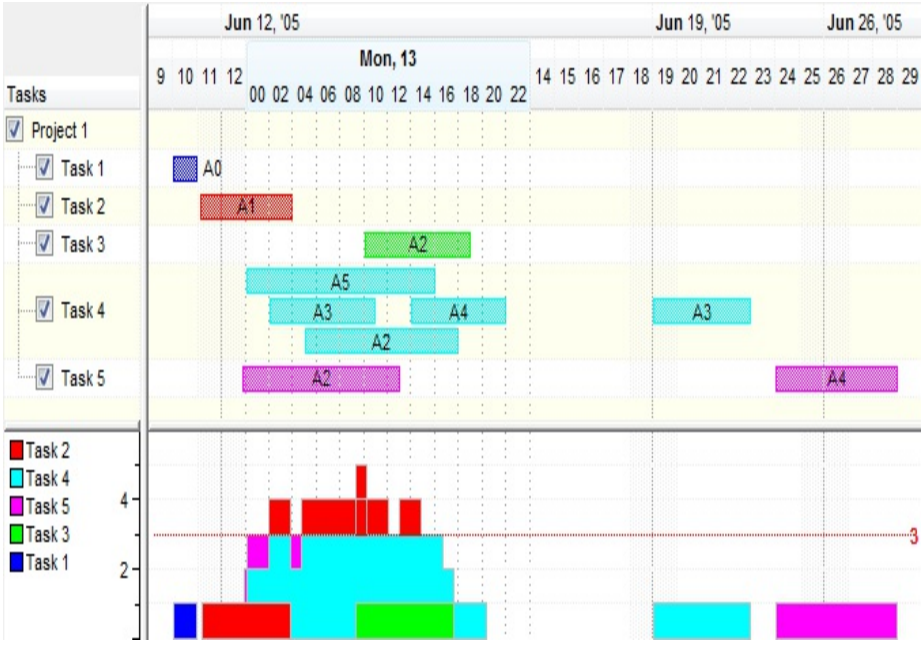
The following screen shot shows the bars using the original color in the items that generates the histogram (when HistogramCumulativeOriginalColorBars property is

-1 (True), by default). All bars in the same item does not change their color, instead the reflection of the bar in the histogram gets a cumulative color.

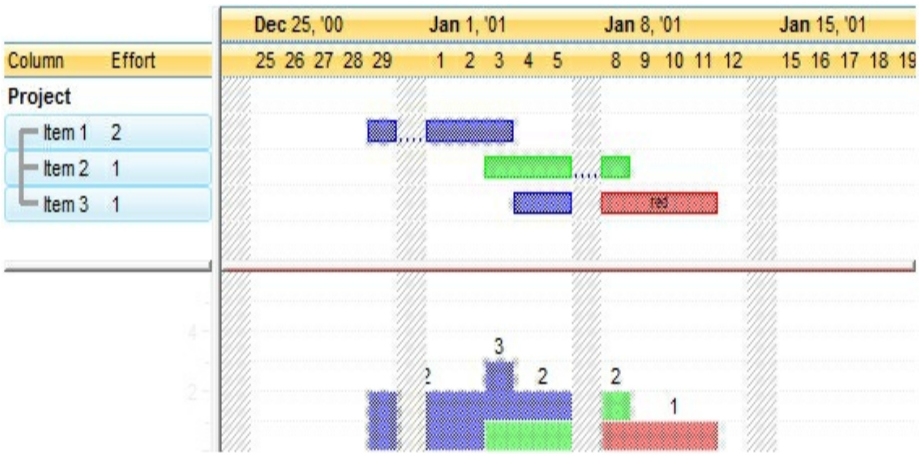


exHistCumulative 256

The following screen shot shows the bars that generates the cumulative histogram using cumulative colors when HistogramCumulativeOriginalColorBars property is 0 (False). All bars in the same item gets a cumulative color.

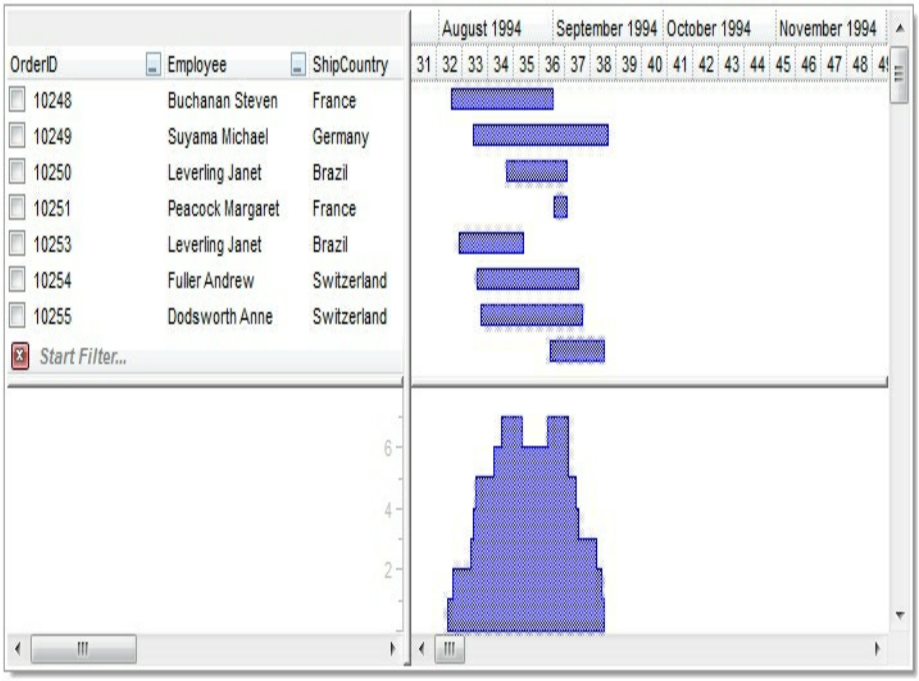


The following screen shot shows the bars that generates the cumulative histogram using

		cumulative colors when HistogramCumulativeOriginalColorBars property is 1 (exKeepOriginalColor). The bars keeps their original color in the histogram. 
exHistOverAllocationFixed	512	The histogram shows in percents, the over-allocations of your current planning situation using the effort of the task (effort). The exHistOverAllocationFixed can be combined with the exHistOverAllocation flag only.
exHistOverAllocationMultiply	1024	The histogram shows in percents, the over-allocations of your current planning situation using the effort of the task multiplied by the length of the task (effort * length). The exHistOverAllocationFixed can be combined with the exHistOverAllocation flag only.

constants HistogramViewEnum

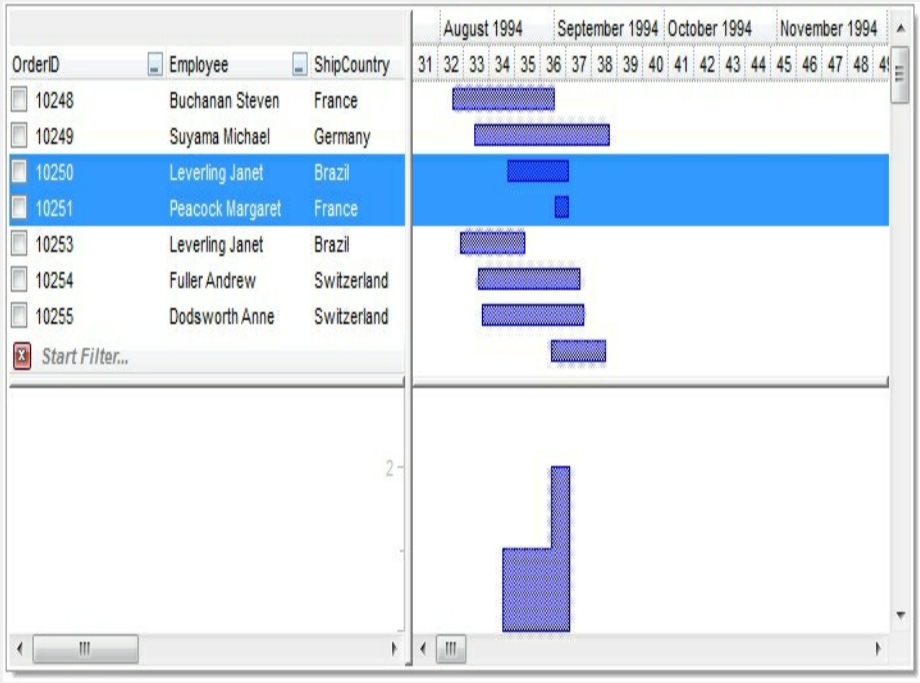
The HistogramViewEnum type specifies the items being included in the histogram. Use the HistogramView property to specify the items or bars being displayed in the histogram. The HistogramViewEnum type supports the following values:

Name	Value	Description
exHistogramVisibleItems	1	The histogram is shown for the visible items only. The Histogram is updated as soon as the control changes its first visible item (FirstVisibleItem, NextVisibleItem and IsItemVisible properties determines the items to be included in the histogram), in other words the control gets vertically scrolled. This flag can be combined with • exHistogramLeafItems • exHistogramRecLeafItems • exHistogramNoGrouping 
		The histogram is shown for the selected items only (SelectCount, SelectedItem property determines the items to be shown in the chart's histogram). Use the SingleSel property to specify whether the control can select multiple items. Use the SelectOnClick property to disable selecting new items when the user clicks the chart area. The Histogram is updated as soon as the selection is

changed. This flag can be combined with:

- exHistogramLeafItems
- exHistogramRecLeafItems
- exHistogramNoGrouping

exHistogramSelectedItems 2

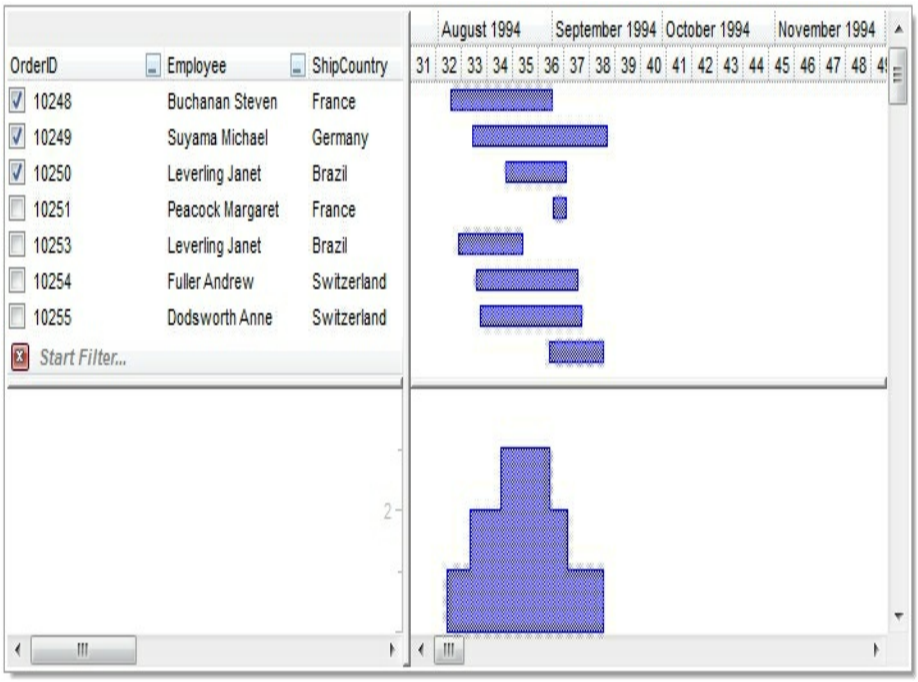


exHistogramCheckedItems 4

The histogram is shown for the checked items only. *You must combine this with exHistogramUnlockedItems, exHistogramLockedTopItems or exHistogramLockedBottomItems.* The CellState property specifies the state of the cell / item. The histogram includes only items that have the CellState property on 1 (locked and unlocked items). By default, the check box should be on the first column (the column with the index 0). Use the high word of the HistogramView property to specify a different column. For instance, if you need to display the histogram based on the check boxes of the column index 5, the HistogramView property should be 0x5**0000 + exHistogramCheckedItems + exHistogramUnlockedItems**. *Another sample, if the HistogramView property is exHistogramCheckedItems + exHistogramLockedBottomItems the histogram shows only the checked items in the bottom locked area.* The Histogram is updated as soon as the user changes the state of the cell's check box. This

flag can be combined with:

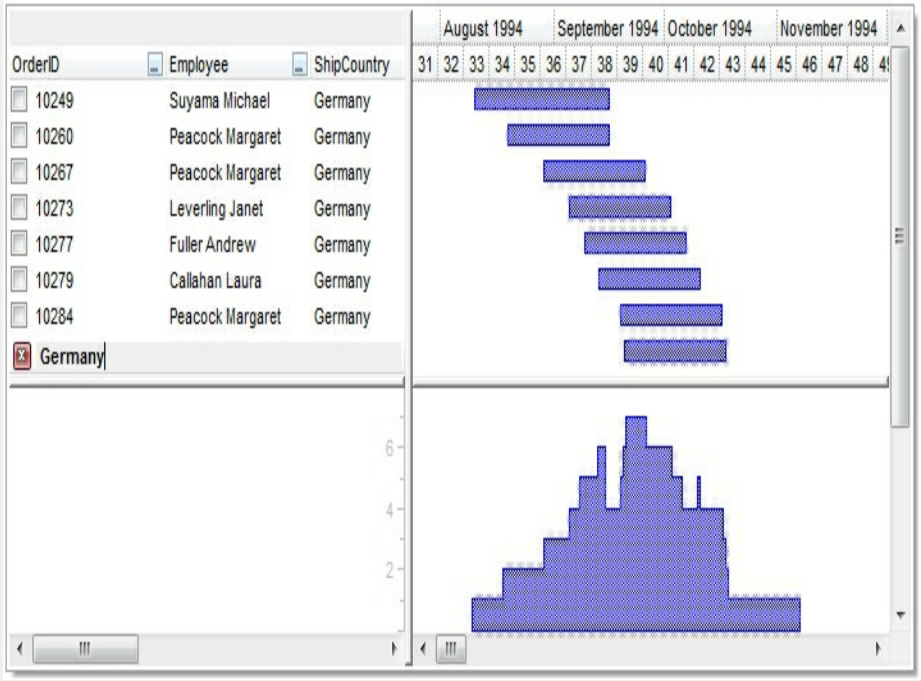
- exHistogramLeafItems
- exHistogramRecLeafItems
- exHistogramNoGrouping



The histogram is shown for the filtered items only. The Histogram is updated as soon as the user changes the control's filter. This flag can be combined with:

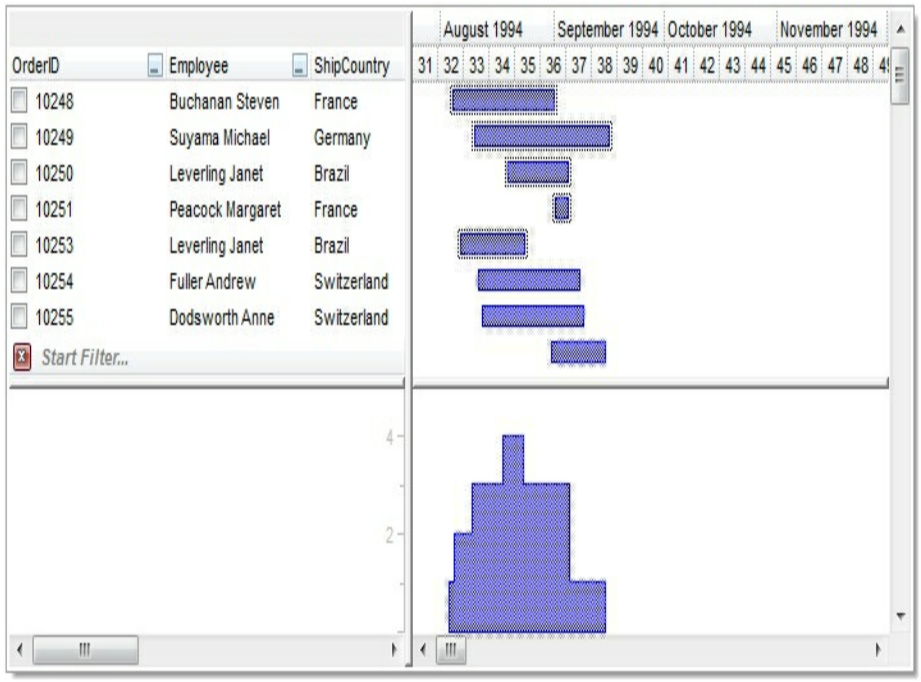
- exHistogramLeafItems
- exHistogramRecLeafItems
- exHistogramNoGrouping

exHistogramFilteredItems 128



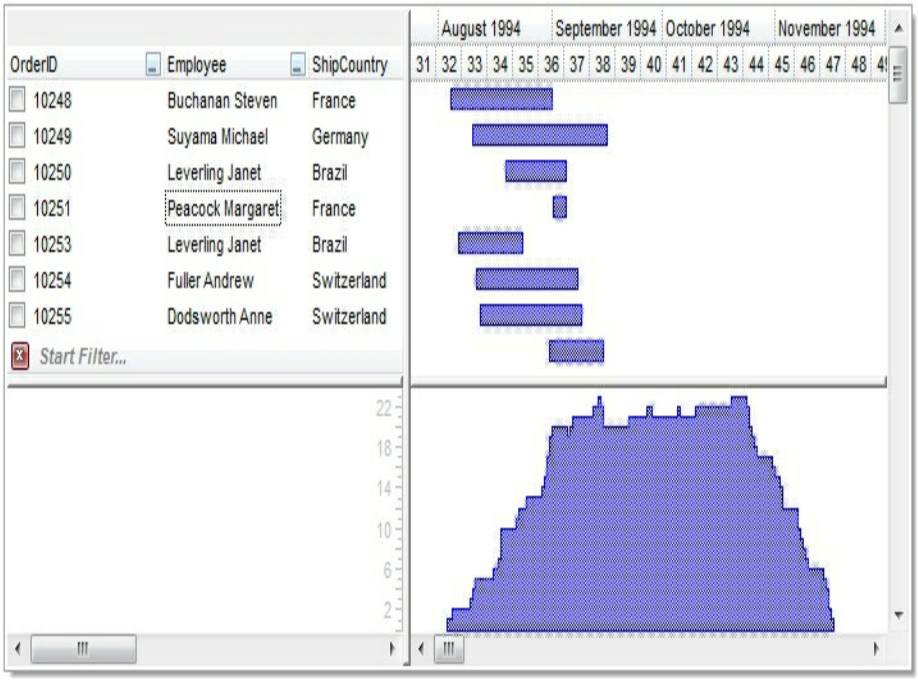
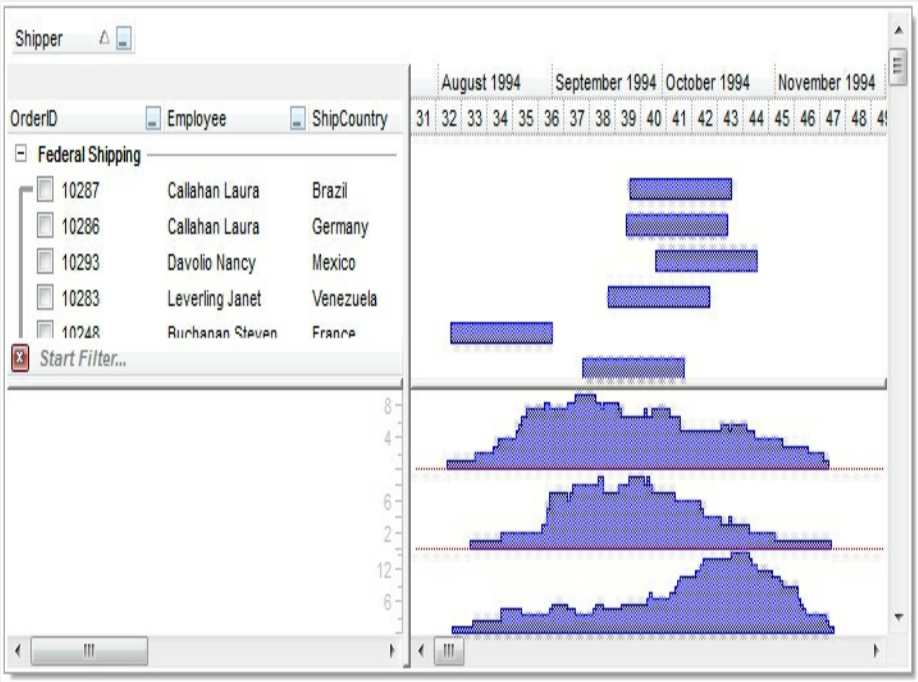
The histogram is shown for the selected bars only. The ItemBar(exBarSelected) property specifies whether a bar is selected or unselected. The ChartSelectionChanged event notifies the application once a new bar is selected or unselected.

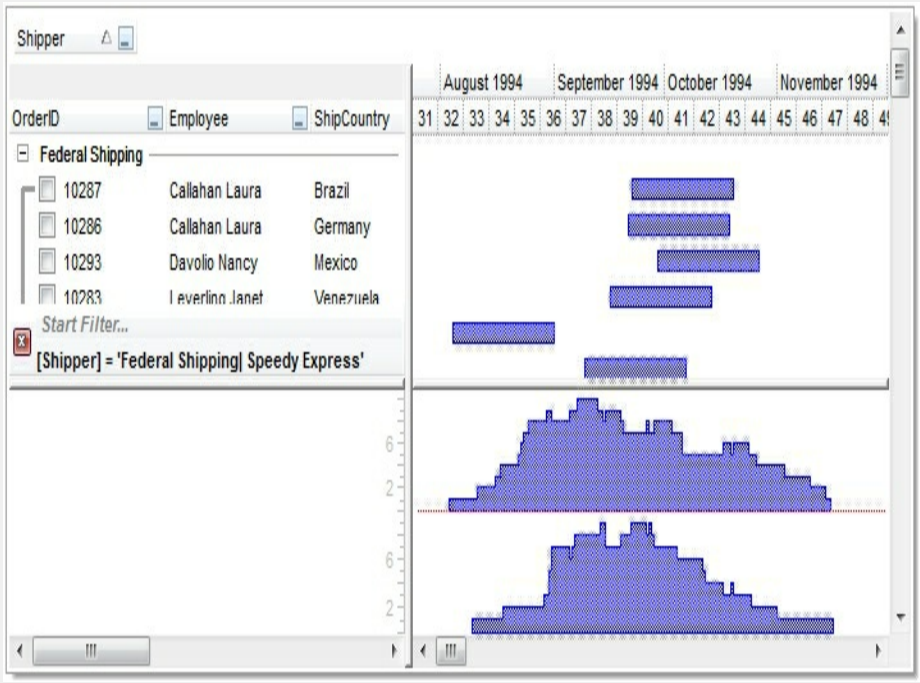
exHistogramSelectedBars 8

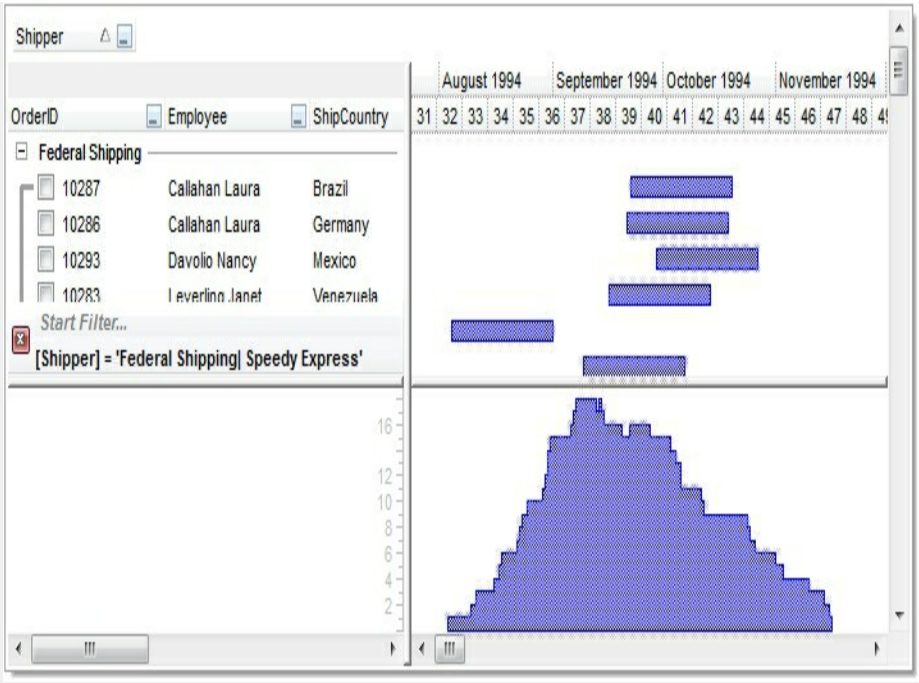


The histogram is shown only for unlocked items. Use the AddItem/InsertItem methods to add unlocked items. This option can be combined with exHistogramCheckedItems, exHistogramLockedTopItems or

exHistogramUnlockedItems	16	exHistogramLockedBottomItems. *For instance, if the HistogramView property is exHistogramUnlockedItems + exHistogramLockedTopItems the histogram shows all the items in the unlocked plus the items in the top locked area.*
exHistogramLockedTopItems	32	The histogram is shown only for locked items in the top side of the control. Use the LockedItemCount property to specify how many items are in the locked area. This option can be combined with exHistogramCheckedItems, exHistogramUnlockedItems or exHistogramLockedBottomItems. *For instance, if the HistogramView property is exHistogramUnlockedItems + exHistogramLockedTopItems the histogram shows all the items in the unlocked plus the items in the top locked area.*
exHistogramLockedBottomItems	64	The histogram is shown only for locked items in the bottom side of the control. Use the LockedItemCount property to specify how many items are in the locked area. This option can be combined with exHistogramCheckedItems, exHistogramUnlockedItems or exHistogramLockedTopItems. *For instance, if the HistogramView property is exHistogramUnlockedItems + exHistogramLockedBottomItems the histogram shows all the items in the unlocked plus the items in the bottom locked area.*
		The histogram is shown for all items, locked and unlocked items too. The exHistogramAllItems is a shortcut for the exHistogramUnlockedItems + exHistogramLockedTopItems + exHistogramLockedBottomItems. This flag can be combined with: • exHistogramLeafItems • exHistogramRecLeafItems • exHistogramNoGrouping

exHistogramAllItems	112	
exHistogramLeafItems	256	The histogram shows the bars for leaf items, in other words, the item itself if contains no child items, or all child items that contains no other child items. Use this flag to include in the histogram the bars in the child items too. 
exHistogramRecLeafItems	512	The histogram shows all bars for all recursive leaf items, so all child leaf items are displayed. Use this flag to include in the histogram the bars in all child items (recursively) too.
		(exHistogramNoGrouping/1024) If present, the histogram shows all bars without grouping based on

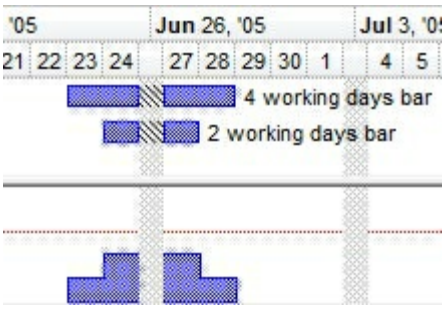
exHistogramNoGrouping	1024	the item's parent, and so all bars shares the same space for the histogram. If missing, the bars included in the histogram are grouped based on their parents, and each group has allocated a space in the histogram, so each group is shown separately. The exHistogramNoGrouping flag can not be combined with exHistogramNoGroupCaption or exHistogramGroupCumulative flag. The following screen shot shows the histogram grouped by parent items (exHistogramNoGrouping is **not** used):  The following screen shot shows the histogram without grouping (exHistogramNoGrouping is used):



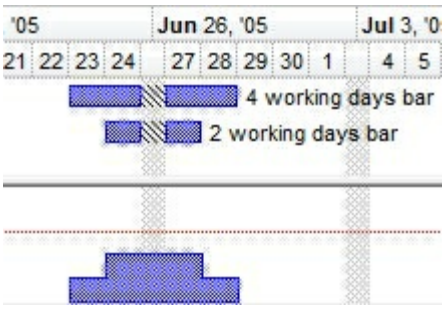
(exHistogramBackground/2048) The histogram's chart goes on the background, while the non-working part is shown on front (erases the non-working parts).

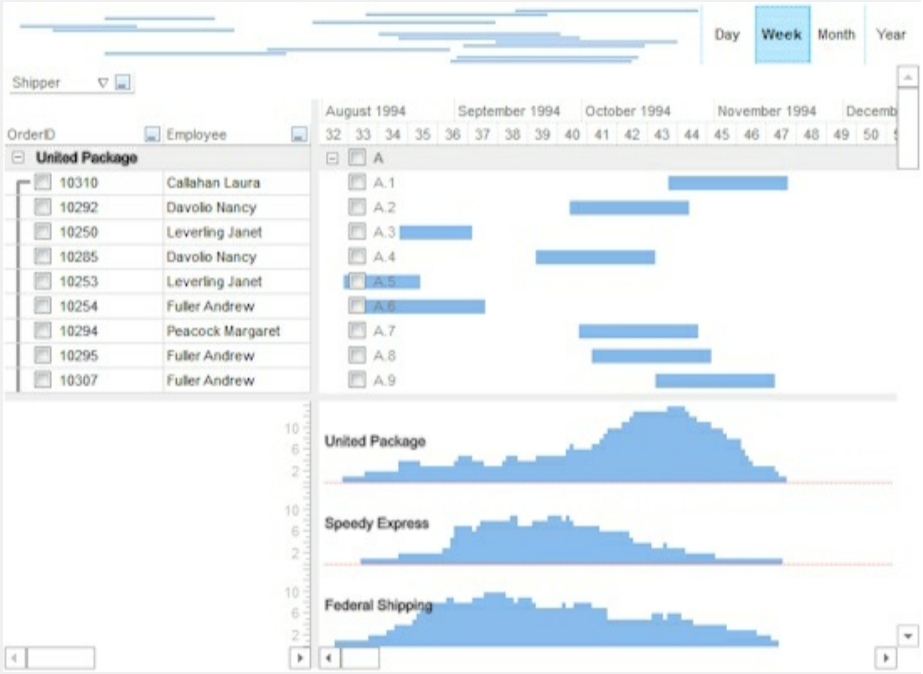
The following screen shot shows the histogram with the exHistogramBackground flag set (histogram on background):

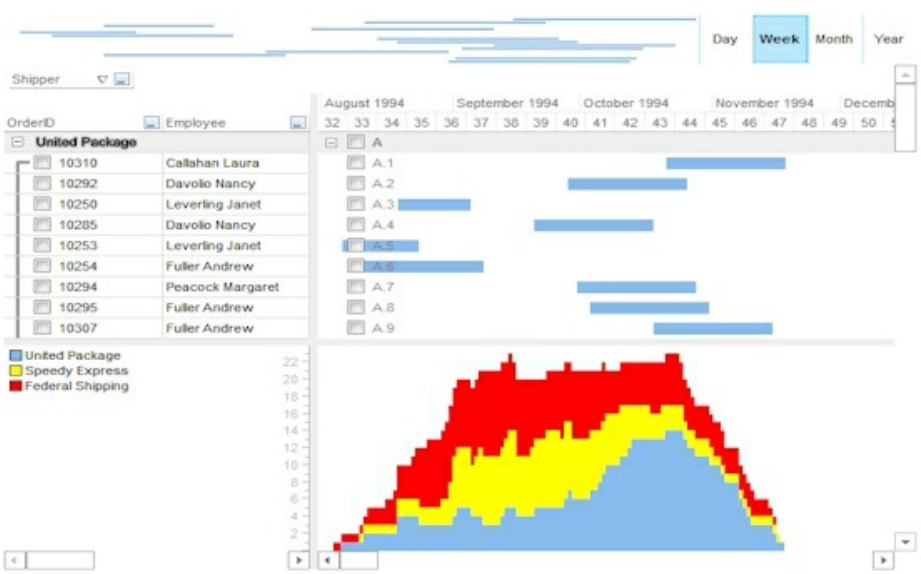
exHistogramBackground 2048



while the following shows the same histogram with no exHistogramBackground flag (histogram on front):



exHistogramNoGroupCaption 4096	(exHistogramNoGroupCaption/4096) The histogram shows no caption for groups being shown. The exHistogramNoGroupCaption flag has no effect if the exHistogramNoGrouping flag is set. The following screen shot shows shows the group captions when exHistogramNoGroupCaption flag is **not** set: 
exHistogramGroupCumulative 8192	(exHistogramGroupCumulative/8192) The histogram shows cumulative groups. The exHistogramGroupCumulative flag has no effect if the exHistogramNoGrouping flag is set. The HistogramCumulativeColors property of the Bar indicates the number of colors that can be used in the representation. The HistogramCumulativeColor property of the Bar specifies the color to be used in the representation. The following screen shot shows shows the histogram when the exHistogramGroupCumulative flag is set:



Sample 1, will display the histogram for **all** Task bars:

```
With G2antt1
   .BeginUpdate
   .SingleSel = False
   With .Chart
      .FirstVisibleDate = #1/1/2001#
      .LevelCount = 2
      .HistogramVisible = True
      .HistogramHeight = 32
      .HistogramView = 112
      .Bars.Item("Task").HistogramPattern = 6
   End With
   .Columns.Add "Column"
   With .Items
      .AddBar .AddItem("Item 1"),"Task",#1/2/2001#,#1/4/2001#
      .AddBar .AddItem("Item 2"),"Task",#1/3/2001#,#1/7/2001#
   End With
   .EndUpdate
End With
```

This sample displays the histogram for all Task bars no matter if the selected items/bars is changed.

Sample 2, will display the histogram for Task bars in the **selected items** only:

```
With G2antt1
```

```
    .BeginUpdate
    .SingleSel = False
    With .Chart
        .FirstVisibleDate = #1/1/2001#
        .HistogramVisible = True
        .HistogramView = 2
        .HistogramHeight = 32
        .Bars.Item("Task").HistogramPattern = 6
    End With
    .Columns.Add "Column"
    With .Items
        .AddBar .AddItem("Item 1"),"Task",#1/3/2001#,#1/5/2001#
        .AddBar .AddItem("Item 2"),"Task",#1/4/2001#,#1/7/2001#
        .AddBar .AddItem("Item 3"),"Task",#1/2/2001#,#1/6/2001#
        .SelectAll
    End With
    .EndUpdate
End With
```

This sample displays the histogram for all Task bars in the selected items. Run the sample and selects one or multiple items. The histogram is shown for bars in the selected items only.

Sample 3, will display the histogram for **selected** Task **bars** only:

```
With G2antt1
    .BeginUpdate
    With .Chart
        .LevelCount = 2
        .PaneWidth(False) = 64
        .FirstVisibleDate = #1/1/2001#
        .HistogramVisible = True
        .HistogramView = 8
        .HistogramHeight = 32
        .Bars.Item("Task").HistogramPattern = 6
    End With
    .Columns.Add "Column"
    With .Items
```

```
    .AddBar .AddItem("Item 1"),"Task",#1/3/2001#,#1/5/2001#,1
    .AddBar .AddItem("Item 2"),"Task",#1/4/2001#,#1/7/2001#,2
    .AddBar .AddItem("Item 3"),"Task",#1/2/2001#,#1/6/2001#,3
    .ItemBar(0,2,257) = True
    .ItemBar(0,3,257) = True
  End With
  .EndUpdate
End With
```

This sample displays the histogram for all Task bars in the selected items. Run the sample and selects one or multiple items. The histogram is shown for bars in the selected items only.

Sample 4, will display the histogram for **checked** Task **bars** only:

```
With G2antt1
  With .Chart
    .FirstVisibleDate = #1/1/2001#
    .HistogramVisible = True
    .HistogramHeight = 32
    .HistogramView = 276
    .Bars.Item("Task").HistogramPattern = 6
  End With
  .Columns.Add "Column"
  With .Items
    h = .AddItem("Project")
    .CellHasCheckBox(h,0) = True
    .AddBar .InsertItem(h,0,"Item 1"),"Task",#1/2/2001#,#1/4/2001#
    .AddBar .InsertItem(h,0,"Item 2"),"Task",#1/3/2001#,#1/7/2001#
    .ExpandItem(h) = True
  End With
End With
```

This sample displays the histogram for all Task bars in the checked items. Run the sample and check one or multiple items..

Sample 5, displays the non-working pattern over the bars:

```
With G2antt1
```

```
   .BeginUpdate
   .Columns.Add "Tasks"
   With .Chart
      .NonworkingDaysPattern = exPatternBDiagonal
      .NonworkingDaysColor = RGB(0,0,0)
      .PaneWidth(0) = 40
      .FirstVisibleDate = #6/20/2005#
      .HistogramVisible = True
      .HistogramHeight = 64
      .HistogramView = HistogramViewEnum.exHistogramUnlockedItems Or
HistogramViewEnum.exHistogramLockedTopItems Or
HistogramViewEnum.exHistogramLockedBottomItems Or
HistogramViewEnum.exHistogramBackground
      .LevelCount = 2
      With .Bars
         With .Add("Empty")
            .Color = RGB(0,0,0)
            .Pattern = exPatternFDiagonal
            .Shape = exShapeSolidFrameless
         End With
         With .Add("Task:Empty")
            .Shortcut = "Task"
            .HistogramItems = -5
            .HistogramCriticalValue = 3
            .HistogramType = exHistOverload
            .HistogramPattern = .Pattern
            .Def(exBarCaption) = "<%=%258%> working days bar"
            .Def(exBarHAlignCaption) = 18
            .Def(exBarKeepWorkingCount) = True
         End With
      End With
      .UnitWidthNonworking = -12
      .Level(1).FormatLabel = "weekday(dvalue) in (0,6) ? `` : value"
   End With
   With .Items
      .AddBar .AddItem("Task A"),"Task",#6/23/2005#,#6/29/2005#,""
      .AddBar .AddItem("Task B"),"Task",#6/24/2005#,#6/28/2005#,""
```

```
    End With
    .EndUpdate
End With
```

constants HitTestInfoEnum

The HitTestInfoEnum expression defines the hit area within a cell. Use the ItemFromPoint property to determine the hit test code within the cell.

Name	Value	Description
exHTCell	0	In the cell's client area.
exHTExpandButton	1	In the +/- button associated with a cell. The HasButtons property specifies whether the cell displays a +/- sign to let user expands the item.
exHTCellIndent	2	In the indentation associated with a cell. The Indent property retrieves or sets the amount, in pixels, that child items are indented relative to their parent items.
exHTCellInside	4	On the icon, picture, check or caption associated with a cell.
exHTCellCaption	20	(HEXA 14) In the caption associated with a cell. The CellValue property specifies the cell's value.
exHTCellCheck	36	(HEXA 24) In the check/radio button associated with a cell. The CellHasCheckBox or CellHasRadioButton property specifies whether the cell displays a checkbox or a radio button.
exHTCellIcon	68	(HEXA 44) In first icon associated with a cell. The CellImage or CellImages property specifies the cell's icon displayed next to the cell's caption.
exHTCellPicture	132	(HEXA 84)In a picture associated to a cell.
exHTCellCaptionIcon	1044	(HEXA 414) In the icon's area inside the cell's caption. The <img> tag inserts an icon inside the cell's caption. The <img> tag is valid only if the CellValueFormat property exHTML
exHTBottomHalf	2048	(HEXA 800) The cursor is in the bottom half of the row. If this flag is not set, the cursor is in the top half of the row. This is an OR combination with the rest of predefined values. For instance, you can check if the cursor is in the bottom half of the row using HitTestCode AND 0x800
exHTBetween	4096	(HEXA 1000) The cursor is between two rows. This is an OR combination with the rest of predefined values. For instance, you can check if the cursor is

		between two items using HitTestCode AND 0x1000
exHTItemChart	256	(HEXA 100) The cursor is in the chart's area over an item.

constants LevelLineEnum

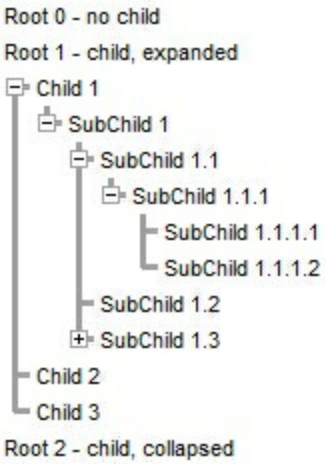
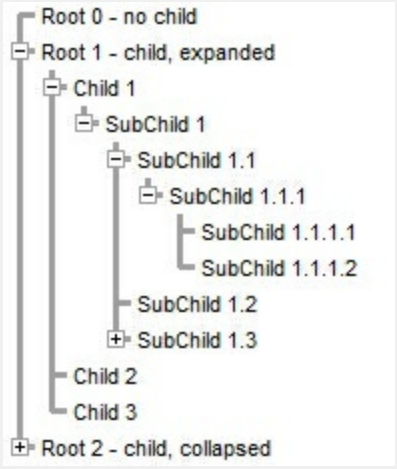
The LevelLineEnum type specifies the style of lines being shown in the chart's levels. Use the DrawLevelSeparator, DrawTickLines and DrawTickLinesFrom properties to show or hide lines in the chart's header. For instance, if the DrawTickLines property is exLevelSolidLine + exLevelMiddleLine, the level shows tick lines in the middle part of the time unit. The DrawTickLines and DrawTickLinesFrom properties draw vertically the lines, while the DrawLevelSeparator property draw horizontally the line. The LevelLineEnum type supports the following values.

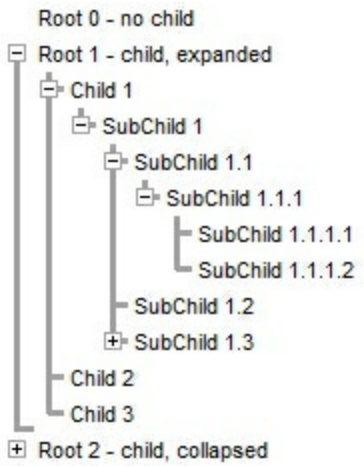
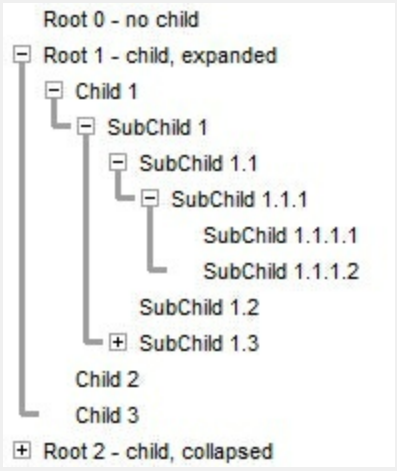
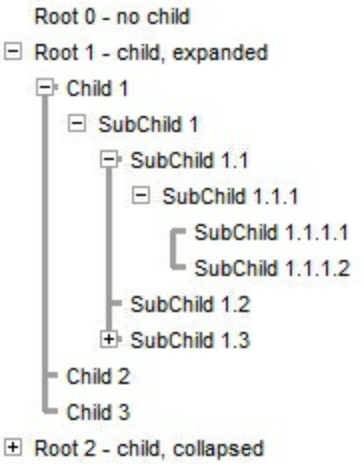
Name	Value	Description
exLevelNoLine	0	No line is shown.
exLevelDefaultLine	-1	The default line indicates a dotted line.
exLevelDotLine	1	Indicates a dotted line. For vertical/tick lines, it can be combined with exLevelLowerHalf, exLevelUpperHalf or exLevelMiddleLine. Can be combined with exLevelLowerHalf, exLevelUpperHalf or exLevelMiddleLine option.
exLevelSolidLine	2	Indicates a solid line. For vertical/tick lines, it can be combined with exLevelLowerHalf, exLevelUpperHalf or exLevelMiddleLine. Can be combined with exLevelLowerHalf, exLevelUpperHalf or exLevelMiddleLine option.
exLevelLowerHalf	16	Indicates that the line is shown in the lower half of the level. For vertical/tick lines, it can be combined with exLevelDotLine or exLevelSolidLine
exLevelUpperHalf	32	Indicates that the line is shown in the upper half of the level. For vertical/tick lines, it can be combined with exLevelDotLine or exLevelSolidLine
exLevelMiddleLine	64	Indicates that the line is shown in the middle. For vertical/tick lines, it can be combined with exLevelDotLine or exLevelSolidLine
exLevelQuarterHeight	256	Indicates that the line is shown as a quarter of the full height. Specify the exLevelQuarterHeight option to show shorter tick lines in the chart's level. Can be combined with exLevelLowerHalf, exLevelUpperHalf or exLevelMiddleLine option.

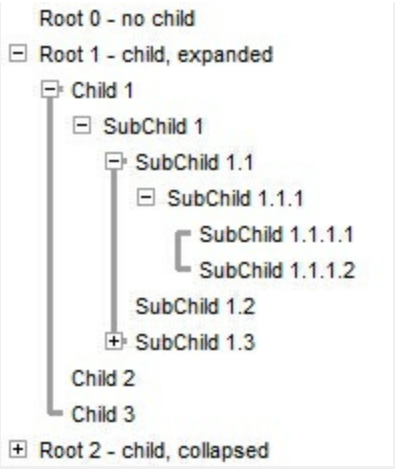
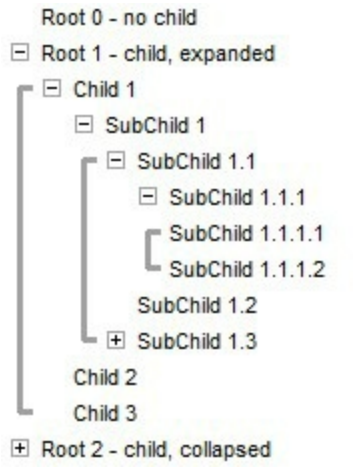
constants LinesAtRootEnum

Defines how the control displays the lines at root. The LinesAtRoot property defines the way the tree lines are shown. The HasLines property defines the type of the line to be shown. The HasButtons property defines the expand/collapse buttons for parent items.

The LinesAtRootEnum type support the following values:

Name	Value	Description
exNoLinesAtRoot	0	No lines at root items. 
exLinesAtRoot	-1	The control links the root items. 
		The control shows no links between roots, and divides them as being in the same group.

exGroupLinesAtRoot	1	
exGroupLines	2	The lines between root items are no shown, and the links show the items being included in the group. 
exGroupLinesInside	3	The lines between root items are no shown, and the links are shown between child only. 
		The lines between root items are no shown, and the links are shown for first and last visible child item.

exGroupLinesInsideLeaf	4	
exGroupLinesOutside	5	The lines between root items are no shown, and the links are shown for first and last visible child item. A parent item that contains flat child items only, does not indent the child part. By a flat child we mean an item that does not contain any child item. 

constants LinkPropertyEnum

Use the Link property to access a specified link. The AddLink method can be used to add links programmatically. The AllowLinkBars property indicates whether the user can link bars at runtime. The AllowLink event notifies your application when the user creates a link at runtime. The AddLink event notifies your application once the user adds a link between two bars. The ShowLinksColor property specifies the color for links that starts or ends on selected bars. The ShowLinksStyle property specifies the width to show the links when the link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. The SelBarColor property specifies the color to display the selected bars.

The /NET Assembly version defines get/set shortcut properties as follow (they start with get_ or set_ keywords):

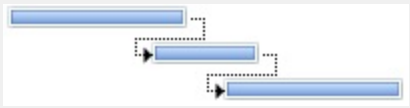
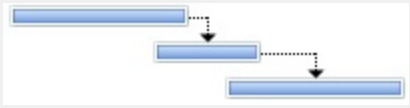
- **LinkStartItem** : Integer, retrieves or sets a value that indicates the handle of the item where the link start
- **LinkStartBar** : Object, retrieves or sets a value that indicates the key of the bar where the link starts
- **LinkEndItem** : Integer, retrieves or sets a value that indicates the handle of the item where the link ends
- **LinkEndBar** : Object, retrieves or sets a value that indicates the key of the bar where the link ends
- **LinkVisible** : Boolean, specifies whether the link is visible or hidden
- **LinkUserData** : Object, specifies an extra data associated with the link
- **LinkStartPos** : AlignmentEnum, specifies the position where the link starts in the source item
- **LinkEndPos** : AlignmentEnum, specifies the position where the link ends in the target item
- **LinkColor** : Color, specifies the color to paint the link
- **LinkArrowColor** : Color, specifies the color to paint the arrow of the link
- **LinkArrowColor32** : Color, specifies the color to paint the arrow of the link
- **LinkStyle** : LinkStyleEnum, specifies the style to paint the link
- **LinkWidth** : Integer, specifies the width in pixels of the link
- **LinkShowDir** : Boolean, specifies whether the link shows the direction
- **LinkShowRound** : Boolean, specifies whether the link is round or rectangular
- **LinkText** : String, specifies the HTML text being displayed on the link
- **LinkToolTip** : String, specifies the HTML text being shown when the cursor hovers the link
- **LinkSelected** : Boolean, specifies whether the link is selected or unselected
- **LinkGroupBars** : GroupBarsOptionsEnum, groups or ungroups the bars being linked with the specified options
- **LinkKey** : renames the link's key
- **LinkType** : defines the link's type as SF, FS, FF or SS

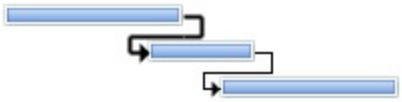
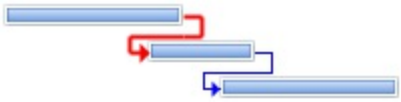
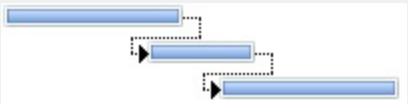
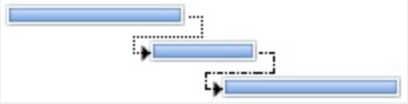
- **LinksCount** : Integer, specifies the number of the links within the chart

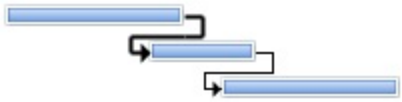
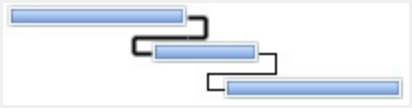
The link between two bars supports the following properties:

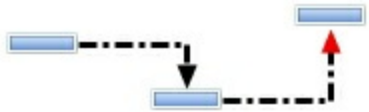
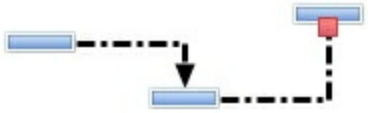
Name	Value	Description
exLinkStartItem	0	Retrieves or sets a value that indicates the handle of the item where the link start. A HITEM expression (long), that indicates the handle of the item where the link starts. *(Long/HITEM expression)*
exLinkStartBar	1	Retrieves or sets a value that indicates the key of the bar where the link starts. A String expression that indicates the key of the bar where the link starts. *(Variant expression)*
exLinkEndItem	2	Retrieves or sets a value that indicates the handle of the item where the link ends. A HITEM expression (long), that indicates the handle of the item where the link ends. *(Long/HITEM expression)*
exLinkEndBar	3	Retrieves or sets a value that indicates the key of the bar where the link ends. A String expression that indicates the key of the bar where the link ends. *(Variant expression)*
exLinkVisible	4	By default, the exLinkVisible property is True. Specifies whether the link is visible or hidden. A Boolean expression that indicates whether the link is visible or hidden. Use the ShowLinks property to hide all links in the control. *(Boolean expression)*
		Specifies an extra data associated with the link.

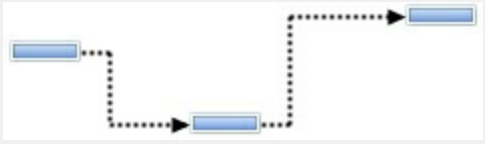
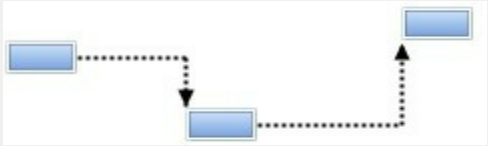
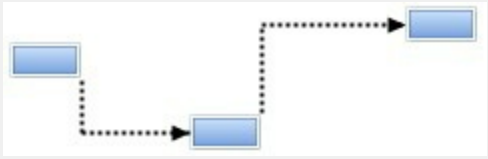
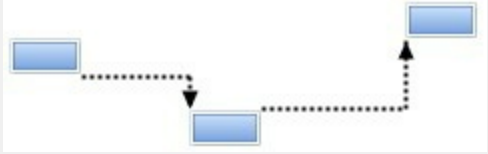
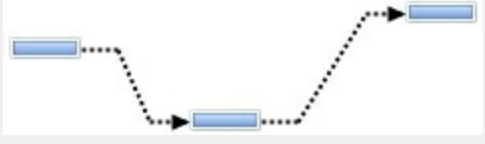
exLinkUserData	5	Use the exLinkUserData option to associate an extra data to your link. *(Variant expression)*
exLinkStartPos	6	By default, the exLinkStartPos property is 2(RightAlignment). Specifies the position where the link starts in the source item. An AlignmentEnum expression that indicates the position where the link starts. The exLinkType property defines the link's type as SF, FS(default), FF or SS. The exLinkShowRound property specifies whether the link is shown as round, rectangular, direct or straight. Links start from right and end on the left part of the bar:  Links start from center and end on right part of the bar:  A link between two bars is: • SF (Start-Finish), if the exLinkStartPos is 0(Left) and exLinkEndPos is 2(Right) • FS (Finish-Start), if the exLinkStartPos is 2(Right) and exLinkEndPos is 0(Left) (default) • FF (Finish-Finish), if the exLinkStartPos is 2(Right) and exLinkEndPos is 2(Right) • SS (Start-Start), if the exLinkStartPos is 0(Left) and exLinkEndPos is 0(Left) The SchedulePDM method arranges the activities on the plan based on the links / relationships / dependencies. *(AlignmentEnum expression)*

exLinkEndPos	7	By default, the exLinkEndPos property is 0 (LeftAlignment). Specifies the position where the link ends in the target item. An AlignmentEnum expression that indicates the position where the link ends. The exLinkType property defines the link's type as SF, FS(default), FF or SS. The exLinkShowRound property specifies whether the link is shown as round, rectangular, direct or straight. Links start on right and end on the left part of the bar(default):  Links start on right and end on the center part of the bar:  A link between two bars is: • SF (Start-Finish), if the exLinkStartPos is 0(Left) and exLinkEndPos is 2(Right) • FS (Finish-Start), by default, if the exLinkStartPos is 2(Right) and exLinkEndPos is 0(Left) • FF (Finish-Finish), if the exLinkStartPos is 2(Right) and exLinkEndPos is 2(Right) • SS (Start-Start), if the exLinkStartPos is 0(Left) and exLinkEndPos is 0(Left) The SchedulePDM method arranges the activities on the plan based on the links / relationships / dependencies. *(AlignmentEnum expression)*
		By default, the exLinkColor property is -1 (0xFFFFFFFF). Specifies the color to paint the link.If the exLinkColor property is -1, the control uses the LinksColor property to show the link. If the

exLinkColor	8	exLinkColor property is not -1, it indicates the color to draw the link. Use the exLinkArrowColor property to specify a different color to show the link's arrow. The ShowLinksColor property specifies the color to show the links when a bar is being selected. Links show same color:  Links show different colors:  *(Long/Color expression)*
exLinkStyle	9	By default, the exLinkStyle property is -1. Specifies the style to paint the link. A LinkStyleEnum expression that indicates the style of the link between two bars. If the exLinkStyle property is -1, the LinksStyle property specifies the style of the link. The ShowLinksStyle property specifies the width to show the links when the link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. Links show default style:  Links show different styles:  *(LinkStyleEnum expression)*
		By default, the exLinkWidth property is -1. Specifies the width in pixels of the link. A long expression that indicates the width of the pen, in pixels, to draw the

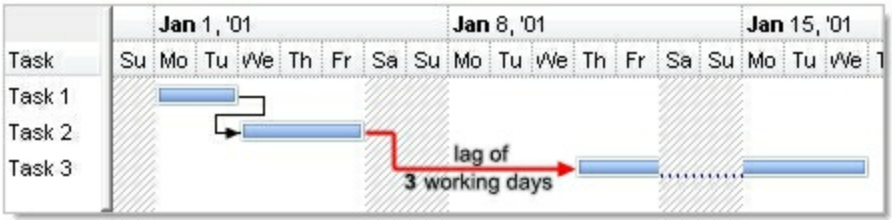
exLinkWidth	10	link between two bars. If the exLinkWidth property is -1, the LinksWidth property indicates the width of the link. The ShowLinksWidth property specifies the width to show the links when the link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. Links show with different widths:  *(Long/Color expression)*
exLinkShowDir	11	By default, the exLinkShowDir property is True. Specifies whether the link shows the direction. A Boolean expression that indicates whether the arrow in the link that specifies the direction, is visible or hidden. Links show no direction (no arrow, False):  *(Boolean expression)*
exLinkText	12	By default, the exLinkText property is empty, and so the link displays no text or picture. Specifies the HTML text being displayed on the link. Use the <img> tag to display an icon or a custom size picture on the link. Use the HTMLPicture property to include custom size picture to HTML captions. The link shows a caption on: (<img>word</img> <bgcolor=FFFFFF><a>Link</bgcolor>):  *(String expression)*
		By default, the exLinkToolTip property is empty, and

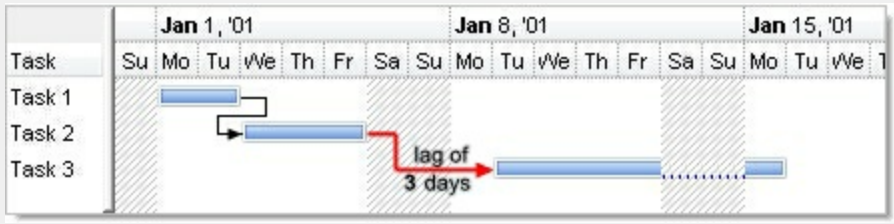
exLinkToolTip	13	so nothing is shown when cursor is hovering the link. Specifies the HTML text being shown when the cursor hovers the link. Use the <font> element to specify a different font or size for the tooltip, or use the ToolTipFont property to specify a different font or size for all tooltips in the control. The Tooltip(0, -3, , , , ,) event occurs once the link's tooltip (exLinkToolTip) is about to be shown (-3 if the mouse pointer hovers the links of the chart). *(String expression)*
exLinkArrowColor	14	By default, the exLinkArrowColor is -1 (0xFFFFFFFF) which indicates that the exLinkColor property indicates the color to show the link's arrow (same color as the link itself). Specifies the color to show the link's arrow. If the exLinkArrowColor is not -1, it indicates the color to display the arrow of the link or if the the last 7 bits in the high significant byte of the color indicates the identifier of the skin being used to show the arrow. Use the Add method to add new skins to the control. The arrow or the direction of the Link is displayed with a solid color:  The arrow or the direction of the Link is displayed with an EBN color:  *(Long/Color/EBN expression)*
		By default, the exLinkShowRound property is 0, which indicates that the link is displayed rectangular. Specifies whether the link is shown as round, rectangular, direct or straight. The exLinkStartPos and exLinkEndPos defines the position/side of the bar where the link

exLinkShowRound	15	starts/ends. The exLinkType property defines the link's type as SF, FS(default), FF or SS. The exLinkShowRound property supports the following values/types: • **-1** (round)  • **0** (rectangular, starts horizontally, ends horizontally, default)  • **3** (rectangular EV, starts horizontally, ends vertically)  • **4** (rectangular SV, starts vertically, ends horizontally)  • **5** (rectangular SEV, starts vertically, ends vertically)  • **1** (direct) 

- **2** (straight)

(Long expression, valid values are -1, 0, 1, 2, 3, 4 and 5)

exLinkPDMWorkingDelay	16	By default, the exLinkPDMWorkingDelay is 0. Specifies the working delay for the activity during PDM scheduling. This property specifies the number of working days between two linked bars. The property keeps count on the non-working area of the chart. The SchedulePDM method uses the exLinkPDMWorkingDelay property when arranging bars, if it is not zero. For instance, if the bar A links to bar B using a FS (Finish-Start type), and the exLinkPDMWorkingDelay property is 2 (working days), it means the bar B starts 2 working days after activity A ends. Use the exLinkPDMDelay proeprty to specify the LAG in days, rather than working days. Only one of these 2 properties have effect at once. The first non-zero value in order of exLinkPDMWorkingDelay, exLinkPDMDelay is used by SchedulePDM method. For instance, if both properties are set to 2, the SchedulePDM takes/considers exLinkPDMWorkingDelay as primary, and ignores the other. The following screen shot shows a LAG of 3 working days:  *(Double expression)*
		By default, the exLinkPDMDelay is 0. Specifies the delay for the activity during PDM scheduling. Specifies the delay for the activity during PDM

exLinkPDMDelay	17	scheduling. This property specifies the number of days between two linked bars. The property does not keeps count on the non-working area of the chart. The SchedulePDM method uses the exLinkPDMDelay property when arranging bars, if it is not zero. For instance, if the bar A links to bar B using a FS (Finish-Start type), and the exLinkPDMWorkingDelay property is 2 (days), it means the bar B starts 2 days after activity A ends. The first non-zero value in order of exLinkPDMWorkingDelay, exLinkPDMDelay is used by SchedulePDM method. For instance, if both properties are set to 2, the SchedulePDM takes/considers exLinkPDMWorkingDelay as primary, and ignores the other. The following screen shot shows a LAG of 3 days:  *(Double expression)*
exLinkSelected	257	By default, the exLinkSelected is False. Specifies whether the link is selected or unselected.The AllowSelectObjects property allows users to select at runtime the bars and links in the chart area. The ChartSelectionChanged event is fired when the selection in the chart is changed. The ShowLinksColor property specifies the color to show the links when a bar is being selected. The ShowLinksWidth property specifies the width to show the links when the link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. The ShowLinksStyle property specifies the width to show the links when the link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. *(Boolean expression)*

exLinkGroupBars	258	Groups or ungroup the bars being linked with the specified options . For instance. this option is equivalent with grouping the end of starting bar with the start of the ending bar of the link. For instance, the *.Link(LinkKey, exLinkGroupBars) = GroupBarsOptionsEnum.exPreserveBarLength + GroupBarsOptionsEnum.exFlexibleInterval + GroupBarsOptionsEnum.exIgnoreOriginalInterval* is equivalent with *.GroupBars .Link(LinkKey, exLinkStartItem), .Link(LinkKey, exLinkStartBar), False, .Link(LinkKey, exLinkEndItem), .Link(LinkKey, exLinkEndBar), True, GroupBarsOptionsEnum.exPreserveBarLength + GroupBarsOptionsEnum.exFlexibleInterval + GroupBarsOptionsEnum.exIgnoreOriginalInterval.* The GroupBars method groups two bars. If calling the set property, the value of the exLinkGroupBars option can be a a long expression that specifies a combination of GroupBarsOptionsEnum type, or a string expression in format groupbarsoptions;options, where the first argument indicates the value of GroupBarsOptionsEnum type, since the rest of asguments, are passed to Options parameter of the GroupBars method to specify a fixed interval, a minimum interval value and so on. The AddLink event notifies your application once the user adds a link between two bars. The following VB sample groups the bars being linked: `Private Sub G2antt1_AddLink(ByVal LinkKey As String)` `  With G2antt1.Items` `  .Link(LinkKey, exLinkGroupBars) = GroupBarsOptionsEnum.exFlexibleInterval Or GroupBarsOptionsEnum.exPreserveBarLength Or GroupBarsOptionsEnum.exIgnoreOriginalInterval` `  End With` `End Sub`

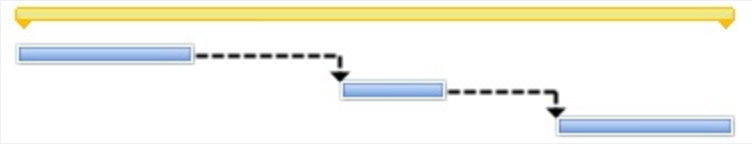
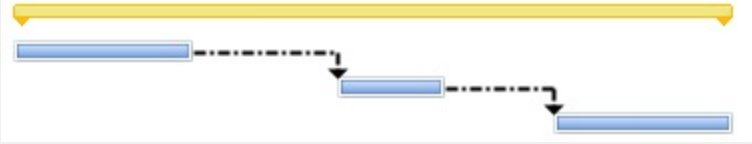
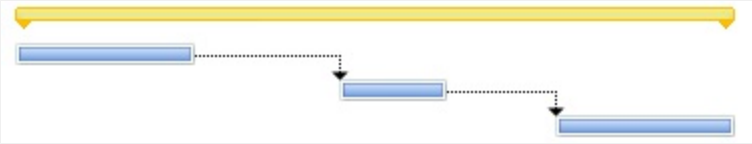
		The following C# sample groups the bars being linked: `private void exg2antt1_AddLink(object sender, string LinkKey)` `{` ` exg2antt1.Items.set_LinkGroupBars(LinkKey, exontrol.EXG2ANTTLib.GroupBarsOptionsEnum.exF	exontrol.EXG2ANTTLib.GroupBarsOptionsEnum.exl	exontrol.EXG2ANTTLib.GroupBarsOptionsEnum.exF` `}` Once the user moves a grouped bar, the relative bar is moved resized accordingly with the grouping options. *([GroupBarsOptionsEnum](#) expression)*
exLinkKey	259	Changes the key of the giving link. The get function retrieves the link's key if the link is found, else it returns an empty string. You can use the get_LinkKey function to check if a specified link is found or not. The set function may be used to rename the key of the link, when [AddLink](#) event is fired. The key of the link may be changed if the new key is available, in other words, if there is no other link with the new key. *(String expression)*
		By default the exLinkType property is **2** (FS or Finish-Start). The exLinkStartPos / exLinkEndPos defines the position/side of the bar the link starts or ends. The exLinkType property depends on exLinkStartPos / exLinkEndPos properties, to

exLinkType	260	define the link's type as one of the following values: • **1** (Start to Finish (SF), the exLinkStartPos is 0(Left) and exLinkEndPos is 2(Right)) • **2** (Finish to Start (FS), the exLinkStartPos is 2(Right) and exLinkEndPos is 0(Left)) • **4** (Finish to Finish (FF), the exLinkStartPos is 2(Right) and exLinkEndPos is 2(Right)) • **8** (Start to Start (SS), the exLinkStartPos is 0(Left) and exLinkEndPos is 0(Left)) Tasks may have multiple predecessors or multiple successors. Before you begin establishing dependencies, its important to understand that there are four types: • Finish to Start (FS), the predecessor ends before the successor can begin • Start to Start (SS), the predecessor begins before the successor can begin • Finish to Finish (FF), the predecessor ends before the successor can end • Start to Finish (SF), the predecessor begins before the successor can end The SchedulePDM method arranges the activities on the plan based on the links / relationships / dependencies. The Link(exLinkType) = *value* changes the link's type to *value*, where *value* can be any of the following values: • **1** or **"SF"** to define a Start to Finish (SF) link, changes the exLinkStartPos to 0(Left) and exLinkEndPos to 2(Right) • **2** or **"FS"** to define a Finish to Start (FS) link, changes the exLinkStartPos to 2(Right) and exLinkEndPos to 0(Left) • **4** or "**FF"** to define a Finish to Finish (FF) link, changes the exLinkStartPos to 2(Right) and exLinkEndPos to 2(Right) • **8** or **"SS"** to define a Start to Start (SS) link, changes the exLinkStartPos to 0(Left) and

		exLinkEndPos to 0(Left) *(Long expression)*
exLinksCount	512	Specifies the number of the links within the chart. This property requires no key to be invoked, so it counts the number of links in your chart. Use the FirstLink and NextLink properties to enumerate the links in the control. For instance, Items.Link(Nothing,exLinksCount) gets the number of links in the chart. *(Long expression)*

constants LinkStyleEnum

Use the LinksStyle property to specify the style of the pen to draw all links in the chart. Use the Link(exLinkStyle) property to change the style for a specific link. Use the Link(exLinkShowRound) property to show round links. Use the AntiAliasing property to specify anti-aliasing rendering to show the links within the chart. The link can have one of the following styles:

Name	Value	Description
exLinkSolid	0	 The link is solid.
exLinkDash	1	 The link is dashed.
exLinkDot	2	 The link is dotted.
exLinkDashDot	3	 The link has alternating dashes and dots.
exLinkDashDotDot	4	 The link has alternating dashes and double dots.
exLinkTDot	255	 Default. The link is dotted. This style is valid only when the exLinkWidth is 1.

constants NoteLinkTypeEnum

The NoteLinkTypeEnum expression specifies whether the link between the parts of the note is visible or hidden, whether the link is shown the direction between parts. The ShowLink property specifies whether the note shows or hides the link between parts of the notes. Use the PartVisible property to specify whether the start or ending part of the note is visible or hidden. The LinkStyle property determines the style of the link between parts of the note. The LinkColor property specifies the color of the link between parts of the notes, while the LinkWidth property determines the width of the link between parts of the notes. The link between parts of the note is shown if the ShowLink property includes the exNoteLinkVisible flag, LinkWidth property is greater than 0, the start and end part of the note do not intersect.

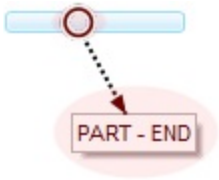
The following screen shows the link between starting and ending parts of the note:

Name	Value	Description
exNoteLinkHidden	0	The link between parts of the note is not shown.
exNoteLinkVisible	1	The link between parts of the note is shown.
exNoteLinkShowDirStartToEnd	2	The link between parts of the note shows the direction from start to end.
exNoteLinkShowDirEndToStart	4	The link between parts of the note shows the direction from end to start.
exNoteLinkStartToEnd	3	The link shows its direction and goes from start to end.
exNoteLinkEndToStart	5	The link shows its direction and goes from end to start.

constants NotePartEnum

The NotePartEnum expression determines the part of note being accessed, by properties prefixed with *Part*, such as: PartCanMove, PartText, and so on.

The following screen shot shows (in red) the starting and ending part of a note:



The position of the starting part is relative to the associated object (DATE or BAR), while the ending part is relative to the starting part. The NotePartEnum type contains the following values:

Name	Value	Description
exNoteStart	0	Indicates the starting part of the note.
exNoteEnd	1	Indicates the ending part of the note.

constants NotesClipToEnum

The NotesClipToEnum type specifies how the chart's notes are clipped. The ClipTo property specifies the chart's notes limits. The NotesClipToEnum type supports the following values:

Name	Value	Description
exNotesClipNone	0	(By default) Indicates that the notes are shown on the chart with no clipping.
exNotesClipToList	1	Notes are clipped to list portion of the chart.
exNotesClipToItems	2	Notes are clipped to items portion of the chart.

constants InplaceAppearanceEnum

Defines the editor's appearance. Use the Appearance property to change the editor's appearance. Use the PopupAppearance property to define the appearance of the editor's drop-down window, if it exists.

Name	Value	Description
NoApp	0	No border
FlatApp	1	Flat appearance
SunkenApp	2	Sunken appearance
RaisedApp	3	Raised appearance
EtchedApp	4	Etched appearance
BumpApp	5	Bump appearance
ShadowApp	6	Shadow appearance
InsetApp	7	Inset appearance
SingleApp	8	Single appearance

constants NumericEnum

Use the Numeric property to specify the format of numbers when editing a field.

Name	Value	Description
exInteger	-1	Allows editing numbers of integer type. The format of the integer number is: **[+/-]digit**, where **digit** is any combination of digit characters. This flag can be combined with exDisablePlus, exDisableMinus or exDisableSigns flags. For instance, the 0x3FF (hexa representation, 1023 decimal) value indicates an integer value with no +/- signs.
exAllChars	0	Allows all characters. No filtering.
exFloat	1	Allows editing floating point numbers. The format of the floating point number is: **[+/-]digit[.digit[[e/E/d/D][+/-]digit]]**, where digit is any combination of **digit** characters. Use the exEditDecimalSymbol option to assign a new symbol for '**.**' character (decimal values). This flag can be combined with exDisablePlus, exDisableMinus or exDisableSigns flags.
exFloatInteger	2	Allows editing floating point numbers without exponent characters such as e/E/d/D, so the accepted format is **[+/-]digit[.digit]**. Use the exEditDecimalSymbol option to assign a new symbol for '**.**' character (decimal values). This flag can be combined with exDisablePlus, exDisableMinus or exDisableSigns flags.
exDisablePlus	256	Prevents using the + sign when editing numbers. If this flag is included, the user can not add any + sign in front of the number.
exDisableMinus	512	Prevents using the - sign when editing numbers. If this flag is included, the user can not add any - sign in front of the number.
exDisableSigns	768	Prevents using the +/- signs when editing numbers. If this flag is included, the user can not add any +/- sign in front of the number. For instance exFloatInteger + exDisableSigns allows editing floating points numbers without using the exponent and plus/minus characters, so the allowed format is

digit[.digit]

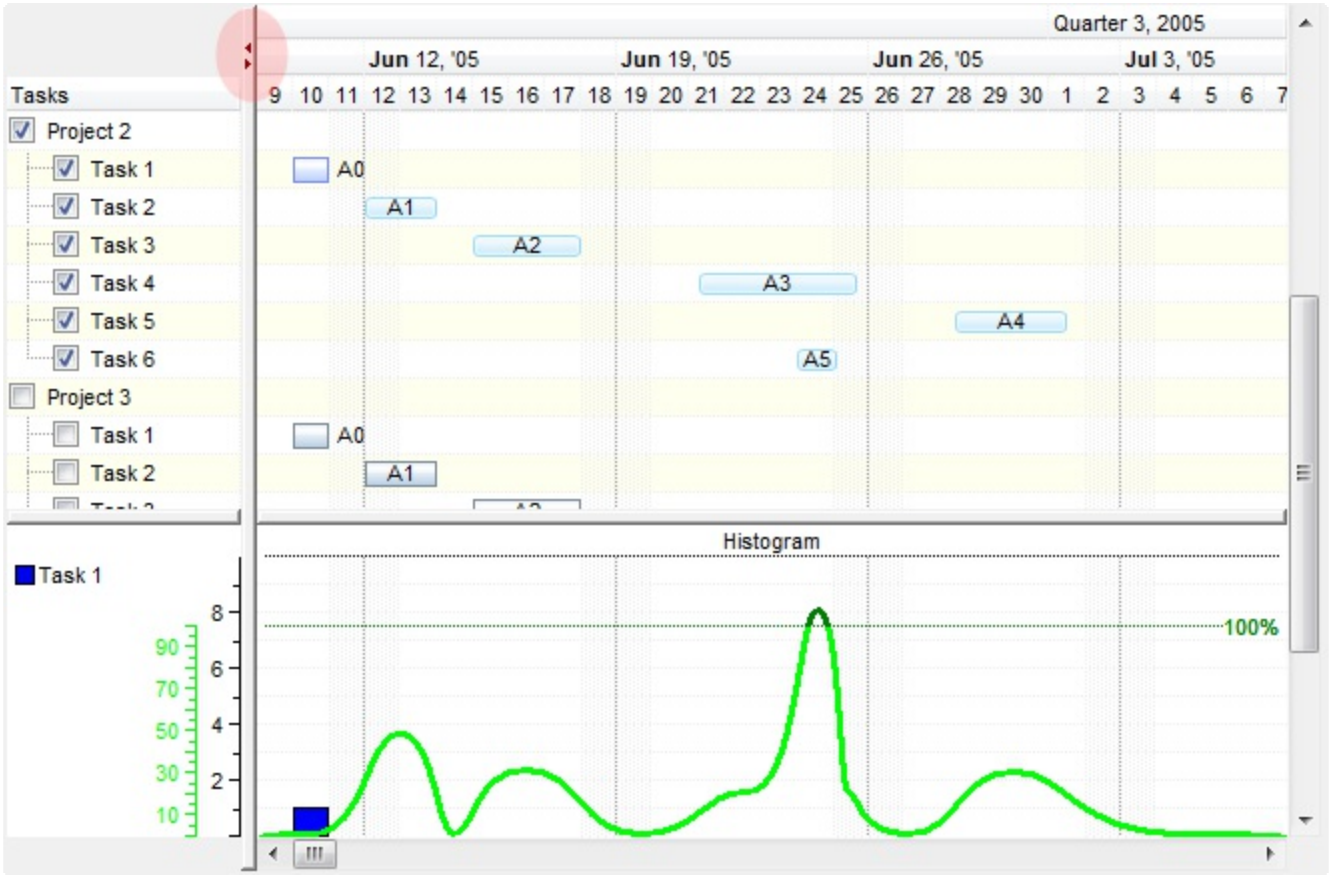
constants OnResizeControlEnum

The OnResizeControlEnum type specifies the parts of the controls being resized when the control itself gets resized. Use the OnControlResize property to specify which part list or chart of the control is getting resized once the control itself is resized. For instance, the OnControlResize property may be (exResizeChart or exDisableSplitter) that specifies that the control resizes the chart area, and the vertical splitter is disabled.

Name	Value	Description
exResizeList	0	Resizes the list part of the control. The control fires the ChartStartChanging(exVSplitterChange) and ChartEndChanging(exVSplitterChange) event when the list or chart area gets resized. The PaneWidth property specifies the width of the list/item/chart area.
exResizeChart	1	Resizes the chart part of the control. The control fires the ChartStartChanging(exHSplitterChange) and ChartEndChanging(exHSplitterChange) event when the list or chart area gets resized. The PaneWidth property specifies the width of the list/item/chart area. The controls vertical splitter is hidden if the OnControlResize property is exResizeChart + exDisableSplitter (129) and the PaneWidth(False) property is 0.
exDisableSplitter	128	Disables the splitter. If this option is set the user can not resize the chart or the list using the control's splitter, at runtime. The disabled cursor is shown when the cursor-mouse hovers the vertical splitter. The controls vertical splitter is hidden if the OnControlResize property is exResizeChart + exDisableSplitter (129) and the PaneWidth(False) property is 0.
exDisableHistogram	256	Disables resizing the histogram at runtime. If this is set the user can't resize the histogram at runtime. Use the HistogramVisible property to show or hide the chart's histogram. The HistogramHeight property specifies the height in pixels of the chart's histogram. The control fires the ChartStartChanging(exHSplitterChange) and ChartEndChanging(exHSplitterChange) event when the histogram's bound is changed. The disabled cursor is shown when the cursor-mouse hovers the

		horizontal splitter.
exSplitterShowButtons	512	Shows the resize buttons on the vertical splitter.
exDisableOverview	1024	Disables resizing the overview. If this flag is present, the user is not able to resize the control's overview by dragging the horizontal split bar.
exDisableSplitPane	2048	Disables the splitting the pane. If this flag is present, the user is not able to resize any split panels by dragging the vertical split bar. This flag has effect, only if the AllowSplitPane property is set.

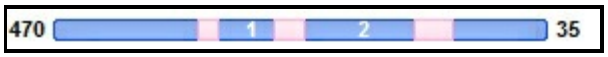
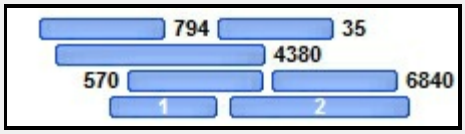
The following screen shot shows the resizing buttons when the exSplitterShowButtons is set:



constants OverlaidBarsTypeEnum

The OverlaidBarsTypeEnum type specifies the type of the overlay bar supported. *The OverlaidType property specifies how two or multiple bars inside the item covers each other.*

Name	Value	Description
exOverlaidBarsNone	0	No overlaid bars are shown (default).
exOverlaidBarsOffset	1	The overlaid bars are shown using a different vertical offset . *The Overlaid(exOverlaidBarsOffset) specifies the vertical offset, in pixels, to display the overlaid bars. The Overlaid(exOverlaidBarsTransparent) specifies the percent of transparency being applied to bars in the same item that are not moved or resized.* By default, the Overlaid(exOverlaidBarsOffset) property is 3 pixels. The ItemBar(exBarOffset) property specifies the vertical offset to display the bar. The exOverlaidBarsOffset flag can be combined with exOverlaidBarsTransparent or exOverlaidBarsIncludeCaption flag. *This option does NOT change the height of the item.*  If three bars get intersected each other, the first is shown on the top, the second on the middle, and the third on the bottom. Click here to watch a movie on how the exOverlaidBarsOffset mechanism works.
		The overlaid portion is shown using a different type of bar. *The Overlaid(exOverlaidBarsIntersect) specifies the name of the bar to be displayed on the portion that laid over bars.* By default, the Overlaid(exOverlaidBarsIntersect) property is empty, so nothing is displayed if the Overlaid is exOverlaidBarsIntersect. You MUST specify the

exOverlaidBarsIntersect	2	name of the task to display the portion that covers the bars if the Overlaid is exOverlaidBarsIntersect. The exOverlaidBarsIntersect flag can be combined with exOverlaidBarsTransparent flag. *This option does NOT change the height of the item.*  The intersection between bars is shown with a different color, pattern, shape or EBN object. Click here to watch a movie on how the exOverlaidBarsIntersect mechanism works.
exOverlaidBarsStack	3	The bars that covers each other are shown as a stack. *This option changes the height of the item so the bars that covers each other are displayed entirely. The Overlaid(*exOverlaidBarsStack*) specifies the distance in pixels between two bars that covers each other. The Overlaid(exOverlaidBarsTransparent) specifies the percent of transparency being applied to bars in the same item that are not moved or resized.* The exOverlaidBarsStack flag can be combined with exOverlaidBarsStackAutoArrange, exOverlaidBarsTransparent or exOverlaidBarsIncludeCaption flag. The ItemHeight property specifies the height of the item. The ItemMaxHeight property specifies the maximum height for the item. Use the ScrollBySingleLine property to allow the entire chart to be scrollable (using items with different heigths).  The bars get arranged into a stack. If the exOverlaidBarsStackAutoArrange flag is not used, each bar is shown on a row, else the bars get automatically arranged.

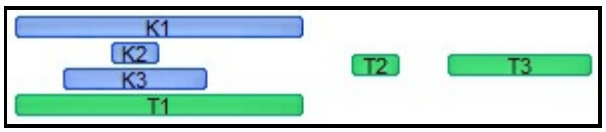
Click here to watch a movie on how the exOverlaidBarsStack + exOverlaidBarsStackAutoArrange mechanism works.

exOverlaidBarsCascade 4

The bars gets arranged as a cascade with the z-order being indicated by ItemBar(exBarOverlaidCascade) key. Arrange the bars on the same level for those with the same exBarOverlaidCascade key, and on a different level for bars with a different exBarOverlaidCascade key like in the following picture (The K1, K2, K3 are being arranged on the level A, while the T1, T2, and T3 on level B). *This option changes the height of the item so the bars that covers each other are displayed entirely.*

The difference between cascade and stack is that if two bars with the same exBarOverlaidCascade key are shown in the same level if they do not intersect, and two bars with a different exBarOverlaidCascade key shows in different levels like shown in the following screen shot.

The bars get arranged into a cascade based on the key (exOverlaidBarsCascade). The T1, T2, T3 are shown on the same level, as they have the same exBarOverlaidCascade key, and does not intersect the K1, K2, K3 level.



The bars get arranged into a stack (exOverlaidBarsStack).

Click here to watch a movie on how the

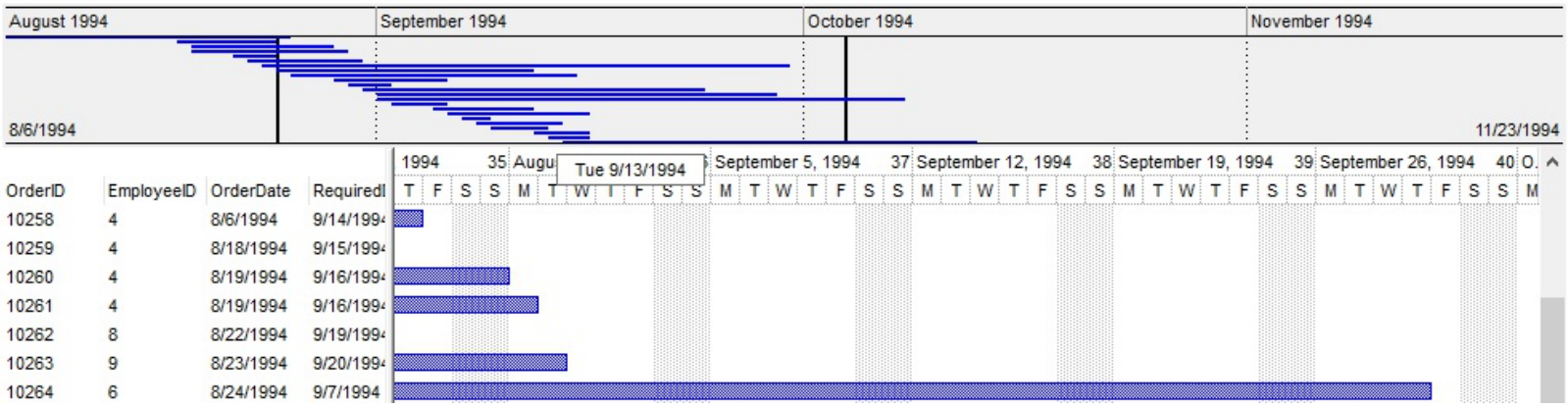
		exOverlaidBarsCascade mechanism works.
exOverlaidBarsTransparent	256	The overlaid portion is shown using a semi-transparent color. The Overlaid(exOverlaidBarsTransparent) specifies the percent of transparency being applied to the covered bar. By default, the Overlaid(exOverlaidBarsTransparent) property is 50 (semi-transparent). The ItemBar(exBarTransparent) property specifies the vertical offset to display the bar. The exOverlaidBarsTransparent flag maybe combined with exOverlaidBarsOffset, exOverlaidBarsIntersect or exOverlaidBarsStack flag.
exOverlaidBarsStackAutoArrange	512	The overlaid stack is automatically arranged for best fit in the item. The exOverlaidBarsStackAutoArrange flag can be combined with exOverlaidBarsStack flag only.
exOverlaidBarsIncludeCaption	4096	The overlaid mechanism includes the bar's caption. Use the ItemBar(exBarCaption) property to specify a caption being displayed with the bar. Use the ItemBar(exBarHAlignCaption) property to specify the alignment of the bar's caption or to specify whether the caption is displayed inside or outside of the bar. For instance, you can use the exOverlaidBarsIncludeCaption flag to specify whether the caption of the bar does not cover with other bars. The exOverlaidBarsIncludeCaption flag can be combined with exOverlaidBarsOffset or exOverlaidBarsStack flags.
exOverlaidBarsStrict	8192	The overlaid mechanism includes only bars in the same group (OverlaidGroup property) but of different types. This flag can be used in combination with: exOverlaidBarsOffset, exOverlaidBarsIntersect, exOverlaidBarsStack or exOverlaidBarsCascade. For instance, you can use this option to stack or cascade a specified type of bar when it get intersected with other type of bars. Click here to watch a movie on how the

exOverlaidBarsCascade + exOverlaidBarsStrict works.

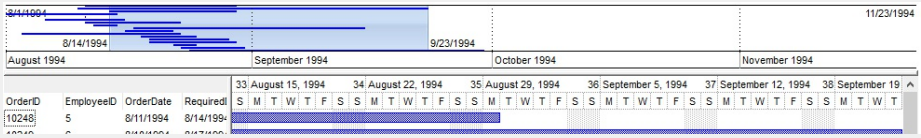
constants OverviewVisibleEnum

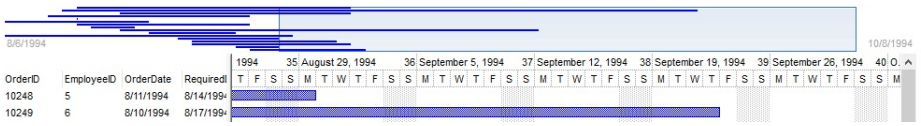
The OverviewVisibleEnum type specifies the way items are represented in the overview area. Use the OverviewVisible property to specify whether the control's overview visible is hidden or shown.

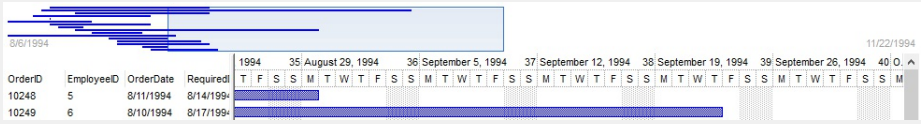
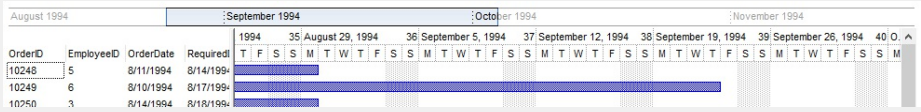
The following screen shot the control's overview part:

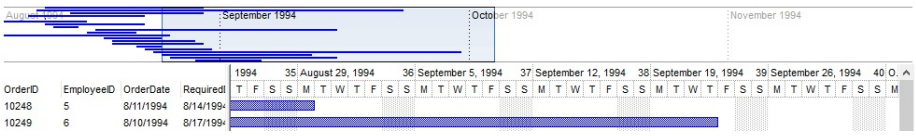
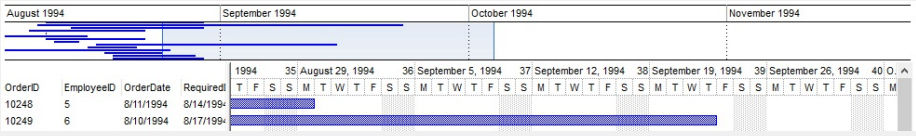


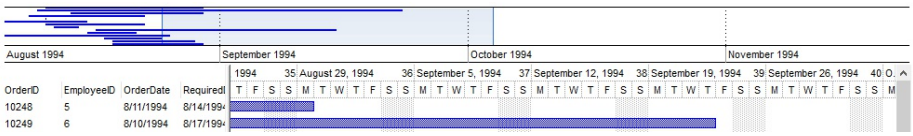
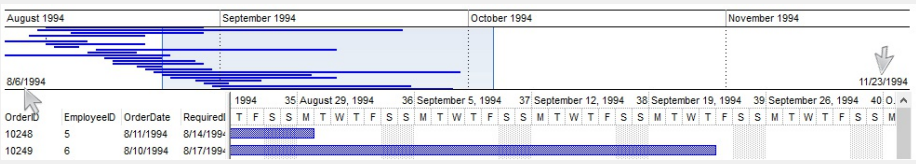
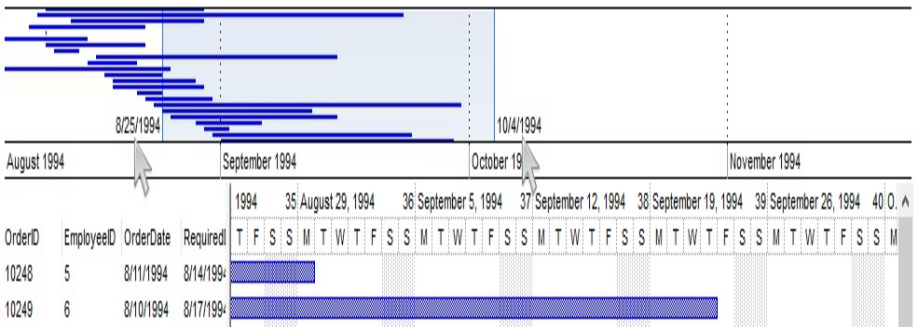
The OverviewVisibleEnum type includes the following values:

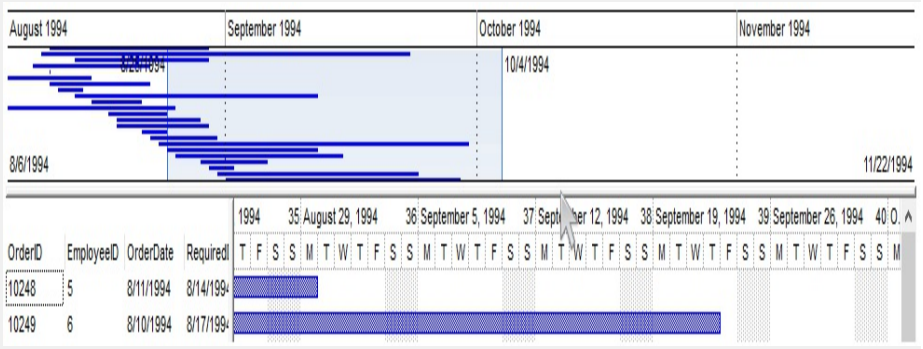
Name	Value	Description
exOverviewHidden	0	The control's overview is not visible.
exOverviewShowAll	-1	The control's overview shows the bars from the visible items using the range for all bars in the chart. The exOverviewShowAll ignores the exOverviewHideBars. This flag should not be used with newer versions, it is provided for backward compatibility. The following screen shot shows the control's overview when exOverviewShowAll flag is specified only: 
exOverviewShowOnlyVisible	1	The control's overview shows the bars from the visible items using the range of bars in the visible items only. The following screen shot shows the control's overview when exOverviewShowOnlyVisible flag is specified only:



exOverviewShowAllVisible	2	The control's overview shows the bars from the visible items using the range for all bars in the chart. The following screen shot shows the control's overview when exOverviewShowAllVisible flag is specified only: 
exOverviewAllowVerticalScroll	256	Indicates whether the user can vertically scroll the chart while navigating up or down the overview part of the control. For instance, you can click the overview panel, the chart displays the selected area, and you can drag the cursor left or right to select a new date-time range to be displayed, or you can go up or down, to scroll items up or down.
exOverviewHideBars	512	Prevents showing the bars in the overview part of the control. For instance, you can use this flag in combination of any other flag to show just the time-scale in the overview part of the control, to allow the user to quickly scroll the chart's content to a specific time-zone. The exOverviewHideBars is ignored if the OverviewVisible property is exOverviewShowAll. The following screen shot shows the control's overview when exOverviewHideBars flag is specified: 
		Specifies whether the overview part of the control displays the date-time scale. This flag includes the time-scale on the overview. The time-scale intersects the bars in the overview. By default, the

exOverviewShowDateTimeScale	4096	time-scale of the overview part is shown on the top of it, so you can combine the exOverviewShowDateTimeScale flag with exOverviewShowDateTimeScaleBottom flag, to display the time-scale on the bottom side of the overview part of the control. The following screen shot shows the control's overview when exOverviewShowDateTimeScale flag is specified: 
exOverviewShowDateTimeScaleSplit	8192	Specifies whether the overview's date-time scale is displayed into a separate portion of the overview. This flag includes the time-scale on the overview. The time-scale does not intersect the bars in the overview. By default, the time-scale of the overview part is shown on the top of it, so you can combine the exOverviewShowDateTimeScaleSplit flag with exOverviewShowDateTimeScaleBottom flag, to display the time-scale on the bottom side of the overview part of the control. The following screen shot shows the control's overview when exOverviewShowDateTimeScaleSplit flag is specified: 
		Specifies whether the overview's date-time scale is displayed on the bottom side of the overview. By default, the time-scale of the overview part is shown on the top of it, so you can use the exOverviewShowDateTimeScaleBottom flag with exOverviewShowDateTimeScale or exOverviewShowDateTimeScaleSplit to display the time-scale on the bottom side of the overview part of the control.

exOverviewShowDateTimeScaleBottom	4134	The following screen shot shows the control's overview when exOverviewShowDateTimeScaleBottom flag is specified: 
exOverviewShowMargins	8192	Displays the limits of the overview bars. You can include the exOverviewShowMargins flag to display the margins/limits of all (project) / visible bars. In other words, the exOverviewShowMargins flag displays the minimal ItemBar(exBarStart) value, and the maximal ItemBar(exBarEnd) value. The following screen shot shows the control's overview when exOverviewShowMargins flag is specified: 
exOverviewShowSelMargins	16384	Displays the selection limits (first/last visible date in the chart). The following screen shot shows the control's overview when exOverviewShowSelMargins flag is specified: 
		Specifies whether the overview's horizontal splitter is visible or hidden. Include the exDisableOverview in the OnResizeControl property to disable resizing

exOverviewSplitter	65536	the chart's overview part of the control. You can use the Background(exSplitBar) property to change the back color / visual appearance of the overview's split bar. The following screen show shows the splitter of the control's overview, so the exOverviewSplitter flag is set: 

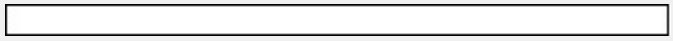
constants OverviewZoomEnum

The OverviewZoomEnum type specifies when the zooming scale is displayed. Use the AllowOverviewZoom property to specify whether the zooming zoom is shown or hidden.

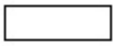
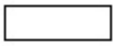
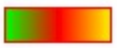
Name	Value	Description
exDisableZoom	0	Zooming the chart at runtime is disabled.
exAlwaysZoom	1	The zooming scale is displayed on the overview area.
exSelectOnRClick	2	The user selects a portion of the chart to be zoomed, if the user right clicks the overview area.
exZoomOnRClick	-1	The zooming scale is displayed only if the user right clicks the overview area.

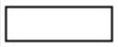
constants PatternEnum

The PatternEnum expression indicates the type of brush. Use the NonworkingDaysPattern property to specify the pattern to fill non-working days. Use the Pattern property to specify the brush to fill the bar. The HistogramPattern property defines the pattern to be shown when the bar is included in the histogram. The Color property specifies the pattern's color or an EBN object to define the skin to be applied on the bar. The Color property is applied to all bars of the same type, while the ItemBar(exBarColor) property specifies a different color/skin for a particular bar. You can use the ItemBar(exBarPattern) property to specify a different pattern for a particular bar.

Name	Value	Description
exPatternEmpty	0	The pattern/bar is not visible.
exPatternSolid	1	
exPatternDot	2	
exPatternShadow	3	
exPatternNDot	4	
exPatternFDiagonal	5	
exPatternBDiagonal	6	
exPatternDiagCross	7	
exPatternVertical	8	
exPatternHorizontal	9	
exPatternCross	10	
exPatternBrick	11	
exPatternYard	12	
		 The Color property specifies the color for the border, while the StartColor and EndColor properties defines the start and ending color to show a linear-horizontal gradient bar. The liner gradient is shown if the StartColor or EndColor is not zero, and have different values. If the StartColor and EndColor are different that zero and have the same the same value the exPatternBox bar shows solid fill with a solid border being defined by the Color property. This option can be combined with any predefined pattern, exPatternGradientVBox, exPatternGradient3Colors, exPatternThickBox or

		exPatternFrameShadow. This option can not be applied to EBN bars. The following pictures where generated if the bar's Pattern is exPatternBox
exPatternBox	32	•  StartColor and EndColor properties are not used (0). •  StartColor is RGB(0,255,0) [green] and EndColor is RGB(255,255,0) [yellow]. •  StartColor is RGB(0,255,0) [green], EndColor is RGB(255,255,0) [yellow] and Color is RGB(255,0,0) [red]. The following pictures where generated if the bar's Pattern is exPatternBox + exPatternDot •  StartColor and EndColor properties are not used (0). •  StartColor is RGB(0,255,0) [green] and EndColor is RGB(255,255,0) [yellow]. •  StartColor is RGB(0,255,0) [green], EndColor is RGB(255,255,0) [yellow] and Color is RGB(255,0,0) [red].
		The Color property specifies the color for the border, while the StartColor and EndColor properties defines the start and ending color to show a linear-vertical gradient bar. The liner gradient is shown if the StartColor or EndColor is not zero, and have different values. If the StartColor and EndColor are different that zero and have the same the same value the exPatternBox bar shows solid fill with a solid border being defined by the Color property. This option must be combined with exPatternBox, and can be combined with any predefined pattern, exPatternGradient3Colors, exPatternThickBox or exPatternFrameShadow. This option can not be applied to EBN bars. The following pictures where generated if the bar's Pattern is exPatternBox + exPatternGradientVBox

exPatternGradientVBox	64	•  StartColor and EndColor properties are not used (0). •  StartColor is RGB(0,255,0) [green] and EndColor is RGB(255,255,0) [yellow]. •  StartColor is RGB(0,255,0) [green], EndColor is RGB(255,255,0) [yellow] and Color is RGB(255,0,0) [red]. The following pictures where generated if the bar's Pattern is exPatternBox + exPatternGradientVBox + exPatternDot •  StartColor and EndColor properties are not used (0). •  StartColor is RGB(0,255,0) [green] and EndColor is RGB(255,255,0) [yellow]. •  StartColor is RGB(0,255,0) [green], EndColor is RGB(255,255,0) [yellow] and Color is RGB(255,0,0) [red].
exPatternGradient3Colors	128	This option defines the gradient from 3 colors defined by StartColor, Color and EndColor. The gradient starts with StartColor, continue to Color and ends on EndColor color. This option must be combined with exPatternBox and can be combined with any predefined pattern, exPatternGradientVBox, exPatternThickBox or exPatternFrameShadow. This option can not be applied to EBN bars. The following pictures where generated if the bar's Pattern is exPatternBox + exPatternGradient3Colors •  StartColor and EndColor properties are not used (0). •  StartColor is RGB(0,255,0) [green] and EndColor is RGB(255,255,0) [yellow]. •  StartColor is RGB(0,255,0) [green], EndColor is RGB(255,255,0) [yellow] and Color is RGB(255,0,0) [red].

		The following pictures where generated if the bar's Pattern is exPatternBox + exPatternGradientVBox + exPatternGradient3Colors •  StartColor and EndColor properties are not used (0). •  StartColor is RGB(0,255,0) [green] and EndColor is RGB(255,255,0) [yellow]. •  StartColor is RGB(0,255,0) [green], EndColor is RGB(255,255,0) [yellow] and Color is RGB(255,0,0) [red].
exPatternThickBox	4096	Use this option to specify a thicker border for bars. This option can be combined with any predefined pattern, exPatternBox, exPatternGradientVBox, exPatternGradient3Colors or exPatternFrameShadow. This option can not be applied to EBN bars too. The following pictures where generated based on the exPatternThickBox flag: •  exPatternThickBox flag is not set (Pattern = exPatternBDiagonal). •  exPatternThickBox flag is set (Pattern = exPatternBDiagonal + exPatternThickBox)
exPatternFrameShadow	8192	This option can be used to display a shadow for the bars. This option can be combined with any predefined pattern, exPatternBox, exPatternGradientVBox, exPatternGradient3Colors or exPatternThickBox. This option can be applied to EBN bars too. The following pictures where generated based on the exPatternThickBox flag: •  exPatternFrameShadow flag is not set (Pattern = exPatternShadow). •  exPatternFrameShadow flag is set (Pattern = exPatternShadow + exPatternFrameShadow)

		•  exPatternFrameShadow flag is set (Pattern = exPatternShadow + exPatternFrameShadow + exPatternBox + exPatternGradientVBox)
exBezierCurve	512	This option is valid for HistogramPattern property only. The exBezierCurve flag can be combined with any of the predefined patterns to define a filled bezier or empty curve (if combined with exPatternEmpty (ForeColor) or exPatternBox (BackColor)). For instance, if the HistogramPattern property is exBezierCurve+ exPatternEmpty, the bar's histogram shows a bezier curve not filled.
exRoundCurve	1024	This option is valid for HistogramPattern property only. The exRoundCurve flag can be combined with any of the predefined patterns to define a filled round curve or empty curve (if combined with exPatternEmpty (ForeColor) or exPatternBox (BackColor)). For instance, if the HistogramPattern property is exRoundCurve+ exPatternDiagCross, the bar's histogram shows a round curve filled with exPatternDiagCross pattern.
exRectangularCurve	2048	This option is valid for HistogramPattern property only. The exRectangularCurve flag can be combined with any of the predefined patterns to define a filled rectangular curve or empty rectangular curve (if combined with exPatternEmpty (ForeColor) or exPatternBox (BackColor)). For instance, if the HistogramPattern property is exRectangularCurve + exPatternYard, the bar's histogram shows a rectangular curve filled with exPatternYard pattern.

constants PictureDisplayEnum

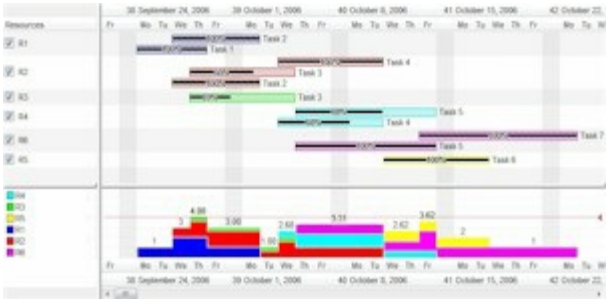
Specifies how the picture is displayed on the control's background. Use the PictureDisplay property to specify how the control displays its picture.

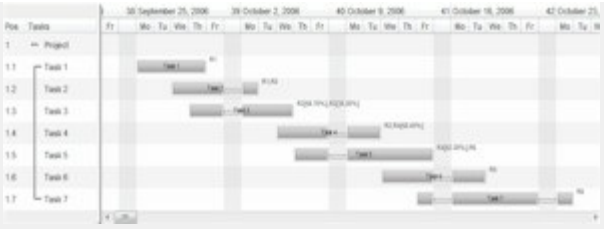
Name	Value	Description
UpperLeft	0	Aligns the picture to the upper left corner.
UpperCenter	1	Centers the picture on the upper edge.
UpperRight	2	Aligns the picture to the upper right corner.
MiddleLeft	16	Aligns horizontally the picture on the left side, and centers the picture vertically.
MiddleCenter	17	Puts the picture on the center of the source.
MiddleRight	18	Aligns horizontally the picture on the right side, and centers the picture vertically.
LowerLeft	32	Aligns the picture to the lower left corner.
LowerCenter	33	Centers the picture on the lower edge.
LowerRight	34	Aligns the picture to the lower right corner.
Tile	48	Tiles the picture on the source.
Stretch	49	The picture is resized to fit the source.

constants PutResEnum

The PutResEnum type indicates the values to load or save the resources associated to a bar using the Items.ItemBar(exBarResources), while using the PutRes method. The PutRes method saves or loads the bar's resources. The PutRes feature adds the ability to display the resources usage to a different exg2antt component. The PutRes method saves or updates the resources to or from a different exg2antt component. The PutRes feature must use 2 different eXG2antt controls.

The PutResEnum type supports the following values:

Name	Value	Description
exPutResLoad	1	Loads the bar's resources from another control. For instance, the Target.PutRes(Source.ResHandle, exPutResLoad) method loads the bar's usage of resources in Target. The exPutResLoad option must be used on the Target component that displays the Resources column, where all resources being found are set on lines, while bars in the chart indicates the bars in the Source that uses the current resource. The following picture shows the Target once the Target.PutRes(Source.ResHandle, exPutResLoad) is called: 
exPutResSave	2	Saves the bar's resources to another control. , For instance the Source.PutRes(Target.ResHandle, exPutResSave) saves the bar's allocations to Source taking data from the Target. The exPutResSave option must be used on Source control, so the changes in the Target control are updated to the Source control. The following picture shows the Source (the original Gantt control) once the Source.PutRes(Target.ResHandle, exPutResSave) is called:


38 September 25, 2006
39 October 2, 2006
40 October 9, 2006
41 October 16, 2006
42 October 23,
Fr Mo Tu We Th Fr Mo Tu We Th Fr Mo Tu We Th Fr Mo Tu We Th Fr Mo Tu W
Pos Tasks
1 Project
1.1 Task 1
1.2 Task 2
1.3 Task 3
1.4 Task 4
1.5 Task 5
1.6 Task 6
1.7 Task 7

constants ReadOnlyEnum

The ReadOnly property makes the control read-only. Use the Enabled property to disable the control. Use the Locked property to lock a specific editor. Use the CellEditorVisible property to hide the cell's editor.

Name	Value	Description
exReadWrite	0	(boolean False) The control allows changes. The user can use the cell's editor to change the cell's value.
exReadOnly	-1	(boolean True) The control is read only and the cell's editor is not visible.
exLocked	1	The control is read only, and the cell's editor is visible but locked. For instance, if the cell's editor contains a drop down portion, the user can display the drop down portion of the control, but it can't select a new value. Also, if the editor contains multiple buttons they are active as the control is not read only.

constants ResizeChartEnum

The ResizeChartEnum type indicates whether the user can **enlarge** or magnify (zoom-in,zoom-out) the entire chart, by dragging the header or resizing it using the middle mouse button, with or without re-scaling the chart. The AllowResizeChart property specifies whether the user can perform zoom-in/zoom-out over the control's chart area. The AllowResizeChart property supports the following options:

Name	Value	Description
exDisableResizeChart	0	The user can not enlarge or magnify (zoom-in,zoom-out) the entire chart, by dragging the header or resizing it using the middle mouse button.
exAllowResizeChartHeader	2	The user can enlarge or magnify (zoom-in,zoom-out) the entire chart, by dragging the header. Can be combined with any other non-zero option.
exAllowResizeChartMiddle	4	The user can enlarge or magnify (zoom-in,zoom-out) the entire chart, by resizing it using the middle mouse button. Can be combined with any other non-zero option.
exAllowChangeUnitScale	256	The chart's unit scale can be changed by resizing the chart at runtime. Can be combined with any other non-zero option.

constants ScrollBarEnum

The ScrollBarEnum type specifies the vertical or horizontal scroll bar in the control. Use the ScrollBars property to specify whether the vertical or horizontal scroll bar is visible or hidden. Use the ScrollPartVisible property to specify the visible parts in the control's scroll bars.

Name	Value	Description
exVScroll	0	Indicates the vertical scroll bar.
exHScroll	1	Indicates the horizontal scroll bar.
exHChartScroll	2	Indicates the horizontal scroll bar in the chart area.

constants ScrollBarsEnum

Specifies which scroll bars will be visible on a control. The ScrollBars property of the control specifies the scroll bars being visible in the control. By default, the ScrollBars property is exBoth, which indicates that both scroll bars of the component are being displayed only when they require.

- The horizontal scroll bar is not shown, if the ColumnAutoResize property is True, or if the ScrollBars property is exNoScroll. The horizontal scroll bar is shown if required, if the ScrollBars property is exBoth or exHorizontal, else it is always shown if the ScrollBars property is exDisableBoth or exDisableNoHorizontal
- The vertical scroll bar of the control is shown if required, if the ScrollBars is exBoth or exVertical, else if it is always shown if the ScrollBars property is exDisableBoth or exDisableVertical. For instance, if the ScrollBars property is exBoth OR exVScrollOnThumbRelease, the control's content is scrolled when the user releases the vertical thumb.

Use the Scroll method to programmatically scroll the control's content to specified position. The ScrollPos property determines the position of the control's scroll bars. The ScrollWidth property specifies the width in pixels, of the vertical scroll bar. The ScrollHeight property specifies the height in pixels of the horizontal scroll bar. The ScrollOrderParts property specifies the order to display the parts of the scroll bar (buttons, thumbs and so on). The ScrollPartCaption property specifies the caption to be shown on any part of the scroll bar. Use the SelectPos property to select items giving its position.

The ScrollBars property supports a bitwise OR combination of the following values:

Name	Value	Description
exNoScroll	0	No scroll bars are shown
exHorizontal	1	Only horizontal scroll bars are shown.
exVertical	2	Only vertical scroll bars are shown.
exBoth	3	Both horizontal and vertical scroll bars are shown.
exDisableNoHorizontal	5	The horizontal scroll bar is always shown, it is disabled if it is unnecessary.
exDisableNoVertical	10	The vertical scroll bar is always shown, it is disabled if it is unnecessary.
exDisableBoth	15	Both horizontal and vertical scroll bars are always shown, disabled if they are unnecessary.
exHScrollOnThumbRelease	256	Scrolls the control's content when the user releases the thumb of the horizontal scroll bar. Use this option to specify that the user scrolls the control's

		content when the thumb of the scroll box is released.
exVScrollOnThumbRelease	512	Scrolls the control's content when the user releases the thumb of the vertical scroll bar. Use this option to specify that the user scrolls the control's content when the thumb of the scroll box is released.
exHScrollEmptySpace	1024	Allows empty space, when the control's content is horizontally scrolled to the end. If this flag is set, the user can horizontally scrolls the control's content until last column is visible, and so you can have empty space to the right or left of the columns.
exVScrollEmptySpace	2048	Allows empty space, when the control's content is vertically scrolled to the end. If this flag is set, the user can vertically scrolls the control's content until last item is visible, and so you can have empty space after the last visible item.

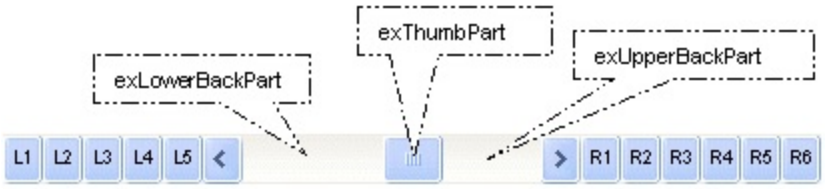
constants ScrollEnum

The ScrollEnum expression indicates the type of scroll that control supports. Use the Scroll method to scroll the control's content by code.

Name	Value	Description
exScrollUp	0	Scrolls up the control by a single line.
exScrollDown	1	Scrolls down the control by a single line.
exScrollVTo	2	Scrolls vertically the control to a specified position.
exScrollLeft	3	Scrolls the control to the left by a single pixel, or by a single column if the ContinueColumnScroll property is True.
exScrollRight	4	Scrolls the control to the right by a single pixel, or by a single column if the ContinueColumnScroll property is True.
exScrollHTo	5	Scrolls horizontaly the control to a specified position.

constants ScrollPartEnum

The ScrollPartEnum type defines the parts in the control's scrollbar. Use the ScrollPartVisible property to specify the visible parts in the control's scroll bar. Use the ScrollPartCaption property to specify the caption being displayed in any part of the control's scrollbar. The control fires the ScrollButtonClick event when the user clicks any button in the control's scrollbar.



Name	Value	Description
exExtentThumbPart	65536	The thumb-extension part.
exLeftB1Part	32768	(L1) The first additional button, in the left or top area. By default, this button is hidden.
exLeftB2Part	16384	(L2) The second additional button, in the left or top area. By default, this button is hidden.
exLeftB3Part	8192	(L3) The third additional button, in the left or top area. By default, this button is hidden.
exLeftB4Part	4096	(L4) The forth additional button, in the left or top area. By default, this button is hidden.
exLeftB5Part	2048	(L5) The fifth additional button, in the left or top area. By default, this button is hidden.
exLeftBPart	1024	(<) The left or top button. By default, this button is visible.
exLowerBackPart	512	The area between the left/top button and the thumb. By default, this part is visible.
exThumbPart	256	The thumb part or the scroll box region. By default, the thumb is visible.
exUpperBackPart	128	The area between the thumb and the right/bottom button. By default, this part is visible.
exBackgroundPart	640	The union between the exLowerBackPart and the exUpperBackPart parts. By default, this part is visible.
exRightBPart	64	(>) The right or down button. By default, this button is visible.

exRightB1Part	32	(R1) The first additional button in the right or down side. By default, this button is hidden.
exRightB2Part	16	(R2) The second additional button in the right or down side. By default, this button is hidden.
exRightB3Part	8	(R3) The third additional button in the right or down side. By default, this button is hidden.
exRightB4Part	4	(R4) The forth additional button in the right or down side. By default, this button is hidden
exRightB5Part	2	(R5) The fifth additional button in the right or down side. By default, this button is hidden.
exRightB6Part	1	(R6) The sixth additional button in the right or down side. By default, this button is hidden.
exPartNone	0	No part.

constants ScrollRangeEnum

The ScrollRangeEnum type specifies the positions being accessed by the ScrollRange property. The ScrollRange method specifies that the chart to be scrolled within a range of dates. Use the ItemBar property to access properties of a created bar. The CreateBar event is called once the user creates at runtime a new bar by drag and drop on the chart section.

The ScrollRangeEnum type supports the following values.

Name	Value	Description
exStartDate	0	Indicates that the starting date or time of the scrolling range is accessed or requested.
exEndDate	1	Indicates that the ending date or time of the scrolling range is accessed or requested.
exMinDate	2	This option is read-only, so setting the exMinDate has no effect. Instead use the exStartDate. Retrieves the minimum date when the chart's scrolling range is specified by Chart.ScrollRange(exStartDate) and Chart.ScrollRange(exEndDate). For instance, you can specify the .Items.ItemBar(Item, "newbar", exBarMinStart) = .Chart.ScrollRange(exMinDate) to limit the starting point of the bar to the scrolling range.
exMaxDate	3	This option is read-only, so setting the exMaxDate has no effect. Instead use the exEndDate. Retrieves the maximum date when the chart's scrolling range is specified by Chart.ScrollRange(exStartDate) and Chart.ScrollRange(exEndDate). For instance, you can specify the .Items.ItemBar(Item, "newbar", exBarMaxEnd) = .Chart.ScrollRange(exMaxDate) to limit the ending point of the bar to the scrolling range.

The following samples shows how can I limit the bars to scrolling range only.

VBA

```
' CreateBar event - Fired when the user creates a new bar.
Private Sub G2antt1_CreateBar(ByVal Item As Long,ByVal DateStart As Date,ByVal DateEnd
```

```
As Date)
   With G2antt1
      With .Items
         .ItemBar(Item,"newbar",22) = G2antt1.Chart.ScrollRange(2)
         .ItemBar(Item,"newbar",25) = G2antt1.Chart.ScrollRange(3)
      End With
   End With
End Sub

With G2antt1
   .BeginUpdate
   .Columns.Add "Task"
   With .Chart
      .LevelCount = 2
      .PaneWidth(0) = 56
      .ScrollRange(0) = #1/1/2001#
      .ScrollRange(1) = #1/15/2001#
      .FirstVisibleDate = #1/12/2001#
      .AllowCreateBar = 1
   End With
   With .Items
      .AddItem "Task 1"
      .AddItem "Task 2"
      .AddItem "Task 3"
   End With
   .EndUpdate
End With
```

VB6

```
' CreateBar event - Fired when the user creates a new bar.
Private Sub G2antt1_CreateBar(ByVal Item As EXG2ANTTLibCtl.HITEM,ByVal DateStart As
Date,ByVal DateEnd As Date)
   With G2antt1
      With .Items
         .ItemBar(Item,"newbar",exBarMinStart) = G2antt1.Chart.ScrollRange(exMinDate)
         .ItemBar(Item,"newbar",exBarMaxEnd) = G2antt1.Chart.ScrollRange(exMaxDate)
```

```
      End With
   End With
End Sub

With G2antt1
   .BeginUpdate
   .Columns.Add "Task"
   With .Chart
      .LevelCount = 2
      .PaneWidth(0) = 56
      .ScrollRange(exStartDate) = #1/1/2001#
      .ScrollRange(exEndDate) = #1/15/2001#
      .FirstVisibleDate = #1/12/2001#
      .AllowCreateBar = exCreateBarAuto
   End With
   With .Items
      .AddItem "Task 1"
      .AddItem "Task 2"
      .AddItem "Task 3"
   End With
   .EndUpdate
End With
```

VB.NET

```
' CreateBar event - Fired when the user creates a new bar.
Private Sub Exg2antt1_CreateBar(ByVal sender As System.Object,ByVal Item As
Integer,ByVal DateStart As Date,ByVal DateEnd As Date) Handles Exg2antt1.CreateBar
   With Exg2antt1
      With .Items

.set_ItemBar(Item,"newbar",exontrol.EXG2ANTTLib.ItemBarPropertyEnum.exBarMinStart,Ex



.set_ItemBar(Item,"newbar",exontrol.EXG2ANTTLib.ItemBarPropertyEnum.exBarMaxEnd,Ex

      End With
```

```
   End With
End Sub

With Exg2antt1
   .BeginUpdate()
   .Columns.Add("Task")
   With .Chart
      .LevelCount = 2
      .set_PaneWidth(False,56)
      .set_ScrollRange(exontrol.EXG2ANTTLib.ScrollRangeEnum.exStartDate,#1/1/2001#)
      .set_ScrollRange(exontrol.EXG2ANTTLib.ScrollRangeEnum.exEndDate,#1/15/2001#)
      .FirstVisibleDate = #1/12/2001#
      .AllowCreateBar = exontrol.EXG2ANTTLib.CreateBarEnum.exCreateBarAuto
   End With
   With .Items
      .AddItem("Task 1")
      .AddItem("Task 2")
      .AddItem("Task 3")
   End With
   .EndUpdate()
End With
```

VB.NET for /COM

```
' CreateBar event - Fired when the user creates a new bar.
Private Sub AxG2antt1_CreateBar(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_CreateBarEvent) Handles AxG2antt1.CreateBar
   With AxG2antt1
      With .Items
         .ItemBar(e.item,"newbar",EXG2ANTTLib.ItemBarPropertyEnum.exBarMinStart) =
AxG2antt1.Chart.ScrollRange(EXG2ANTTLib.ScrollRangeEnum.exMinDate)
         .ItemBar(e.item,"newbar",EXG2ANTTLib.ItemBarPropertyEnum.exBarMaxEnd) =
AxG2antt1.Chart.ScrollRange(EXG2ANTTLib.ScrollRangeEnum.exMaxDate)
      End With
   End With
End Sub
```

```
With AxG2antt1
   .BeginUpdate()
   .Columns.Add("Task")
   With .Chart
      .LevelCount = 2
      .PaneWidth(False) = 56
      .ScrollRange(EXG2ANTTLib.ScrollRangeEnum.exStartDate) = #1/1/2001#
      .ScrollRange(EXG2ANTTLib.ScrollRangeEnum.exEndDate) = #1/15/2001#
      .FirstVisibleDate = #1/12/2001#
      .AllowCreateBar = EXG2ANTTLib.CreateBarEnum.exCreateBarAuto
   End With
   With .Items
      .AddItem("Task 1")
      .AddItem("Task 2")
      .AddItem("Task 3")
   End With
   .EndUpdate()
End With
```

C++

```
// CreateBar event - Fired when the user creates a new bar.
void OnCreateBarG2antt1(long Item,DATE DateStart,DATE DateEnd)
{
   /*
      Copy and paste the following directives to your header file as
      it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'
      #import <ExG2antt.dll>
      using namespace EXG2ANTTLib
   */
   EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown()
   EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems()
      var_Items->PutItemBar(Item,"newbar",EXG2ANTTLib::exBarMinStart,spG2antt1-
>GetChart()->GetScrollRange(EXG2ANTTLib::exMinDate))
      var_Items->PutItemBar(Item,"newbar",EXG2ANTTLib::exBarMaxEnd,spG2antt1-
>GetChart()->GetScrollRange(EXG2ANTTLib::exMaxDate))
```

```
}

EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
spG2antt1->GetColumns()->Add(L"Task");
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutLevelCount(2);
   var_Chart->PutPaneWidth(VARIANT_FALSE,56);
   var_Chart->PutScrollRange(EXG2ANTTLib::exStartDate,"1/1/2001");
   var_Chart->PutScrollRange(EXG2ANTTLib::exEndDate,"1/15/2001");
   var_Chart->PutFirstVisibleDate("1/12/2001");
   var_Chart->PutAllowCreateBar(EXG2ANTTLib::exCreateBarAuto);
EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems();
   var_Items->AddItem("Task 1");
   var_Items->AddItem("Task 2");
   var_Items->AddItem("Task 3");
spG2antt1->EndUpdate();
```

C++ Builder

```
// CreateBar event - Fired when the user creates a new bar.
void __fastcall TForm1::G2antt1CreateBar(TObject *Sender,Exg2anttlib_tlb::HITEM
Item,DATE DateStart,DATE DateEnd)
{
   Exg2anttlib_tlb::IItemsPtr var_Items = G2antt1->Items
      var_Items-
>set_ItemBar(Item,TVariant("newbar"),Exg2anttlib_tlb::ItemBarPropertyEnum::exBarMinSta
>Chart->get_ScrollRange(Exg2anttlib_tlb::ScrollRangeEnum::exMinDate)))
      var_Items-
>set_ItemBar(Item,TVariant("newbar"),Exg2anttlib_tlb::ItemBarPropertyEnum::exBarMaxEnd
>Chart->get_ScrollRange(Exg2anttlib_tlb::ScrollRangeEnum::exMaxDate)))
}

G2antt1->BeginUpdate();
G2antt1->Columns->Add(L"Task");
Exg2anttlib_tlb::IChartPtr var_Chart = G2antt1->Chart;
```

```
    var_Chart->LevelCount = 2;
    var_Chart->set_PaneWidth(false,56);
    var_Chart-
>set_ScrollRange(Exg2anttlib_tlb::ScrollRangeEnum::exStartDate,TVariant(TDateTime(2001
 double()));
    var_Chart-
>set_ScrollRange(Exg2anttlib_tlb::ScrollRangeEnum::exEndDate,TVariant(TDateTime(2001,
 double()));
    var_Chart->set_FirstVisibleDate(TVariant(TDateTime(2001,1,12).operator double()));
    var_Chart->AllowCreateBar = Exg2anttlib_tlb::CreateBarEnum::exCreateBarAuto;
Exg2anttlib_tlb::IItemsPtr var_Items = G2antt1->Items;
    var_Items->AddItem(TVariant("Task 1"));
    var_Items->AddItem(TVariant("Task 2"));
    var_Items->AddItem(TVariant("Task 3"));
G2antt1->EndUpdate();
```

C#

```
// CreateBar event - Fired when the user creates a new bar.
private void exg2antt1_CreateBar(object sender,int Item,DateTime DateStart,DateTime
DateEnd)
{
    exontrol.EXG2ANTTLib.Items var_Items = exg2antt1.Items

var_Items.set_ItemBar(Item,"newbar",exontrol.EXG2ANTTLib.ItemBarPropertyEnum.exBarM



var_Items.set_ItemBar(Item,"newbar",exontrol.EXG2ANTTLib.ItemBarPropertyEnum.exBarM

}
//this.exg2antt1.CreateBar += new
exontrol.EXG2ANTTLib.exg2antt.CreateBarEventHandler(this.exg2antt1_CreateBar);



exg2antt1.BeginUpdate();
exg2antt1.Columns.Add("Task");
exontrol.EXG2ANTTLib.Chart var_Chart = exg2antt1.Chart;
```

```
   var_Chart.LevelCount = 2;
   var_Chart.set_PaneWidth(false,56);

var_Chart.set_ScrollRange(exontrol.EXG2ANTTLib.ScrollRangeEnum.exStartDate,Convert.T



var_Chart.set_ScrollRange(exontrol.EXG2ANTTLib.ScrollRangeEnum.exEndDate,Convert.To

   var_Chart.FirstVisibleDate = Convert.ToDateTime("1/12/2001");
   var_Chart.AllowCreateBar = exontrol.EXG2ANTTLib.CreateBarEnum.exCreateBarAuto;
exontrol.EXG2ANTTLib.Items var_Items = exg2antt1.Items;
   var_Items.AddItem("Task 1");
   var_Items.AddItem("Task 2");
   var_Items.AddItem("Task 3");
exg2antt1.EndUpdate();
```

JavaScript

```
<SCRIPT FOR="G2antt1" EVENT="CreateBar(Item,DateStart,DateEnd)"
LANGUAGE="JScript">
   var var_Items = G2antt1.Items
      var_Items.ItemBar(Item,"newbar",22) = G2antt1.Chart.ScrollRange(2)
      var_Items.ItemBar(Item,"newbar",25) = G2antt1.Chart.ScrollRange(3)
</SCRIPT>

<OBJECT classid="clsid:CD481F4D-2D25-4759-803F-752C568F53B7" id="G2antt1">
</OBJECT>

<SCRIPT LANGUAGE="JScript">
   G2antt1.BeginUpdate()

   G2antt1.Columns.Add("Task")

   var var_Chart = G2antt1.Chart

      var_Chart.LevelCount = 2
```

```
    var_Chart.PaneWidth(0) = 56

    var_Chart.ScrollRange(0) = "1/1/2001"

    var_Chart.ScrollRange(1) = "1/15/2001"

    var_Chart.FirstVisibleDate = "1/12/2001"

    var_Chart.AllowCreateBar = 1

  var var_Items = G2antt1.Items

    var_Items.AddItem("Task 1")

    var_Items.AddItem("Task 2")

    var_Items.AddItem("Task 3")

  G2antt1.EndUpdate()

</SCRIPT>
```

C# for /COM

```
// CreateBar event - Fired when the user creates a new bar.
private void axG2antt1_CreateBar(object sender,
AxEXG2ANTTLib._IG2anttEvents_CreateBarEvent e)
{
  EXG2ANTTLib.Items var_Items = axG2antt1.Items

var_Items.set_ItemBar(e.item,"newbar",EXG2ANTTLib.ItemBarPropertyEnum.exBarMinStart,


var_Items.set_ItemBar(e.item,"newbar",EXG2ANTTLib.ItemBarPropertyEnum.exBarMaxEnd,a

}
//this.axG2antt1.CreateBar += new
```

```
AxEXG2ANTTLib._IG2anttEvents_CreateBarEventHandler(this.axG2antt1_CreateBar);



axG2antt1.BeginUpdate();
axG2antt1.Columns.Add("Task");
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.LevelCount = 2;
   var_Chart.set_PaneWidth(false,56);

var_Chart.set_ScrollRange(EXG2ANTTLib.ScrollRangeEnum.exStartDate,Convert.ToDateTim



var_Chart.set_ScrollRange(EXG2ANTTLib.ScrollRangeEnum.exEndDate,Convert.ToDateTime

   var_Chart.FirstVisibleDate = Convert.ToDateTime("1/12/2001");
   var_Chart.AllowCreateBar = EXG2ANTTLib.CreateBarEnum.exCreateBarAuto;
EXG2ANTTLib.Items var_Items = axG2antt1.Items;
   var_Items.AddItem("Task 1");
   var_Items.AddItem("Task 2");
   var_Items.AddItem("Task 3");
axG2antt1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
// CreateBar event - Fired when the user creates a new bar.
void onEvent_CreateBar(int _Item,date _DateStart,date _DateEnd)
{
   COM com_Items
   anytype var_Items
   var_Items = exg2antt1.Items()
    com_Items = var_Items

com_Items.ItemBar(_Item,"newbar",22/*exBarMinStart*/,exg2antt1.Chart().ScrollRange(2/*



com_Items.ItemBar(_Item,"newbar",25/*exBarMaxEnd*/,exg2antt1.Chart().ScrollRange(3/*e
```

```
}

public void init()
{
   COM com_Chart,com_Items

   anytype var_Chart,var_Items



   super()


   exg2antt1.BeginUpdate()

   exg2antt1.Columns().Add("Task")

   var_Chart = exg2antt1.Chart()
    com_Chart = var_Chart

      com_Chart.LevelCount(2)

   /*should be called during the form's activate method*/     com_Chart.PaneWidth(0,56);

com_Chart.ScrollRange(0/*exStartDate*/,COMVariant::createFromDate(str2Date("1/1/2001



com_Chart.ScrollRange(1/*exEndDate*/,COMVariant::createFromDate(str2Date("1/15/2001


      com_Chart.FirstVisibleDate(COMVariant::createFromDate(str2Date("1/12/2001",213)))

      com_Chart.AllowCreateBar(1/*exCreateBarAuto*/)

   var_Items = exg2antt1.Items()
```

```
    com_Items = var_Items

       com_Items.AddItem("Task 1")

       com_Items.AddItem("Task 2")

       com_Items.AddItem("Task 3")

    exg2antt1.EndUpdate()

}

/*
public void activate(boolean _active)
{
    super(_active)


    exg2antt1.Chart().PaneWidth(0,56)

}
*/
```

VFP

```
*** CreateBar event - Fired when the user creates a new bar. ***
LPARAMETERS Item,DateStart,DateEnd
    with thisform.G2antt1
        with .Items
            .ItemBar(Item,"newbar",22) = thisform.G2antt1.Chart.ScrollRange(2)
            .ItemBar(Item,"newbar",25) = thisform.G2antt1.Chart.ScrollRange(3)
        endwith
    endwith

with thisform.G2antt1
    .BeginUpdate
    .Columns.Add("Task")
```

```
   with .Chart
      .LevelCount = 2
      .PaneWidth(0) = 56
      .ScrollRange(0) = {^2001-1-1}
      .ScrollRange(1) = {^2001-1-15}
      .FirstVisibleDate = {^2001-1-12}
      .AllowCreateBar = 1
   endwith
   with .Items
      .AddItem("Task 1")
      .AddItem("Task 2")
      .AddItem("Task 3")
   endwith
   .EndUpdate
endwith
```

dBASE Plus

```
/*
with (this.ACTIVEX1.nativeObject)
   CreateBar = class::nativeObject_CreateBar
endwith
*/
// Fired when the user creates a new bar.
function nativeObject_CreateBar(Item,DateStart,DateEnd)
   local oG2antt,var_Items
   oG2antt = form.Activex1.nativeObject
   var_Items = oG2antt.Items
      // var_Items.ItemBar(Item,"newbar",22) = oG2antt.Chart.ScrollRange(2)
      with (oG2antt)
         TemplateDef = [Dim var_Items,Item]
         TemplateDef = var_Items
         TemplateDef = Item
         Template = [var_Items.ItemBar(Item,"newbar",22) = oG2antt.Chart.ScrollRange(2)]
      endwith
      // var_Items.ItemBar(Item,"newbar",25) = oG2antt.Chart.ScrollRange(3)
      with (oG2antt)
```

```
        TemplateDef = [Dim var_Items,Item]
        TemplateDef = var_Items
        TemplateDef = Item
        Template = [var_Items.ItemBar(Item,"newbar",25) = oG2antt.Chart.ScrollRange(3)]
    endwith
return

local oG2antt,var_Chart,var_Items

oG2antt = form.Activex1.nativeObject
oG2antt.BeginUpdate()
oG2antt.Columns.Add("Task")
var_Chart = oG2antt.Chart
  var_Chart.LevelCount = 2
  // var_Chart.PaneWidth(false) = 56
  with (oG2antt)
    TemplateDef = [Dim var_Chart]
    TemplateDef = var_Chart
    Template = [var_Chart.PaneWidth(false) = 56]
  endwith
  // var_Chart.ScrollRange(0) = "01/01/2001"
  with (oG2antt)
    TemplateDef = [Dim var_Chart]
    TemplateDef = var_Chart
    Template = [var_Chart.ScrollRange(0) = "01/01/2001"]
  endwith
  // var_Chart.ScrollRange(1) = "01/15/2001"
  with (oG2antt)
    TemplateDef = [Dim var_Chart]
    TemplateDef = var_Chart
    Template = [var_Chart.ScrollRange(1) = "01/15/2001"]
  endwith
  var_Chart.FirstVisibleDate = "01/12/2001"
  var_Chart.AllowCreateBar = 1
var_Items = oG2antt.Items
  var_Items.AddItem("Task 1")
  var_Items.AddItem("Task 2")
```

```
   var_Items.AddItem("Task 3")
oG2antt.EndUpdate()
```

XBasic (Alpha Five)

```
' Fired when the user creates a new bar.
function CreateBar as v (Item as OLE::Exontrol.G2antt.1::HITEM,DateStart as T,DateEnd as T)
   Dim oG2antt as P
   Dim var_Items as P
   oG2antt = topparent:CONTROL_ACTIVEX1.activex
   var_Items = oG2antt.Items
      ' var_Items.ItemBar(Item,"newbar",22) = oG2antt.Chart.ScrollRange(2)
      oG2antt.TemplateDef = "Dim var_Items,Item"
      oG2antt.TemplateDef = var_Items
      oG2antt.TemplateDef = Item
      oG2antt.Template = "var_Items.ItemBar(Item,\"newbar\",22) =
oG2antt.Chart.ScrollRange(2)"
      ' var_Items.ItemBar(Item,"newbar",25) = oG2antt.Chart.ScrollRange(3)
      oG2antt.TemplateDef = "Dim var_Items,Item"
      oG2antt.TemplateDef = var_Items
      oG2antt.TemplateDef = Item
      oG2antt.Template = "var_Items.ItemBar(Item,\"newbar\",25) =
oG2antt.Chart.ScrollRange(3)"
end function

Dim oG2antt as P
Dim var_Chart as P
Dim var_Items as P

oG2antt = topparent:CONTROL_ACTIVEX1.activex
oG2antt.BeginUpdate()
oG2antt.Columns.Add("Task")
var_Chart = oG2antt.Chart
   var_Chart.LevelCount = 2
   ' var_Chart.PaneWidth(.f.) = 56
   oG2antt.TemplateDef = "Dim var_Chart"
   oG2antt.TemplateDef = var_Chart
```

```
    oG2antt.Template = "var_Chart.PaneWidth(False) = 56"

    ' var_Chart.ScrollRange(0) = {01/01/2001}
    oG2antt.TemplateDef = "Dim var_Chart"
    oG2antt.TemplateDef = var_Chart
    oG2antt.Template = "var_Chart.ScrollRange(0) = #01/01/2001#"

    ' var_Chart.ScrollRange(1) = {01/15/2001}
    oG2antt.TemplateDef = "Dim var_Chart"
    oG2antt.TemplateDef = var_Chart
    oG2antt.Template = "var_Chart.ScrollRange(1) = #01/15/2001#"

    var_Chart.FirstVisibleDate = {01/12/2001}
    var_Chart.AllowCreateBar = 1
var_Items = oG2antt.Items
    var_Items.AddItem("Task 1")
    var_Items.AddItem("Task 2")
    var_Items.AddItem("Task 3")
oG2antt.EndUpdate()
```

Delphi 8 (.NET only)

```
// CreateBar event - Fired when the user creates a new bar.
procedure TWinForm1.AxG2antt1_CreateBar(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_CreateBarEvent);
begin
   with AxG2antt1 do
   begin
      with Items do
      begin
         ItemBar[e.item,'newbar',EXG2ANTTLib.ItemBarPropertyEnum.exBarMinStart] :=
AxG2antt1.Chart.ScrollRange[EXG2ANTTLib.ScrollRangeEnum.exMinDate];
         ItemBar[e.item,'newbar',EXG2ANTTLib.ItemBarPropertyEnum.exBarMaxEnd] :=
AxG2antt1.Chart.ScrollRange[EXG2ANTTLib.ScrollRangeEnum.exMaxDate];
      end
   end
end;
```

```
with AxG2antt1 do
begin
   BeginUpdate();
   Columns.Add('Task');
   with Chart do
   begin
      LevelCount := 2;
      PaneWidth[False] := 56;
      ScrollRange[EXG2ANTTLib.ScrollRangeEnum.exStartDate] := '1/1/2001';
      ScrollRange[EXG2ANTTLib.ScrollRangeEnum.exEndDate] := '1/15/2001';
      FirstVisibleDate := '1/12/2001';
      AllowCreateBar := EXG2ANTTLib.CreateBarEnum.exCreateBarAuto;
   end;
   with Items do
   begin
      AddItem('Task 1');
      AddItem('Task 2');
      AddItem('Task 3');
   end;
   EndUpdate();
end
```

Delphi (standard)

```
// CreateBar event - Fired when the user creates a new bar.
procedure TForm1.G2antt1CreateBar(ASender: TObject; Item : HITEM;DateStart :
TDateTime;DateEnd : TDateTime);
begin
   with G2antt1 do
   begin
      with Items do
      begin
         ItemBar[Item,'newbar',EXG2ANTTLib_TLB.exBarMinStart] :=
G2antt1.Chart.ScrollRange[EXG2ANTTLib_TLB.exMinDate];
         ItemBar[Item,'newbar',EXG2ANTTLib_TLB.exBarMaxEnd] :=
G2antt1.Chart.ScrollRange[EXG2ANTTLib_TLB.exMaxDate];
```

```
      end
   end
end;

with G2antt1 do
begin
   BeginUpdate();
   Columns.Add('Task');
   with Chart do
   begin
      LevelCount := 2;
      PaneWidth[False] := 56;
      ScrollRange[EXG2ANTTLib_TLB.exStartDate] := '1/1/2001';
      ScrollRange[EXG2ANTTLib_TLB.exEndDate] := '1/15/2001';
      FirstVisibleDate := '1/12/2001';
      AllowCreateBar := EXG2ANTTLib_TLB.exCreateBarAuto;
   end;
   with Items do
   begin
      AddItem('Task 1');
      AddItem('Task 2');
      AddItem('Task 3');
   end;
   EndUpdate();
end
```

Visual Objects

```
METHOD OCX_Exontrol1CreateBar(Item,DateStart,DateEnd) CLASS MainDialog
   // CreateBar event - Fired when the user creates a new bar.
   local var_Items as IItems
   var_Items := oDCOCX_Exontrol1:Items
      var_Items:[ItemBar,Item,"newbar",exBarMinStart] := oDCOCX_Exontrol1:Chart:
[ScrollRange,exMinDate]
      var_Items:[ItemBar,Item,"newbar",exBarMaxEnd] := oDCOCX_Exontrol1:Chart:
[ScrollRange,exMaxDate]
RETURN NIL
```

```
local var_Chart as IChart
local var_Items as IItems

oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:Columns:Add("Task")
var_Chart := oDCOCX_Exontrol1:Chart
   var_Chart:LevelCount := 2
   var_Chart:[PaneWidth,false] := 56
   var_Chart:[ScrollRange,exStartDate] := SToD("20010101")
   var_Chart:[ScrollRange,exEndDate] := SToD("20010115")
   var_Chart:FirstVisibleDate := SToD("20010112")
   var_Chart:AllowCreateBar := exCreateBarAuto
var_Items := oDCOCX_Exontrol1:Items
   var_Items:AddItem("Task 1")
   var_Items:AddItem("Task 2")
   var_Items:AddItem("Task 3")
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
/*begin event CreateBar(long Item,datetime DateStart,datetime DateEnd) - Fired when the user creates a new bar.*/
/*
   OleObject oG2antt,var_Items
   oG2antt = ole_1.Object
   var_Items = oG2antt.Items
      var_Items.ItemBar(Item,"newbar",22,oG2antt.Chart.ScrollRange(2))
      var_Items.ItemBar(Item,"newbar",25,oG2antt.Chart.ScrollRange(3))
*/
/*end event CreateBar*/

OleObject oG2antt,var_Chart,var_Items

oG2antt = ole_1.Object
oG2antt.BeginUpdate()
oG2antt.Columns.Add("Task")
```

```
var_Chart = oG2antt.Chart
   var_Chart.LevelCount = 2
   var_Chart.PaneWidth(false,56)
   var_Chart.ScrollRange(0,2001-01-01)
   var_Chart.ScrollRange(1,2001-01-15)
   var_Chart.FirstVisibleDate = 2001-01-12
   var_Chart.AllowCreateBar = 1
var_Items = oG2antt.Items
   var_Items.AddItem("Task 1")
   var_Items.AddItem("Task 2")
   var_Items.AddItem("Task 3")
oG2antt.EndUpdate()
```

constants SelectDateEnum

The SelectDateEnum type specifies how the user can select dates in the chart part of the control. The user can select a date in the chart at runtime by clicking the date in the control's header. The AllowSelectDate property specifies whether the user can select a date by clicking the chart's header. The MarkSelectDateColor property indicates the color to show the selected dates within the chart. Programmatically you can select a date using the SelectDate or SelectDates properties. The SelectLevel property indicates the level being selected. The selected dates are not shown if the MarkSelectDateColor property has the same value as BackColor property in the Chart object. The control fires the ChartStartChaning(exSelectDate)/ChartEndChaning(exSelectDate) events once the user selects a new date by clicking the chart's header.

Name	Value	Description
exNoSelectDate	0	No date is selected if the user clicks the chart's header. Even so, you still can programmatically show selected dates using the SelectDate or SelectDates properties.
exSelectDefault	-1	A date is being selected once the user clicks the chart's header. The old selection of dates is cleared if no CTRL key is pressed. Clicking the date by keeping the CTRL key down, will select or unselect the date from the cursor. The SelectLevel property indicates the date from the level is selected. For instance, if the chart displays 2 levels, months and days, and clicking the month header, the entire month is being selected. Clicking the days header will make the day selected. If the chart is zoom-in or zoom-out (a new scale is being selected) the selected zones are shown relative to the new levels. This option can **not** be combined with any flag.
exSelectSingleDate	8	Only a single date can be selected by clicking the chart's header. This is a bit-flag that can be combined with other flags.
exSelectToggle	16	A date is selected if previously was not selected and a date is being unselected if previously was selected. This is a bit-flag that can be combined with other flags. For instance, if the AllowSelectDate property is exSelectToggle + exSelectZone, indicates that the user can toggle the selected dates, and the zones are shown relative to

		any scale when zoom-in or zoom-out is preformed.
exSelectZone	256	The zone is being selected once the user clicks the chart's header. For instance, if the chart displays 2 levels, months and days, and clicking the month header, the entire month is being selected. Clicking the days header will make the day selected. If the control is zoom-in or zoom-out zone will be relative to the new scale, so it is not indicating another zone. The zone is keep even if the user zoom-in or zoom-out the chart, so the original selected zone will be shown relative to the new scale, comparing with with the exSelectDefault, where the zone is lost once a new scale is being selected. This is a bit-flag that can be combined with other flags. For instance, if the AllowSelectDate property is exSelectToggle + exSelectZone, indicates that the user can toggle the selected dates, and the zones are shown relative to any scale when zoom-in or zoom-out is preformed.

constants SelectObjectsEnum

The SelectObjectEnum type specifies whether the user can select bars, links single bar selection or single link selection, and so on. Use the AllowSelectObjects property to specify the objects that can be selected in the chart at runtime. Use the SelectedObjects property to get a list of selected objects based on your criteria.

Name	Value	Description
exNoSelectObjects	0	The user can't select any object in the chart area.
exSelectBarsOnly	1	The user can select bars only.
exSelectLinksOnly	2	The user can select links only.
exSelectObjects	3	The user can select any object in the chart.
exSelectSingleObject	16	If present, it specifies whether the user can select one or multiple objects. For instance, the exSelectBarsOnly Or exSelectSingleObject specifies that the user can select a single bar in the chart. The exSelectLinksOnly Or exSelectSingleObject specifies that the user can select a single link in the chart.
exObjectsJustAdded	32	The SelectedObjects property retrieves only the object being added since last selection change. For instance, the SelectedObjects(exSelectBarsOnly Or exObjectsJustAdded) retrieves a collection that specifies only the bars being selected since last selection change.
exObjectsJustRemoved	64	The SelectedObjects property retrieves only the object being removed since last selection change. For instance, the SelectedObjects(exSelectBarsOnly Or exObjectsJustRemoved) retrieves a collection that specifies only the bars being un-selected since last selection change.

constants ShapeBarEnum

The ShapeBarEnum type indicates the height and the alignment of the bar. Use the Shape property to specify the height and the vertical alignment of the bar.

Name	Value	Description
exShapeEmpty	0	The shape is empty.
exShapeSolid	1	The bar draws the frame around.
exShapeSolidUp	2	
exShapeSolidCenter	3	
exShapeSolidDown	4	
exShapeSolidFrameless	17	The bar does not draw the frame around.
exShapeThinUp	18	
exShapeThinCenter	19	
exShapeThinDown	20	

constants ShapeCornerEnum

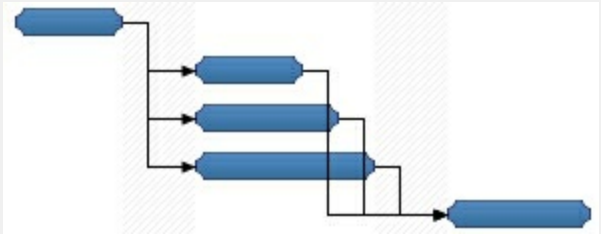
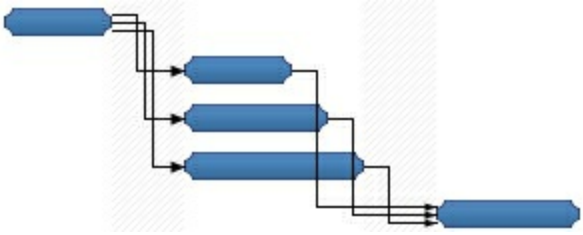
The ShapeCornerEnum expression defines the shape of the start and end part of the bar. Use the StartShape and EndShape properties to define the start and end parts of the bar using custom shapes. Use the AddShapeCorner method to define a corner from an icon. Use the Images or ReplaceIcon method to update the list of control's icons.

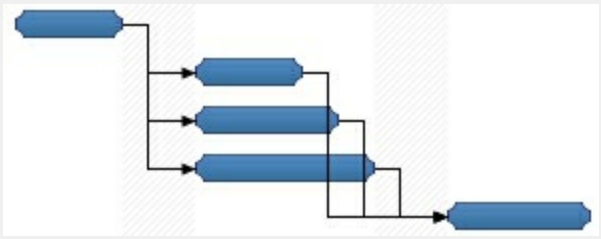
Name	Value	Description
exShapeIconEmpty	0	No corner.
exShapeIconUp1	1	
exShapeIconDown1	2	
exShapeIconRhombus	3	
exShapeIconCircleDot	4	
exShapeIconUp2	5	
exShapeIconDown2	6	
exShapeIconLeft	7	
exShapeIconRight	8	
exShapeIconCircleUp1	9	
exShapeIconCircleDown1	10	
exShapeIconUp3	11	
exShapeIconDown3	12	
exShapeIconCircleUp2	13	
exShapeIconCircleDown2	14	
exShapeIconUp4	15	
exShapeIconDown4	16	
exShapeIconVBar	17	
exShapeIconSquare	18	
exShapeIconCircle	19	
exShapeIconStar	20	
exShapeIconFrameUp1	61441	
exShapeIconFrameDown1	61442	
exShapeIconFrameRhombus	61443	
exShapeIconFrameCircleDot	61444	

exShapeIconFrameUp2	61445	△
exShapeIconFrameDown2	61446	▽
exShapeIconFrameLeft	61447	◁
exShapeIconFrameRight	61448	▷
exShapeIconFrameCircleUp1	61449	
exShapeIconFrameCircleDown1	61450	
exShapeIconFrameUp3	61451	
exShapeIconFrameDown3	61452	
exShapeIconFrameCircleUp2	61453	
exShapeIconFrameCircleDown2	61454	
exShapeIconFrameUp4	61455	
exShapeIconFrameDown4	61456	
exShapeIconFrameVBar	61457	▯
exShapeIconFrameSquare	61458	▫
exShapeIconFrameCircle	61459	○
exShapeIconFrameStar	61460	☆

constants ShowExtendedLinksEnum

The ShowExtendedLinksEnum type specifies the way the control shows the links between bars in the chart area. The ShowLinks property specifies whether the links are shown or hidden. The ShowExtendedLinksEnum type supports the following values:

Name	Value	Description
exHideLinks	0	(False) Hides the links in the chart. No links are shown in the chart 
exShowLinks	-1	(True) (By default) Shows the links in the chart. 
exShowExtendedLinks	1	Shows the extended links in the chart. The extended links are shown when two or more links starts or ends on the same bar, so they will be shown distinctly, rather than showing them one over another. This flag can be combined with the exShowLinksFront, to bring the links in front, or to put over the bar, rather than behind bars. There are some situations when the links are required to be on the background, for instance, if using the ShowLinksColor property, which require changing the color of the bars based on the links of the selected bar. 

exShowDefaultLinks	2	Shows the default links in the chart. This value is identical with the exShowLinks, excepts that this can be combined with the exShowLinksFront, to bring the links in the front. 
exShowLinksFront	16	Shows the links on the front. This flag can be combined with exShowExtendedLinks or exShowLinksFront flags.

constants ShowLinksEnum

The ShowLinksEnum type specifies the type of links that can be show differently based on the selected bars. The ShowLinksColor property specifies the color to display the links when link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. The ShowLinksWidth property specifies the width to show the links when the link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. The ShowLinksStyle property specifies the width to show the links when the link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars.

Name	Value	Description
exShowLinksStartFrom	1	Shows the links that starts from selected bars (outgoing links/bars).
exShowLinksEndTo	2	Shows the links that ends on the selected bars (incoming links/bars).
exShowUnselectedLinks	4	Shows the links that are not related to any selected bar.
exUpdateColorLinksOnly	16	Prevents applying the link's color to related bars. This flag can be combined with any of the above values when calling the set property of the ShowLinksColor property. It has no effect for get method of ShowLinksColor property. For instance, ShowLinksColor(exShowLinksEndTo + exUpdateColorLinksOnly) = RGB(255,0,0) shows in red all incoming links (the color of the incoming bars remains unchanged), Instead if the ShowLinksColor(exShowLinksEndTo) = RGB(255,0,0) the incoming links and bars are shown in red.

constants SortOnClickEnum

Specifies the action that control takes when user clicks the column's header. The SortOnClick Property specifies whether the control sorts a column when its caption is clicked.

Name	Value	Description
exNoSort	0	The column is not sorted when the user clicks the column's header.
exDefaultSort	-1	The control sorts the column when user clicks the column's header.
exUserSort	1	The control displays the sort icons, but it doesn't sort the column. The user is responsible with listing the items as being sorted. Use the ItemByPosition property to access the sorted column in their order.

constants SortOrderEnum

Specifies the column's sort order. Use the SortOrder property to specify the column's sort order.

Name	Value	Description
SortNone	0	The column is not sorted. (if the control supports sorting by multiple columns, the column is removed from the sorting columns collection)
SortAscending	1	The column is sorted ascending. (if the control supports sorting by multiple columns, the column is added to the sorting columns collection)
SortDescending	2	The column is sorted descending. (if the control supports sorting by multiple columns, the column is added to the sorting columns collection)

constants SortTypeEnum

The SortTypeEnum enumeration defines the ways how the control can sort the columns. Use the SortType property to specify how the column gets sorted. The CellValue property indicates the values being sorted.

Name	Value	Description
SortString	0	(Default) Values are sorted as strings.
SortNumeric	1	Values are sorted as numbers. Any non-numeric value is evaluated as 0.
SortDate	2	Values are sorted as dates. Group ranges are one day.
SortDateTime	3	Values are sorted as dates and times. Group ranges are one second.
SortTime	4	Values are sorted using the time part of a date and discarding the date. Group ranges are one second.
SortUserData	5	The CellData property indicates the values being sorted. Values are sorted as numbers.
SortUserDataString	6	The CellData property indicates the values being sorted. Values are sorted as strings.
exSortByValue	16	The column gets sorted by cell's value rather than cell's caption.
exSortByState	32	The column gets sorted by cell's state rather than cell's caption.
exSortByImage	48	The column gets sorted by cell's image rather than cell's caption.

constants ItemBarPropertyEnum

The ItemBarPropertyEnum type specifies a property related to a bar inside an item. Use the ItemBar property to retrieve or sets a value for bars in the item. The AllowCellValueToItemBar property allows the cells to display properties of the bars. The ShowLinksColor property specifies the color for links that starts or ends on selected bars. The ShowLinksStyle property specifies the width to show the links when the link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. The SelBarColor property specifies the color to display the selected bars.

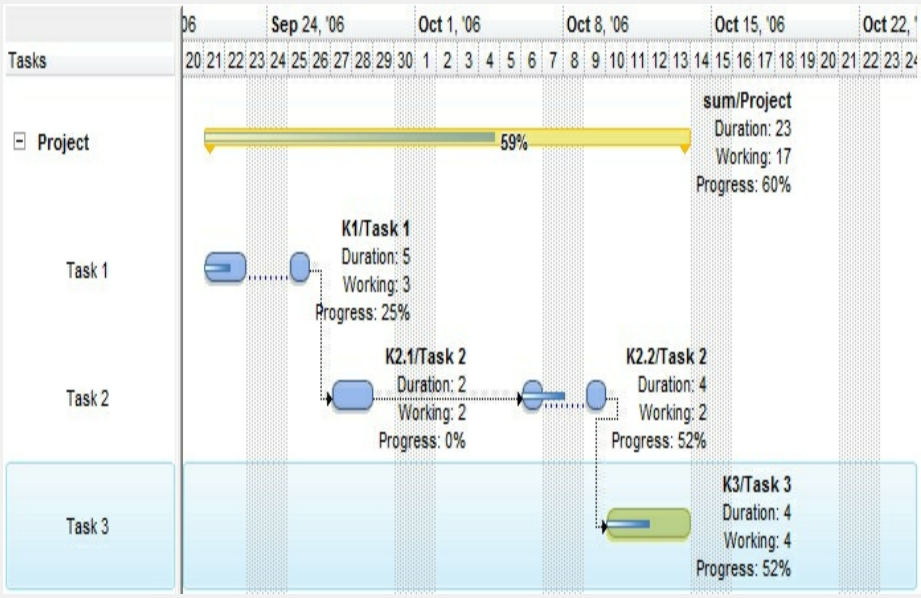
The ItemBarPropertyEnum type supports the following values:

Name	Value	Description
exBarName	0	Retrieves or sets a value that indicates the name of the bar. The name of the bar does not indicate the bar's caption as exBarCaption property. The exBarName indicates a string such as "Task", "Progress", "Split", or any other predefined type of bar using the Add method. The BarName parameter of the AddBar method indicates the name of the bar to be added to an item. Use the Add method to add new type of bars to your chart. Use the exBarCaption or exBarExtraCaption to associate a caption to be displayed on the current bar. Use the exBarToolTip to assign a tooltip to a bar. Use the exBarStart and exBarEnd/exBarEndInclusive to specify the start and end points of the bar. *(String expression)*
exBarStart	1	Retrieves or sets a value that indicates the starting date-time of the bar. The DateStart parameter of the AddBar method indicates the starting date-time of the bar being added. The exBarStart property can be a DATE or a string expression that defines the starting date-time of the bar. Use the exBarMove, exBarMoveStart or exBarDuration to move or resize programmatically the bar. Use the exBarStartPrev property to get the starting date-time before resizing the bar. You can call the AddBar method with the new coordinates, same item and key, to change at once the starting /

		ending date-time of an existing bar. Use the ShowEmptyBars property to show the bars, even if the start and end dates are identical. *(Date expression)*
exBarEnd	2	Retrieves or sets a value that indicates the ending date-time of the bar. The DateEnd parameter of the AddBar method indicates the ending date-time of the bar being added. The exBarEnd property can be a DATE or a string expression that defines the ending date-time of the bar. Use the exBarMove, exBarMoveEnd or exBarDuration to move or resize programmatically the bar. Use the exBarEndPrev property to get the ending date before resizing the bar. You can call the AddBar method with the new coordinates, same item and key, to change at once the starting / ending date-time of an existing bar. Use the ShowEmptyBars property to show the bars, even if the start and end dates are identical. You can use the exBarEndInclusive to display exBarEnd - 1 when associate a cell with a bar, using the AllowCellValueToItemBar property, so the ending point displayed on the list section is one day less. *Changing the exBarEnd value may change the exBarEndInclusive value, or reverse.* For instance, a a task bar from 1/1/2001 to 1/3/2001 shows two days, the exBarEnd displays 1/3/2001, while the exBarEndInclusive displays 1/2/2001. *(Date expression)*
		Retrieves or sets a value that indicates the caption being assigned to the bar. The Text parameter of the AddBar method indicates the caption of the bar. Use the exBarHAlignCaption and exBarVAlignCaption to specify the alignment of the bar's caption. Use the exBarExtraCaption option to specify additional labels or captions for the bar. Use the Add method to associate a note with a bar. You can use the <img> HTML tag to add icons or custom size pictures to your bar. The

exBarCaption	3	exBarShowCaption hides the bar's caption. This option supports built-in **HTML** format including the **<%=formula%>** tag. The **<%=formula%>** tag indicates the result of the giving formula. The formula supports value formatting. The **formula** supports the following keywords: • ***%0**, **%1**, **%2**, ... specifies the corresponding property of the bar, such as %0 indicates the exBarName, %1 exBarStart, %2 exBarEnd, and so on.* • ***%C0**, **%C1**, **%C2**, ... specifies the caption of the cell, or the string the cell displays in the column with the index 0, 1 2, ... The CellCaption property specifies the cell's caption. The cell's value may be different than what the cell displays as a string. For instance, let's say a cell display HTML format. The %0 returns the html format including the HTML tags, while %C0 returns the cell's content as string without HTML tags. For instance, "upper(%C1)" converts the caption of the cell with the index 1, to upper case, while "%C0 left 2" returns the leftmost two characters on the cell with the index 0.* • ***%CD0**, **%CD1**, **%CD2**, ... specifies the cell's extra data in the column with the index 0, 1 2, ... The CellData property associates any extra/user data to a cell. For instance, "%CD0 = `your user data`" specifies all cells whose CellData property is `your user data`, on the column with the index 0.* • ***%CS0**, **%CS1**, **%CS2**, ... specifies the cell's state in the column with the index 0, 1 2, ... The CellState property specifies the cell's state, and so it indicates whether the cell is checked or un-checked. For instance, "%CS0" defines all checked items on the column with the index 0, or "not %CS1" defines all un-checked items in the column with the index 1.*

*For instance the Items.ItemBar(exBarCaption) = "<%=%9 + '/' + %C0%>
Duration: <%=(%2-%1)%>
Working: <%=%258%>
Progress: <%=round(100*%12)+'%'%>" defines the bar's caption as in the following screen shot:*



The <%=formula%> tag allows you to specify custom HTML format for caption, tooltip or bar's legend based on the properties(%) and the cells in the owner item (%C). Newer versions, allow you to specify additional captions to the same bar using the exBarExtraCaption option.

This property/method supports predefined constants and operators/functions as described here.

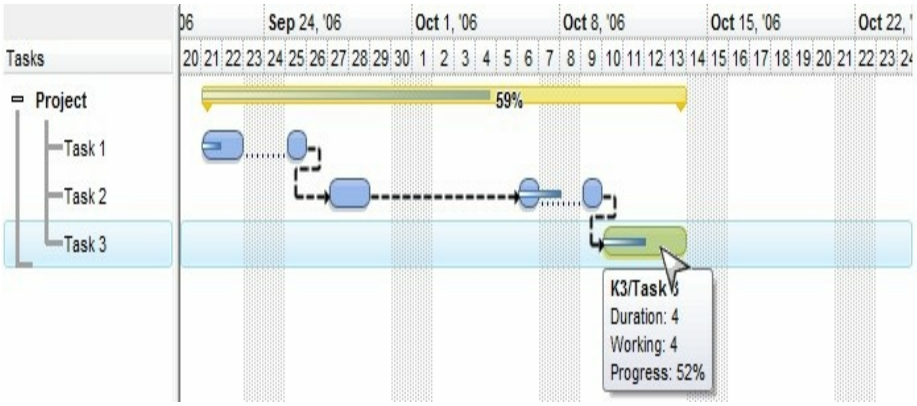
(HTML String expression)

Retrieves or sets a value that indicates the horizontal alignment / clipping of the caption inside / outside the bar. Use the exBarHAlignCaption property to align horizontally the caption being displayed between exBarStart and exBarEnd. The exBarCaptionHOffset / exBarCaptionVOffset specifies the horizontal/vertical offset of the bar's caption relative to its default position.

exBarHAlignCaption	4	Use the exBarHAlignCaption property to clip the bar's caption to bar's client area as described bellow. If the exBarHAlignCaption property is: • 0,1 or 2 the caption is not clipped and it is aligned to the left, center or right side of the bar (***no clip***). • 3, 4 or 5 the caption of the bar gets clipped to the bar's client area, else the caption is aligned to the left, center or right side of the bar (***clip, inside***). • 6, 7 or 8 the bar's caption is hidden if its size is less or equal with MinUnitWidth property, else if it does not fit the bar's client are, gets clipped or else fully aligned to left, center or right side of the bar. (***hide if min, clip if not fit, inside***). • 9, 10 or 11 the bar's caption is hidden if it does not fit entirely into the bar's client area, else it is fully displayed aligned to the left, center or right side of the bar. (***hide if not fit, no clip, inside***). • 12, 13 or 13 the bar's caption is displayed inside of the bar's client area if it fits entirely, else it is displayed outside of the bar aligned to the left, center or right. (***no clip, inside, outside***). • 16, 17, 18, the bar's caption is displayed outside of the bar to the left or to the right. (***no clip, outside***). Also, the field supports the following flag (OR combination with any other value): • 32 (0x20), which indicates that the bar's caption fits the bar and view (that bar's caption is aligned relative to the horizontal-intersection of the bar with the **view**) (for instance, 33 {number}, (1 + 32) the item-bar's caption is always shown within the center (horizontally) of item-bar intersected with the current view)

		By default, the exBarHAlignCaption is CenterAlignment (1, no clip, center) *(AlignmentEnum expression)*
exBarVAlignCaption	5	Retrieves or sets a value that indicates the vertical alignment of the caption inside the bar. Use the exBarHAlignCaption property to align vertically the caption being displayed between exBarStart and exBarEnd. If the exBarVAlignCaption property includes the VAlignmentEnum.exVOutside the caption is displayed outside of the bar at the top or bottom side of the bar. For instance, if the exBarVAlignCaption property is VAlignmentEnum.exTop OR VAlignmentEnum.exVOutside, the caption is displayed outside of the bar in the top side of the bar. If the exBarVAlignCaption property is VAlignmentEnum.exBottom OR VAlignmentEnum.exVOutside, the caption is displayed outside of the bar in the bottom side of the bar. By default, the exBarVAlignCaption is exMiddle. *(VAlignmentEnum expression)*
		Retrieves or sets a value that indicates the tooltip being shown when the cursor hovers the bar. The property supports built-in HTML format. Use the exBarToolTip property to assign a tooltip to a bar or to a text in the chart's area. Use the ShowToolTip method to show a tooltip at runtime for different parts of the chart. The Tooltip(0, -2, , , , ,) event occurs once the bar's tooltip (exBarToolTip) is about to be shown (-2 if the mouse pointer hovers the bars of the chart). This option supports built-in **HTML** format including the **<%=formula%>** tag. The **<%=formula%>** tag indicates the result of the giving formula. The formula supports value formatting.

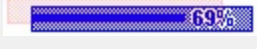
exBarToolTip	6	The **formula** supports the following keywords: • ***%0**, **%1**, **%2**, ... specifies the corresponding property of the bar, such as %0 indicates the exBarName, %1 exBarStart, %2 exBarEnd, and so on.* • ***%C0**, **%C1**, **%C2**, ... specifies the caption of the cell, or the string the cell displays in the column with the index 0, 1 2, ... The CellCaption property specifies the cell's caption. The cell's value may be different than what the cell displays as a string. For instance, let's say a cell display HTML format. The %0 returns the html format including the HTML tags, while %C0 returns the cell's content as string without HTML tags. For instance, "upper(%C1)" converts the caption of the cell with the index 1, to upper case, while "%C0 left 2" returns the leftmost two characters on the cell with the index 0.* • ***%CD0**, **%CD1**, **%CD2**, ... specifies the cell's extra data in the column with the index 0, 1 2, ... The CellData property associates any extra/user data to a cell. For instance, "%CD0 = `your user data`" specifies all cells whose CellData property is `your user data`, on the column with the index 0.* • ***%CS0**, **%CS1**, **%CS2**, ... specifies the cell's state in the column with the index 0, 1 2, ... The CellState property specifies the cell's state, and so it indicates whether the cell is checked or un-checked. For instance, "%CS0" defines all checked items on the column with the index 0, or "not %CS1" defines all un-checked items in the column with the index 1.* *For instance the Items.ItemBar(exBarToolTip) = "<%=%9 + '/' + %C0%> Duration: <%=(%2-%1)%> Working: <%=%258%> Progress: <%=round(100*%12)+'%'%>" defines the bar's tooltip to show as in the following screen shot:*

		 The <%=formula%> tag allows you to specify custom HTML format for caption or bar's tooltip based on the properties(%) and the cells in the owner item (%C). *(HTML String expression)*
exBarBackColor	7	Retrieves or sets a value that indicates the background color for the area being occupied by the bar. Color expression. This option has effect only if the exBarBackColor property is not zero. The last 7 bits in the high significant byte of the color indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part. The exSummaryBarBackColor property specifies the background color for the portion occupied by the bar while it is hosted by a summary bar, while the exBarColor property specifies the bar's color. *(Color expression)*
exBarForeColor	8	Retrieves or sets a value that indicates the foreground color for the caption of the bar. Color expression. This option has effect only if the exBarBackColor property is not zero. *(Color expression)*
		Specifies key of the bar. The Key parameter of the AddBar method indicates the key of the bar.

exBarKey	9	*(Variant expression)*
exBarCanResize	10	Specifies whether the user can resize the bar. Use the BarsAllowSizing property to specify whether the control supports moving or resizing the bars. • If the exBarCanResize is 0/False, the bar can not be resized. • If the exBarCanResize is -1/True, the bar can be resized on both sides. • If the exBarCanResize is 1, the bar can be resized on left side, and can not be resized on right. • If the exBarCanResize is 2, the bar can be resized on right side, and can not be resized on left. The exBarSelectable specifies whether a bar is fixed to its position, in other words if it can be selected or not. *(Boolean/Long expression)*
exBarCanMove	11	Specifies whether the user can move the bar. Use the BarsAllowSizing property to specify whether the control supports moving or resizing the bars. Use the exBarCanMoveToAnother option to specify whether the user can move a bar from one item to another by drag and drop. The exBarSelectable specifies whether the bar can be selected at runtime. *(Boolean expression)*
		 Specifies the percent from the original bar where the progress bar is displayed. This float value should be between 0 and 1 (1 means 100%). You can use also the exBarpercent100 to specify long expression value from 0 to 100. Use the Add("A%B") to add a combination of two bars, so the exBarPercent value specifies the percent from the bar A to be displayed

exBarPercent	12	as bar B. For instance, the *Add("Task%Progress")* adds a combination of Task and Progress bars, so the Task shape is displayed on the full bar, and the Progress shape is displayed only on the portion determined by the exBarPercent value. When you resize the original bar (A), the inside bar (B) is shown proportionally. Use the exBarShowPercentCaption option to show the percent value as caption on the bar. Use the exBarPercentCaptionFormat property to define the format of the percent value being displayed as text. Use the exBarAlignmentPercentCaption property to specify the alignment of the percent on the bar. The BarResize event is fired when the exBarPercent value is changed. You can use the exBarPercent100 option to work with integer values from 0 to 100 for specifying the bar's percent. *The 0 value corresponds to the exBarStart, while 1 corresponds to the exBarEnd, so the formula Items.ItemBar(exBarStart) + (Items.ItemBar(exBarEnd)- Items.ItemBar(exBarStart))*Items.ItemBar(exBarPer gives exactly the date time where progress bar is in the chart.* *(Float expression, between 0 and 1, by default it is 0)*
exBarPercentCaptionFormat	13	Specifies the HTML format to be displayed as percent. The percent is displayed on the bar only if the exBarShowPercentCaption option is True. By default, the exBarPercentCaptionFormat property is "%p%" where the %p is the value of the percent (exBarPercent property), and it displays the percent as **15%**, where exBarPercent is 0.15. The indicates that the text is bolded. *(String expression)*
		Specifies whether the percent is displayed as caption on the bar. By default, the exBarShowPercentCaption property is False, which

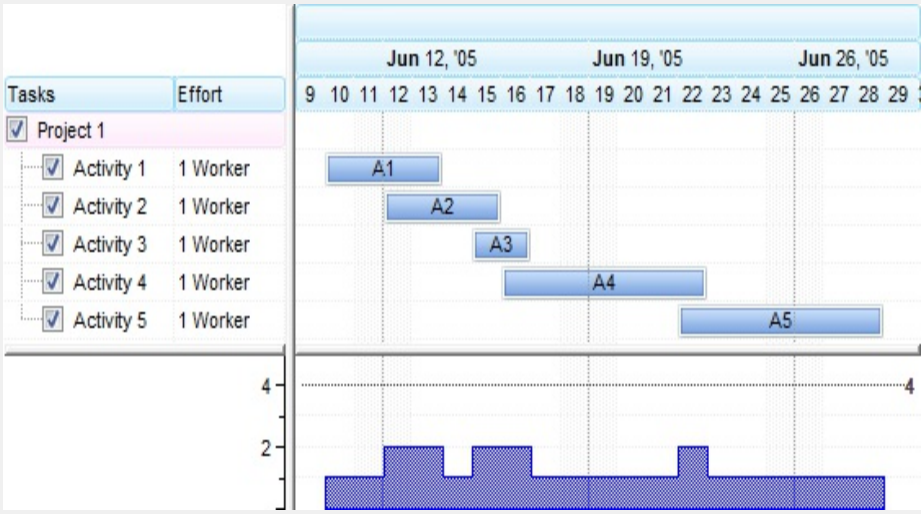
exBarShowPercentCaption	14	means that the percent value is not shown. Use the exBarPercent property to specify the value of the percent. Use the exBarPercentCaptionFormat property to define the format of the percent being displayed on the bar. Use the exBarAlignPercentCaption property to indicates the alignment of the percent in the bar. *(Boolean expression)*
exBarAlignPercentCaption	15	Specifies the horizontal alignment of the percent caption on the bar. Use the exBarVAlignPercent option to specify the vertical alignment of the percent bar relative to the owner bar. If the exBarAlignPercentCaption property is: • 0, 1 or 2 the percent caption is not clipped and it is aligned to the left, center or right side of the progress bar (**no clip**) • 3, 4 or 5 the percent caption of the progress bar gets clipped to the progress bar's client area, else the percent caption is aligned to the left, center or right side of the progress bar (***clip, inside***). • 6, 7 or 8 the percent caption of the progress bar is hidden if its size is less or equal with MinUnitWidth property, else if it does not fit the progress bar's client area, gets clipped or else fully aligned to left, center or right side of the progress bar. (***hide if min, clip if not fit, inside***). • 9, 10 or 11 the percent caption of the progress bar is hidden if it does not fit entirely into the progress bar's client area, else it is fully displayed aligned to the left, center or right side of the progress bar. (***hide if not fit, no clip, inside***). • 12, 13 or 14 the percent caption of the progress bar is displayed inside of the progress bar's client area if it fits entirely, else it is displayed outside of the progress bar aligned to the left, center or right. (***no clip,***

		inside, outside). • 16, 17 or 18, the percent caption of the progress bar is displayed outside of the progress bar to the left or to the right (***no clip, outside***). By default, the exBarAlignPercentCaption is RightAlignment (2, no clip, right). (AlignmentEnum expression)
exBarCanResizePercent	16	Specifies whether the user can resize the percent at runtime. By default, the exBarCanResizePercent is True. Here's how the user can resize the percent value at runtime. Move the mouse to the last portion of the percent, so the *Percent cursor* is shown. Click and drag the bar to a new position, so the exBarPercent value is defined proportionally by the position of the cursor in the original bar. The BarResize event is fired when the user changes the percent value at runtime. *(Boolean expression)*
exBarData	17	Associates an extra data to a bar. Use this property to assign your extra data to any bar in the item. *(Variant expression)*
exBarOffset	18	 Specifies the vertical offset where the bar is shown. By default, this property is 0 and the bar is shown in the center. Use this property to show up or down the bar. Use the OverlaidType property to specify how two or more bars that cross over are displayed. *(Long expression)*
		 Specifies the percent of the transparency to display the bar, or to show or hide a bar and its links. By default, this property is 0, which means that the bar is opaque. If the property

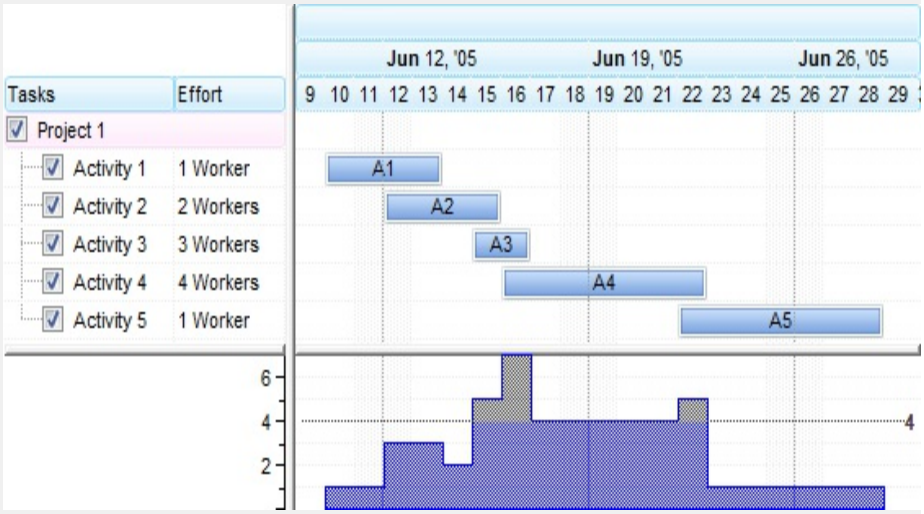
exBarTransparent	19	is 50, the bar is shown semi-transparent. Use the ShowTransparentBars property to draw all bars using a semi- transparent color. Use the OverlaidType property to specify how two or more bars that cross over are displayed. If the exBarTransparent property is 100, the bar is hidden, along with its links if any. *(Long expression between 0-opaque, 50-semi-transparent100-hidden)*
exBarKeepWorkingCount	20	By default, the exBarKeepWorkingCount property is False. Specifies a value that indicates whether the bar keeps constant the working units, while the user moves the bar to a new position. The NonworkingDays property specifies the non-working days. Use the AddNonworkingDate property to add custom non-working days. Use the NonworkingHours property to specify the non-working hours. The exBarWorkingCount option specifies the number of working units being unchanged. This option is ignored for summary bars or exBarTreatAsNonworking bars. We are not recommending using the ShowEmptyBars property on a non-zero value, if using bars with the ItemBar(exBarKeepWorkingCount) property on True. *(Boolean expression)*
		(exBarEffort/21) By default, the exBarEffort is 1 (double expression). Specifies the effort to execute an unit in the task. This property has effect only when the bar is represented in the chart's histogram. Use the HistogramVisible property to specify whether the control shows the chart's histogram. Changes the **HistogramPattern or/and HistogramColor** property, else no bars will be shown in the histogram. The representation of the exBarEffort value depends on the HistogramType property as follow:

- ***exHistOverload***, the exBarEffort value represents the effort to execute an unit in the bar. For instance, if the bars display activities the exBarEffort value may represent the number of workers for each activity so the overload histogram displays the total number of workers on the activity.

The following screen shot shows the exHistOverload histogram when exBarEffort property is 1 (by default):



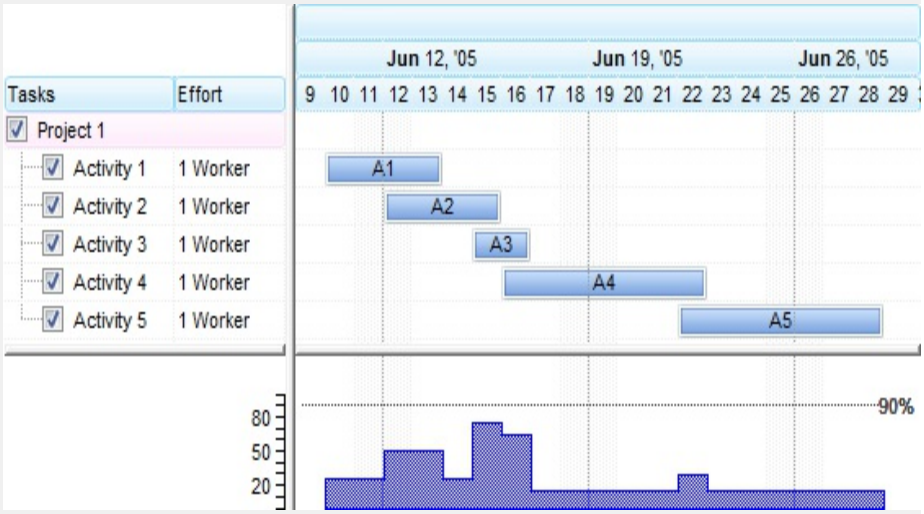
The following screen shot shows the exHistOverload histogram when exBarEffort property is different for bars as seen in the columns section:



- ***exHistOverAllocation***, the exBarEffort value defines the workload to show in the

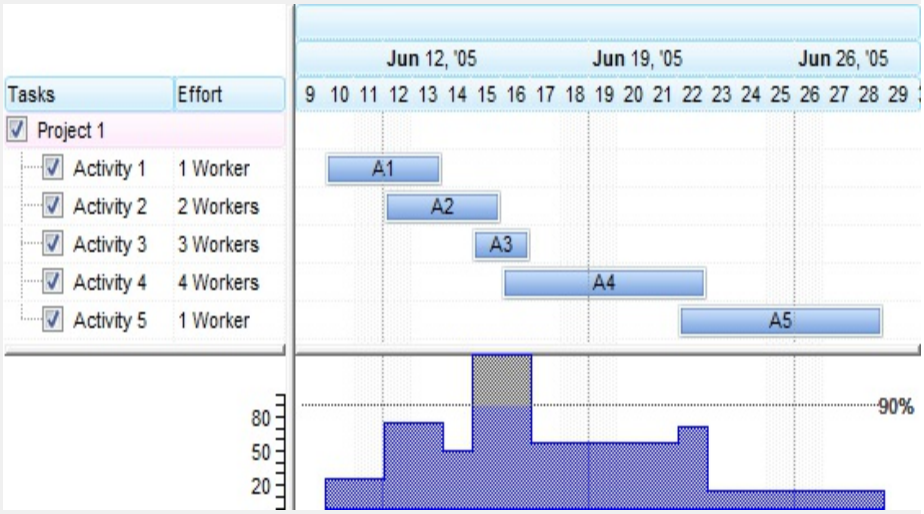
exHistOverAllocation histogram. The work-load for a task is computed as exBarEffort / length of the bar. The work-load for the task is the work effor / task duration. (i.e. If item.exBarEffort = 1 and gantt bar length is 10 days, then the work-load = 0.1 or 10%). The histogram- graph shows the sum of the work-loads (the work-load of each task item is added, unit by unit).

The following screen shot shows the exHistOverallocation histogram when exBarEffort property is 1 (by default):



The following screen shot shows the exHistOverallocation histogram when exBarEffort property is different for bars as seen in the columns section:

exBarEffort 21



Starting from version 14.0, the exBarEffort could be:

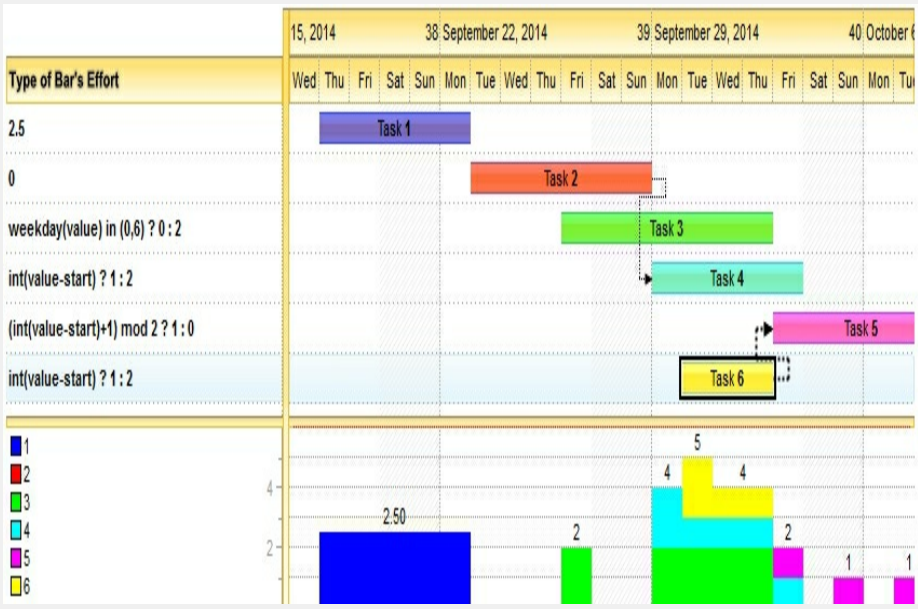
- a numeric value which it is applied for all units in the task
- a string that indicates the expression/formula to get the effort of the bar to be represented on the chart's histogram. The **value** keyword indicates the date-time being queried, the **start** and **end** keywords specify the starting and ending points of the bar as indicated by exBarStart and exBarEnd fields in the ItemBar property. For instance, the exBarEffort on "weekday(value) in (0,6) ? 0 : 2", means that that effort to do the job is 2 for any day in the task, excepts the Sundays(0) and Saturdays(6) (weekend)

Here's a few samples of using the exBarEffort:

- 2.5 means that effort to do the job is 2.5 for any day in the task
- 0 means that the task has no representation on the chart's histogram
- "weekday(value) in (0,6) ? 0 : 2", means that that effort to do the job is 2 for any day, excepts the Sundays(0) and Saturdays(6)
- "weekday(value) = 1 ? 2 : 1" indicates that the effort to do the job is 2 for Mondays(1), else 1
- "month (value) = 7 ? 1 : 0", indicates that the effort to do the job is 1 for any day in July, and 0 for any other
- "(month(value)=month(value+1)) ? 1 : 0", indicates that the effort to do the job is 1 for any day, excepts the last day in the month.
- "int(value-start) ? 1 : 2" indicates that the effort to do the job is 2 for the first day in the task, and 1 for the others.
- "(int(value-start) and int(end-value) != 0) ? 1 : 2" indicates that the effort to do the job is 2 for the first and last days in the task, and 1 for the others.
- "int(value-start) in (0,1,2) ? 1 : 2" indicates that the effort to do the job is 1 for the first three (0,1,2) days in the task, and 1 for the others.
- "(int(value-start)+1) mod 2 ? 1 : 0" indicates

that the effort to do the job is 1 for the first day, 0 for the second day, 1 for the third day, 0 for the forth day, and so on.

Here's a screen shot of a few exBarEffort samples:



Here's what you can represent with the exBarEffort property:

- Click here to watch the movie on how you can use expressions on exBarEffort.

The supported keywords are:

- **value** which indicates the date-time being queried
- **start** and **end** specify the starting and ending points of the bar as indicated by exBarStart and exBarEnd fields in the ItemBar property.

This property/method supports predefined constants and operators/functions as described here.

(Float expression [or String expression (starting with version 14.0)])

Specifies the minimum value for the starting date of the bar. Use this value to limit bar to move in a specified range. Use the exBarMinDuration and

exBarMinStart	22	exBarMaxDuration properties to specify the limits for the bar's duration. *(Date expression)*
exBarMaxStart	23	Specifies the maximum value for the starting date of the bar. Use this value to limit bar to move in a specified range. Use the exBarMinDuration and exBarMaxDuration properties to specify the limits for the bar's duration. *(Date expression)*
exBarMinEnd	24	Specifies the minimum value for the ending date of the bar. Use this value to limit bar to move in a specified range. Use the exBarMinDuration and exBarMaxDuration properties to specify the limits for the bar's duration. *(Date expression)*
exBarMaxEnd	25	Specifies the maximum value for the ending date of the bar. Use this value to limit bar to move in a specified range. Use the exBarMinDuration and exBarMaxDuration properties to specify the limits for the bar's duration. *(Date expression)*
exBarShowRange	26	Indicates whether the bar shows its range where it can be moved or resized. It indicates a PatternEnum expression that specifies the way to mark the range of the bar, or a skin identifier to be used for showing the range. *(Boolean expression)*
exBarShowRangeTransparent	27	Specifies the percent of the transparency to display the range of the bar. By default this property is 0. *(Long expression, from 0 to 100, where 100 means fully transparent)*

exBarCanMoveToAnother	28	By default, the bar can NOT be moved from an item to another. Specifies whether the bar can be moved to another item by drag and drop. Use the exBarPercent property to change the parent item of the bar by code. In order to move a bar from one item to another using the drag and drop operations, this option MUST be set on True. The control fires the BarParentChange event just before moving the bar to another item. Use this event to control the items where your bar can be moved. A bar can be moved to another item, ONLY if the second item does not contain a bar with the same key. The exBarKey property specifies the key of the bar. The AutoDrag property indicates what the control does when the user clicks an item and starts dragging it. A Bar can be moved from an item to another, if the new parent item contains no bars with the same key as the dragging one. *(Boolean expression)*
exBarSelectable	29	Specifies whether the bar can be selected. The exBarCanMove specifies whether a bar can be moved at runtime. The exBarCanResize specifies whether a bar can be resized at runtime. The exBarSelectable specifies whether a bar is fixed to its position. Use a non-selectable bar to add custom non-selectable entries to your chart. *(Boolean expression)*
exBarCanStartLink	30	Specifies whether a link can start from this bar. By default, the exBarCanStartLink is True. The exBarCanStartLink option has effect only if the exBarCanBeLinked option is True. You can control enabling or disabling links at runtime using the AllowLink event. *(Boolean expression)*

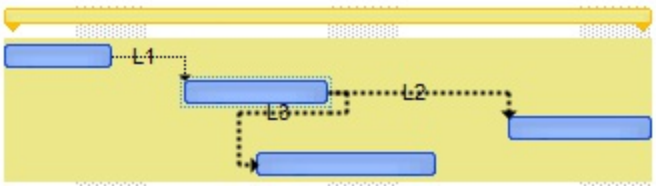
exBarCanEndLink	31	Specifies whether a link can end on this bar. . By default, the exBarCanEndLink is True. The exBarCanEndLink option has effect only if the exBarCanBeLinked option is True. You can control enabling or disabling links at runtime using the AllowLink event. *(Boolean expression)*
exBarCanBeLinked	32	Specifies whether the bar can be linked. . By default, the exBarCanBeLinked is True. You can control enabling or disabling links at runtime using the AllowLink event. (Boolean expression)
exBarColor	33	Specifies the color for a particular bar. If used, it replaces the bar's type color. By default, the exBarColor is 0, which means that the default bar's color is used. The Color property defines the default's bar color. The Color property defines the color for all bars of the same type. Use the exBarColor to change the color for particular bars. As usual, this option may indicates a skin object to display the bar. If the EBN identifier (last 7 bits in the high significant byte of the color) is not specified(0), but the Color property indicates an EBN object, the exBarColor specifies the color to apply to the EBN object. The exBarOverviewColor option may be used to specify the color within the overview part of the control. The HistogramCumulativeOriginalColorBars property indicates whether the bars that generate the histogram change their original color. If available, you can change the bar's pattern using the exBarPattern option. The ItemBar(exBarHistLegend) property specifies the description to show within the histogram's legend for the bar in the control's histogram (exKeepOriginalColor only). *(Color expression)*
		Specifies the item's background color for child bars owned by the summary bar. The last 7 bits in the high significant byte of the color indicates the

identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part. The DefineSummaryBars property defines bars that belongs to a summary bar.

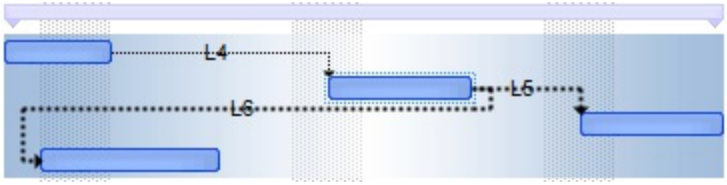
By default, the exSummaryBarBackColor property is 0, which means that it has no effect.

If the exSummaryBarBackColor property is set to a non- zero value it indicates the background color for the portion of the summary bar as seen in the following screen shot:

exSummaryBarBackColor 34



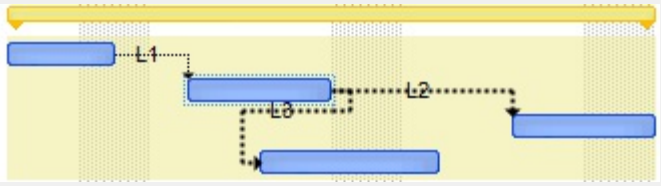
The following screen shot shows the bars that belongs to a summary bars using EBN colors and semi-transparent (exSummaryBarBackColorTransparent = 50):



The exBarBackColor property specifies the background color for the portion occupied by the bar, while the exBarColor property specifies the bar's color.

(Color expression)

Specifies the percent of the transparency to display the background of the bars that belongs to a summary bar. By default, the exSummaryBarBackColorTransparent property is 0, which means opaque since 100 means fully

		transparent, and 50 means semi-transparent. The exSummaryBarBackColorTransparent property has effect only if the exSummaryBarBackColor property is not 0.
exSummaryBarBackColorTransparent	35	The following screen shot displays the bars that belongs to a summary bar using a semi-transparent color (so the non-working portion is visible)  *(Long expression between 0-opaque, 100-transparent)*
exBarMinDuration	36	Specifies the minimum duration of the bar in days . By default, the exBarMinDuration property is 0, which means that there is no lower limit, or the length of the bar can be any value. If the exBarMinDuration property is not 0, the duration of the bar must be greater than specified value, or in other words, the bar will not be shown with its duration less than exBarMinDuration value. Use the exBarDuration property to change the bar's duration. (Float expression)
exBarMaxDuration	37	Specifies the maximum duration of the bar in days. By default, the exBarMaxDuration property is -1, which means that there is no upper limit, or the length of the bar can be any value. If the exBarMaxDuration property is not -1, the duration of the bar must be less than specified value, or in other words, the bar will not be shown with its duration greater than exBarMaxDuration value. Use the exBarDuration property to change the bar's duration. *(Float expression)*
		Specifies whether the bar is treated as non-working part of the item. By default, the

exBarTreatAsNonworking	38	exBarTreatAsNonworking is False. This option has effect only if: • AllowNonworkingBars property is True. • ItemNonworkingUnits property is not empty, and points to a valid expression. The ItemNonworkingUnits property indicates a repetitive expression to determine the parts of the item being non-working. In other words, the exBarTreatAsNonworking bars are treated as regular bars if the AllowNonworkingBars property is False, or if ItemNonworkingUnits property is empty *(Boolean expression)*
exBarPercentColor	39	Specifies the color to show inside percent bar. The option is valid for bars that displays inside a percent bar. A bar can display a percent bar if it was creates using the Chart.Bars.Add("A%B") syntax. For instance Chart.Bars.Add("Task%Progress") adds a task bar that displays inside a percent bar. *(Color expression)*
exBarNonWorkingColor	40	Specifies the color to show non-working parts of the bar. A bar may show different shape, pattern for non-working parts of the bar if it was previously created using the Bars.Add("A:B"). For instance the Bars.Add("Task:Split") adds a a bar that displays the split if it covers a non-working part. *(Color expression)*
		Specifies the color to show the bar in the overview area. The OverviewVisible property specifies whether the control displays the overview/layout map of bars within the chart. The color to specify the bar in the overview area is determined as follows:

exBarOverviewColor	41	• If ItemBar(exBarOverviewColor) property is not 0, the exBarOverviewColor indicates the color to show the bar in the overview area, else • If OverviewColor property is not 0, the OverviewColor property indicates the color to show the bar in the overview area, else • If the ItemBar(exBarColor) is not 0, the exBarColor indicates the color to show the bar in the overview area, else • The Color property of the Bar indicates the color to show the bar in the overview part of the control. (The bar is represented into the control's overview only if its determined color is not -1) *(Color expression)*
exBarPattern	42	By default the exBarPattern option is empty. If the exBarPattern property is empty, the option is ignored. Use the exBarPattern to specify a different pattern to be displayed on the bar in the chart area. The Pattern property of the Bar specifies the pattern to be applied for all bars of the same type. For instance, includes the exPatternFrameShadow in the bar's pattern to show a shadow around the bar. *(PatternEnum expression)*
exBarVAlignPercent	43	Specifies the vertical alignment of the percent bar relative to the owner bar. Use the exBarAlignPercentCaption option to align horizontally the caption inside the percent bar. By default, the exBarVAlignCaption is exMiddle. *(VAlignmentEnum expression)*
		Use the exBarExtraCaption property to assign multiple captions to a bar at once. Retrieves or sets

a collection of extra captions being assigned to the bar. If a single extra caption is being added, you can use the caption, or if multiple you have to build a safe array of strings. By default, any extra caption is added to the center of the bar. Use the exBarExtraCaptionHAlign to specify the horizontal alignment of the extra caption being added. Use the exBarExtraCaptionHOffset to specify the horizontal offset to move the extra caption relative to its default position. Use the exBarExtraCaptionVAlign to specify the vertical alignment of the extra caption being added. Use the exBarExtraCaptionVOffset to specify the vertical offset to move the extra caption relative to its default position. The /NET assembly provides, the *get_BarExtraCaption and set_BarExtraCaption* properties to get and set the extra captions. Use the Add method to associate a note with a bar. You can use the <img> HTML tag to add icons or custom size pictures to your bar. The exBarShowExtraCaption hides the bar's extra caption.

This option supports built-in**HTML** format including the **<%=formula%>** tag. The **<%=formula%>** tag indicates the result of the giving formula. The formula supports value formatting. Inside the formula the %0, %1, ... indicates the value of corresponding property of the bar, such as %0 specifies the exBarName, %1 exBarStart, and so on. The **%C0, %C1, ...** indicates the cell's values. A bar belongs to an item which displays a cell for each column. The %CIndex indicates the cell on the column with the specified index. For instance, the %C0 indicates the cell on the first column, %C1 specifies the cell in the second column, and so on.

*For instance the Items.ItemBar(exBarCaption) = "Duration of <%=%9 + ' of ' + %C0%> is <%=(%2-%1)%> days" specifies that the bar's caption shows the duration of the bar such as : "Duration of **K1 of Task1** is 3 days." where the %9 indicates the exBarKey, while %2 is exBarEnd and %1 is exBarStart. Using the **<%=formula%>** html*

TAG, you will be able to format the bar's caption to display its content based on the current properties of the bar, without having to redefine the caption once a bar is updated.

The following VB sample adds a single extra caption in the right side of the bar:

```
With G2antt1.Items
    .ItemBar(.FocusItem, .FirstItemBar(.FocusItem),
exBarExtraCaption) = "right"
    .ItemBar(.FocusItem, .FirstItemBar(.FocusItem),
exBarExtraCaptionHAlign) = RightAlignment Or
exHOutside
End With
```

The following VB sample adds two extra captions, one in the left side, and one to the right side:

```
With G2antt1.Items
    .ItemBar(.FocusItem, .FirstItemBar(.FocusItem),
exBarExtraCaption) = Array("left", "right")
    .ItemBar(.FocusItem, .FirstItemBar(.FocusItem),
exBarExtraCaptionHAlign) = Array(LeftAlignment
Or exHOutside, RightAlignment Or exHOutside)
End With
```

When retrieving the exBarExtraCaption returns the extra caption, if there is only, one, else it returns a safe arrea (collection) of string that indicates the extra captions of the bar.

The following VB/NET sample adds a single extra caption in the right side of the bar:

exBarExtraCaption 44

```
With Exg2antt1.Items
    .set_BarExtraCaption(.FocusItem,
.get_FirstItemBar(.FocusItem), "right")
    .set_BarExtraCaptionHAlign(.FocusItem,
.get_FirstItemBar(.FocusItem),
```

```
exontrol.EXG2ANTTLib.AlignmentEnum.RightAlign
 Or
exontrol.EXG2ANTTLib.AlignmentEnum.exHOutside

End With
```

The following VB/NET sample adds two extra captions, one in the left side, and one to the right side:

```
With Exg2antt1.Items
    .set_BarExtraCaption(.FocusItem,
.get_FirstItemBar(.FocusItem), New String() {"left",
"right"})
    .set_BarExtraCaptionHAlign(.FocusItem,
.get_FirstItemBar(.FocusItem), _
    New exontrol.EXG2ANTTLib.AlignmentEnum()
{exontrol.EXG2ANTTLib.AlignmentEnum.LeftAlignm
 Or
exontrol.EXG2ANTTLib.AlignmentEnum.exHOutside
 _

exontrol.EXG2ANTTLib.AlignmentEnum.RightAlign
 Or
exontrol.EXG2ANTTLib.AlignmentEnum.exHOutside

End With
```

The following C# sample adds a single extra caption in the right side of the bar:

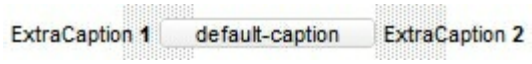
```
exontrol.EXG2ANTTLib.Items items =
exg2antt1.Items;
items.set_BarExtraCaption(items.FocusItem,
items.get_FirstItemBar(items.FocusItem), "right");
items.set_BarExtraCaptionHAlign(items.FocusItem,
items.get_FirstItemBar(items.FocusItem),
exontrol.EXG2ANTTLib.AlignmentEnum.RightAlign
```

```
 |
exontrol.EXG2ANTTLib.AlignmentEnum.exHOutside
```

The following C# sample adds two extra captions, one in the left side, and one to the right side:

```
exontrol.EXG2ANTTLib.Items items =
exg2antt1.Items;
items.set_BarExtraCaption(items.FocusItem,
items.get_FirstItemBar(items.FocusItem),
   new string[2] { "left", "right" });
items.set_BarExtraCaptionHAlign(items.FocusItem,
items.get_FirstItemBar(items.FocusItem),
   new exontrol.EXG2ANTTLib.AlignmentEnum[2]
{
exontrol.EXG2ANTTLib.AlignmentEnum.LeftAlignm
 |
exontrol.EXG2ANTTLib.AlignmentEnum.exHOutside



exontrol.EXG2ANTTLib.AlignmentEnum.RightAlign
 |
exontrol.EXG2ANTTLib.AlignmentEnum.exHOutside
 });
```

The following picture shows a bar with its default caption and two other additional captions.



(HTML string expression or safe array of HTML string expressions)

Specifies the horizontal alignment for each extra caption assigned to the bar.

If the exBarExtraCaptionHAlign property is:

- 0,1 or 2 the caption is **not** clipped and it is aligned to the left, center or right side of the

bar (***no clip***).

- 3, 4 or 5 the caption of the bar gets clipped to the bar's client area, else the caption is aligned to the left, center or right side of the bar (***clip, inside***).
- 6, 7 or 8 the bar's caption is hidden if its size is less or equal with MinUnitWidth property, else if it does not fit the bar's client are, gets clipped or else fully aligned to left, center or right side of the bar. (***hide if min, clip if not fit, inside***).
- 9, 10 or 11 the bar's caption is hidden if it does not fit entirely into the bar's client area, else it is fully displayed aligned to the left, center or right side of the bar. (***hide if not fit, no clip, inside***).
- 12, 13 or 13 the bar's caption is displayed inside of the bar's client area if it fits entirely, else it is displayed outside of the bar aligned to the left, center or right. (***no clip, inside, outside***).
- 16, 17, 18, the bar's caption is displayed outside of the bar to the left or to the right. (***no clip, outside***).

Also, the field supports the following flag (OR combination with any other value):

- 32 (0x20), which indicates that the bar's caption fits the bar and view (that bar's caption is aligned relative to the horizontal-intersection of the bar with the **view**) (for instance, 33 {number}, (1 + 32) the item-bar's caption is always shown within the center (horizontally) of item-bar intersected with the current view)

By default, the exBarExtraCaptionHAlign is CenterAlignment (1, no clip, center)

The following VB sample adds a single extra caption in the right side of the bar:

```
With G2antt1.Items
```

```
    .ItemBar(.FocusItem, .FirstItemBar(.FocusItem),
exBarExtraCaption) = "right"
    .ItemBar(.FocusItem, .FirstItemBar(.FocusItem),
exBarExtraCaptionHAlign) = RightAlignment Or
exHOutside
    .ItemBar(.FocusItem, .FirstItemBar(.FocusItem),
exBarExtraCaptionVAlign) = exBottom Or
exVOutside
End With
```

The following VB sample adds two extra captions, one in the left side, and one to the right side:

```
With G2antt1.Items
    .ItemBar(.FocusItem, .FirstItemBar(.FocusItem),
exBarExtraCaption) = Array("left", "right")
    .ItemBar(.FocusItem, .FirstItemBar(.FocusItem),
exBarExtraCaptionHAlign) = Array(LeftAlignment
Or exHOutside, RightAlignment Or exHOutside)
    .ItemBar(.FocusItem, .FirstItemBar(.FocusItem),
exBarExtraCaptionVAlign) = Array(exTop Or
exVOutside, exBottom Or exVOutside)
End With
```

When retrieving the exBarExtraCaption returns the extra caption, if there is only one, else it returns a safe array (collection) of string that indicates the extra captions of the bar.

The following VB/NET sample adds a single extra caption in the right side of the bar:

```
With Exg2antt1.Items
    .set_BarExtraCaption(.FocusItem,
.get_FirstItemBar(.FocusItem), "right")
    .set_BarExtraCaptionHAlign(.FocusItem,
.get_FirstItemBar(.FocusItem),
exontrol.EXG2ANTTLib.AlignmentEnum.RightAlign
 Or
```

exBarExtraCaptionHAlign	45	exontrol.EXG2ANTTLib.AlignmentEnum.exHOutside .set_BarExtraCaptionVAlign(.FocusItem, .get_FirstItemBar(.FocusItem), exontrol.EXG2ANTTLib.VAlignmentEnum.exBottom Or exontrol.EXG2ANTTLib.VAlignmentEnum.exVOutsid End With The following VB/NET sample adds two extra captions, one in the left side, and one to the right side: With Exg2antt1.Items .set_BarExtraCaption(.FocusItem, .get_FirstItemBar(.FocusItem), New String() {"left", "right"}) .set_BarExtraCaptionHAlign(.FocusItem, .get_FirstItemBar(.FocusItem), _ New exontrol.EXG2ANTTLib.AlignmentEnum() {exontrol.EXG2ANTTLib.AlignmentEnum.LeftAlignm Or exontrol.EXG2ANTTLib.AlignmentEnum.exHOutside _ exontrol.EXG2ANTTLib.AlignmentEnum.RightAlign Or exontrol.EXG2ANTTLib.AlignmentEnum.exHOutside .set_BarExtraCaptionVAlign(.FocusItem, .get_FirstItemBar(.FocusItem), _ New exontrol.EXG2ANTTLib.VAlignmentEnum() {exontrol.EXG2ANTTLib.VAlignmentEnum.exTop Or exontrol.EXG2ANTTLib.VAlignmentEnum.exVOutsid _

```
exontrol.EXG2ANTTLib.VAlignmentEnum.exBottom
 Or
exontrol.EXG2ANTTLib.VAlignmentEnum.exVOutsid

End With
```

The following C# sample adds a single extra caption in the right side of the bar:

```
exontrol.EXG2ANTTLib.Items items =
exg2antt1.Items;
items.set_BarExtraCaption(items.FocusItem,
items.get_FirstItemBar(items.FocusItem), "right");
items.set_BarExtraCaptionHAlign(items.FocusItem,
items.get_FirstItemBar(items.FocusItem),
exontrol.EXG2ANTTLib.AlignmentEnum.RightAlign
 |
exontrol.EXG2ANTTLib.AlignmentEnum.exHOutside

items.set_BarExtraCaptionVAlign(items.FocusItem,
items.get_FirstItemBar(items.FocusItem),
exontrol.EXG2ANTTLib.VAlignmentEnum.exBottom
 |
exontrol.EXG2ANTTLib.VAlignmentEnum.exVOutsid
```

The following C# sample adds two extra captions, one in the left side, and one to the right side:

```
exontrol.EXG2ANTTLib.Items items =
exg2antt1.Items;
items.set_BarExtraCaption(items.FocusItem,
items.get_FirstItemBar(items.FocusItem),
    new string[2] { "left", "right" });
items.set_BarExtraCaptionHAlign(items.FocusItem,
items.get_FirstItemBar(items.FocusItem),
    new exontrol.EXG2ANTTLib.AlignmentEnum[2]
{
exontrol.EXG2ANTTLib.AlignmentEnum.LeftAlignm
```

```
 |
exontrol.EXG2ANTTLib.AlignmentEnum.exHOutside

exontrol.EXG2ANTTLib.AlignmentEnum.RightAlign
 |
exontrol.EXG2ANTTLib.AlignmentEnum.exHOutside
 });
items.set_BarExtraCaptionVAlign(items.FocusItem,
items.get_FirstItemBar(items.FocusItem),
   new
exontrol.EXG2ANTTLib.VAlignmentEnum[2] {
exontrol.EXG2ANTTLib.VAlignmentEnum.exTop |
exontrol.EXG2ANTTLib.VAlignmentEnum.exVOutsid

exontrol.EXG2ANTTLib.VAlignmentEnum.exBottom
 |
exontrol.EXG2ANTTLib.VAlignmentEnum.exVOutsid
 });
```

(AlignmentEnum expression, or a safe array of AlignmentEnum/long/vt_i4 expression)

Specifies the vertical alignment for each extra caption assigned to the bar.

The following VB sample adds a single extra caption in the right side of the bar:

```
With G2antt1.Items
   .ItemBar(.FocusItem, .FirstItemBar(.FocusItem),
exBarExtraCaption) = "right"
   .ItemBar(.FocusItem, .FirstItemBar(.FocusItem),
exBarExtraCaptionHAlign) = RightAlignment Or
exHOutside
   .ItemBar(.FocusItem, .FirstItemBar(.FocusItem),
exBarExtraCaptionVAlign) = exBottom Or
exVOutside
End With
```

The following VB sample adds two extra captions, one in the left side, and one to the right side:

```
With G2antt1.Items
    .ItemBar(.FocusItem, .FirstItemBar(.FocusItem),
exBarExtraCaption) = Array("left", "right")
    .ItemBar(.FocusItem, .FirstItemBar(.FocusItem),
exBarExtraCaptionHAlign) = Array(LeftAlignment
Or exHOutside, RightAlignment Or exHOutside)
    .ItemBar(.FocusItem, .FirstItemBar(.FocusItem),
exBarExtraCaptionVAlign) = Array(exTop Or
exVOutside, exBottom Or exVOutside)
End With
```

When retrieving the exBarExtraCaption returns the extra caption, if there is only one, else it returns a safe array (collection) of string that indicates the extra captions of the bar.

The following VB/NET sample adds a single extra caption in the right side of the bar:

```
With Exg2antt1.Items
    .set_BarExtraCaption(.FocusItem,
.get_FirstItemBar(.FocusItem), "right")
    .set_BarExtraCaptionHAlign(.FocusItem,
.get_FirstItemBar(.FocusItem),
exontrol.EXG2ANTTLib.AlignmentEnum.RightAlign
 Or
exontrol.EXG2ANTTLib.AlignmentEnum.exHOutside

    .set_BarExtraCaptionVAlign(.FocusItem,
.get_FirstItemBar(.FocusItem),
exontrol.EXG2ANTTLib.VAlignmentEnum.exBottom
 Or
exontrol.EXG2ANTTLib.VAlignmentEnum.exVOutsid

End With
```

exBarExtraCaptionVAlign	46	The following VB/NET sample adds two extra captions, one in the left side, and one to the right side: `With Exg2antt1.Items` `.set_BarExtraCaption(.FocusItem, .get_FirstItemBar(.FocusItem), New String() {"left", "right"})` `.set_BarExtraCaptionHAlign(.FocusItem, .get_FirstItemBar(.FocusItem), _` `New exontrol.EXG2ANTTLib.AlignmentEnum() {exontrol.EXG2ANTTLib.AlignmentEnum.LeftAlignm` `Or exontrol.EXG2ANTTLib.AlignmentEnum.exHOutside _` `exontrol.EXG2ANTTLib.AlignmentEnum.RightAlign` `Or exontrol.EXG2ANTTLib.AlignmentEnum.exHOutside` `.set_BarExtraCaptionVAlign(.FocusItem, .get_FirstItemBar(.FocusItem), _` `New exontrol.EXG2ANTTLib.VAlignmentEnum() {exontrol.EXG2ANTTLib.VAlignmentEnum.exTop Or exontrol.EXG2ANTTLib.VAlignmentEnum.exVOutsid _` `exontrol.EXG2ANTTLib.VAlignmentEnum.exBottom` `Or exontrol.EXG2ANTTLib.VAlignmentEnum.exVOutsid` `End With` The following C# sample adds a single extra caption in the right side of the bar: `exontrol.EXG2ANTTLib.Items items =`

```
exg2antt1.Items;
items.set_BarExtraCaption(items.FocusItem,
items.get_FirstItemBar(items.FocusItem), "right");
items.set_BarExtraCaptionHAlign(items.FocusItem,
items.get_FirstItemBar(items.FocusItem),
exontrol.EXG2ANTTLib.AlignmentEnum.RightAlign
 |
exontrol.EXG2ANTTLib.AlignmentEnum.exHOutside

items.set_BarExtraCaptionVAlign(items.FocusItem,
items.get_FirstItemBar(items.FocusItem),
exontrol.EXG2ANTTLib.VAlignmentEnum.exBottom
 |
exontrol.EXG2ANTTLib.VAlignmentEnum.exVOutsid
```

The following C# sample adds two extra captions, one in the left side, and one to the right side:

```
exontrol.EXG2ANTTLib.Items items =
exg2antt1.Items;
items.set_BarExtraCaption(items.FocusItem,
items.get_FirstItemBar(items.FocusItem),
   new string[2] { "left", "right" });
items.set_BarExtraCaptionHAlign(items.FocusItem,
items.get_FirstItemBar(items.FocusItem),
   new exontrol.EXG2ANTTLib.AlignmentEnum[2]
{
exontrol.EXG2ANTTLib.AlignmentEnum.LeftAlignm
 |
exontrol.EXG2ANTTLib.AlignmentEnum.exHOutside

exontrol.EXG2ANTTLib.AlignmentEnum.RightAlign
 |
exontrol.EXG2ANTTLib.AlignmentEnum.exHOutside
 });
items.set_BarExtraCaptionVAlign(items.FocusItem,
items.get_FirstItemBar(items.FocusItem),
   new
```

```
exontrol.EXG2ANTTLib.VAlignmentEnum[2] {
exontrol.EXG2ANTTLib.VAlignmentEnum.exTop |
exontrol.EXG2ANTTLib.VAlignmentEnum.exVOutsid

exontrol.EXG2ANTTLib.VAlignmentEnum.exBottom
 |
exontrol.EXG2ANTTLib.VAlignmentEnum.exVOutsid
 });
```

(VAlignmentEnum expression, or a safe array of VAlignmentEnum/long/vt_i4 expression)

Retrieves or sets the offset to move horizontally the extra caption relative to its default position.

The following VB sample adds a single extra caption in the right side of the bar:

```
With G2antt1.Items
    .ItemBar(.FocusItem, .FirstItemBar(.FocusItem),
exBarExtraCaption) = "right"
    .ItemBar(.FocusItem, .FirstItemBar(.FocusItem),
exBarExtraCaptionHAlign) = RightAlignment Or
exHOutside
    .ItemBar(.FocusItem, .FirstItemBar(.FocusItem),
exBarExtraCaptionVAlign) = exBottom Or
exVOutside
    .ItemBar(.FocusItem, .FirstItemBar(.FocusItem),
exBarExtraCaptionHOffset) = 8
End With
```

The following VB sample adds two extra captions, one in the left side, and one to the right side:

```
With G2antt1.Items
    .ItemBar(.FocusItem, .FirstItemBar(.FocusItem),
exBarExtraCaption) = Array("left", "right")
    .ItemBar(.FocusItem, .FirstItemBar(.FocusItem),
```

```
exBarExtraCaptionHAlign) = Array(LeftAlignment
Or exHOutside, RightAlignment Or exHOutside)
    .ItemBar(.FocusItem, .FirstItemBar(.FocusItem),
exBarExtraCaptionVAlign) = Array(exTop Or
exVOutside, exBottom Or exVOutside)
    .ItemBar(.FocusItem, .FirstItemBar(.FocusItem),
exBarExtraCaptionHOffset) = Array(-8, 8)
End With
```

When retrieving the exBarExtraCaption returns the extra caption, if there is only one, else it returns a safe array (collection) of string that indicates the extra captions of the bar.

The following VB/NET sample adds a single extra caption in the right side of the bar:

```
With Exg2antt1.Items
    .set_BarExtraCaption(.FocusItem,
.get_FirstItemBar(.FocusItem), "right")
    .set_BarExtraCaptionHAlign(.FocusItem,
.get_FirstItemBar(.FocusItem),
exontrol.EXG2ANTTLib.AlignmentEnum.RightAlign
 Or
exontrol.EXG2ANTTLib.AlignmentEnum.exHOutside

    .set_BarExtraCaptionVAlign(.FocusItem,
.get_FirstItemBar(.FocusItem),
exontrol.EXG2ANTTLib.VAlignmentEnum.exBottom
 Or
exontrol.EXG2ANTTLib.VAlignmentEnum.exVOutsid

    .set_BarExtraCaptionHOffset(.FocusItem,
.get_FirstItemBar(.FocusItem), 8)
End With
```

The following VB/NET sample adds two extra captions, one in the left side, and one to the right side:

exBarExtraCaptionHOffset	47	With Exg2antt1.Items .set_BarExtraCaption(.FocusItem, .get_FirstItemBar(.FocusItem), New String() {"left", "right"}) .set_BarExtraCaptionHAlign(.FocusItem, .get_FirstItemBar(.FocusItem), _ New exontrol.EXG2ANTTLib.AlignmentEnum() {exontrol.EXG2ANTTLib.AlignmentEnum.LeftAlignm Or exontrol.EXG2ANTTLib.AlignmentEnum.exHOutside _ exontrol.EXG2ANTTLib.AlignmentEnum.RightAlignı Or exontrol.EXG2ANTTLib.AlignmentEnum.exHOutside .set_BarExtraCaptionVAlign(.FocusItem, .get_FirstItemBar(.FocusItem), _ New exontrol.EXG2ANTTLib.VAlignmentEnum() {exontrol.EXG2ANTTLib.VAlignmentEnum.exTop Or exontrol.EXG2ANTTLib.VAlignmentEnum.exVOutsid _ exontrol.EXG2ANTTLib.VAlignmentEnum.exBottom Or exontrol.EXG2ANTTLib.VAlignmentEnum.exVOutsid .set_BarExtraCaptionHOffset(.FocusItem, .get_FirstItemBar(.FocusItem), _ New Integer() {-8, 8}) End With

The following C# sample adds a single extra caption in the right side of the bar:

```
exontrol.EXG2ANTTLib.Items items =
```

```
exg2antt1.Items;
items.set_BarExtraCaption(items.FocusItem,
items.get_FirstItemBar(items.FocusItem), "right");
items.set_BarExtraCaptionHAlign(items.FocusItem,
items.get_FirstItemBar(items.FocusItem),
exontrol.EXG2ANTTLib.AlignmentEnum.RightAlign
 |
exontrol.EXG2ANTTLib.AlignmentEnum.exHOutside

items.set_BarExtraCaptionVAlign(items.FocusItem,
items.get_FirstItemBar(items.FocusItem),
exontrol.EXG2ANTTLib.VAlignmentEnum.exBottom
 |
exontrol.EXG2ANTTLib.VAlignmentEnum.exVOutsid

items.set_BarExtraCaptionHOffset(items.FocusItem,
 items.get_FirstItemBar(items.FocusItem), 8);
```

The following C# sample adds two extra captions, one in the left side, and one to the right side:

```
exontrol.EXG2ANTTLib.Items items =
exg2antt1.Items;
items.set_BarExtraCaption(items.FocusItem,
items.get_FirstItemBar(items.FocusItem),
    new string[2] { "left", "right" });
items.set_BarExtraCaptionHAlign(items.FocusItem,
items.get_FirstItemBar(items.FocusItem),
    new exontrol.EXG2ANTTLib.AlignmentEnum[2]
{
exontrol.EXG2ANTTLib.AlignmentEnum.LeftAlignm
 |
exontrol.EXG2ANTTLib.AlignmentEnum.exHOutside

exontrol.EXG2ANTTLib.AlignmentEnum.RightAlign
 |
exontrol.EXG2ANTTLib.AlignmentEnum.exHOutside
 });
```

```
items.set_BarExtraCaptionVAlign(items.FocusItem,
items.get_FirstItemBar(items.FocusItem),
   new
exontrol.EXG2ANTTLib.VAlignmentEnum[2] {
exontrol.EXG2ANTTLib.VAlignmentEnum.exTop |
exontrol.EXG2ANTTLib.VAlignmentEnum.exVOutsid

exontrol.EXG2ANTTLib.VAlignmentEnum.exBottom
 |
exontrol.EXG2ANTTLib.VAlignmentEnum.exVOutsid
 });
items.set_BarExtraCaptionHOffset(items.FocusItem,
 items.get_FirstItemBar(items.FocusItem),
   new int[2] { -8, 8 });
```

(long expression, or a safe array of long/vt_i4 expression)

exBarExtraCaptionVOffset	48	Retrieves or sets the offset to move vertically the extra caption relative to its default position. Please check the exBarExtraCaptionHOffset for usage. *(long expression, or a safe array of long/vt_i4 expression)*
		This option can be user with the PutRes method. Specifies the list of resources associated to the bar. The exBarResources property (get/set) indicates the resources to be used by the current bar in the Source, as a string expression. The exBarResources property is a string expression that indicate the list of resources (including its usage, or 100% if missing). The resources are separated by , (comma) character, while the usage is specified as a double expression (using the . dot character as a decimal separator). *For instance the "Resource1,Resource2,Resource3" indicates that the bar uses the Resource1,Resource2,Resource3, while "R1,R2[50%],R3[67.89%]" specifies that the*

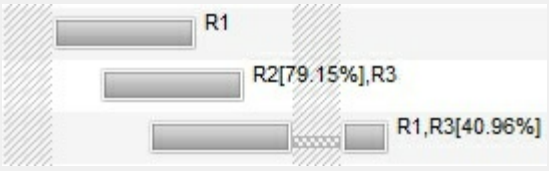
bar uses the R1 on 100%, R2 on 50% and R3 on 67.89%.

You can use the exBarCaption to display the bar's resources using the <%=formula%> format, like in the following VB sample:

```
With G2antt1.Chart.Bars("Task")

.Def(EXG2ANTTLibCtl.ItemBarPropertyEnum.ex
 = "<%=%" &
EXG2ANTTLibCtl.ItemBarPropertyEnum.exBarR
 & "%>"

.Def(EXG2ANTTLibCtl.ItemBarPropertyEnum.ex
 = 18
End With
```

In other words, the sample allows you to display the bar's exBarResources property as shown bellow:

exBarResources 49



The set exBarResources property could be used in the following format based on the first character as listed:

- If the first character is **+(plus)**, the rest of the expression indicates the resources to be assigned to the current bar. *For instance, if the current bar has the exBarResources property as "R1,R2" , and we call set exBarResources as "+R3", it means that the R3 is added to the bar's resources, and so the new exBarResources property is "R1,R2,R3".*
- If the first character is **-(minus)**, the rest of the expression indicates the resources

to be removed from the current bar. *For instance, if the current bar has the exBarResources property as "R1,R2" , and we call set exBarResources as "-R2", it means that the R2 is removed from the bar's resources, and so the new exBarResources property is "R1".*

- If no +,- character, the new expression replaces the exBarResources property. *For instance, if the current bar has the exBarResources property as "R1,R2" , and we call set exBarResources as "R3,R4", it means that the new exBarResources property is "R3,R4".*

(*String expression*)

This option can be user with the PutRes method. Specifies the format to display the bar's resource. The exBarResourceFormat property (get/set) indicates the format or the expression to be used if you need to display the bar's resource in a different format, in Source, as a string expression. The expression supports the **name** keyword which indicates the name of the resource, and the **percent** keyword to get the usage percent a a double expression between 0 and 1. The expression supports all predefined functions listed here. Use the exBarResource**s**Format (NOT exBarResourceFormat, whith no s) to get the HTML value of the formatted string using the bar's resources. *For instance, let's say we need to display the resource names in bold, and the usage percent in a smaller font and a different foreground color, so the Items.ItemBar(exBarResourceFormat) property could be "`<b>` + name + `</b><font ;5> <fgcolor=404040>` + (percent = 1 ? `` : (round(100*percent) format ``) + `%`) + `</fgcolor></font>`", and so the VB sample could show as:*

```
With G2antt1.Chart.Bars("Task")
```

exBarResourceFormat	50	.Def(exBarResourceFormat) = "`<b>` + name + `</b><font ;5><fgcolor=404040>` + (percent = 1 ? `` : (round(100*percent) format ``) + `%`) + `</fgcolor></font>`" .Def(EXG2ANTTLibCtl.ItemBarPropertyEnum.ex = "<%=%" & EXG2ANTTLibCtl.ItemBarPropertyEnum.exBarR & "%>" .Def(EXG2ANTTLibCtl.ItemBarPropertyEnum.ex = 18 End With In other words, the sample allows you to display the bar's exBarResources property as shown bellow:  (*String expression*)
		Specifies the color to show the bar's frame or an additional EBN object that may be displayed on the bar to indicate a frame, a note or a symbol. The last 7 bits in the high significant byte of the color indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part. For instance, you can use the exBarColor to specify the inside bar's color, while the exBarFrameColor indicates the color for the border to be shown. The following screen shot shows a symbol using the exBarFrameColor:

The following sample adds a Task bar with a Red frame:

exBarFrameColor 51

```
With G2antt1.Items
   h = .AddItem("Red-Frame")
   .AddBar h,"Task",#1/3/2001#,#1/6/2001#,"K1"
   .ItemBar(h,"K1",exBarFrameColor) =
RGB(255,0,0)
End With
```

The following sample adds a Task bar with an EBN frame:

```
With G2antt1
   .VisualAppearance.Add 1,".../tickers.ebn"
   With .Items
      h = .AddItem("EBN-Frame")
      .AddBar
h,"Task",#1/3/2001#,#1/6/2001#,"K1"
      .ItemBar(h,"K1",exBarFrameColor) =
&H1000000
   End With
End Withh
```

(Color expression)

Indicates the z-order when the bar is arranged in cascade, when the bar is overlapping with other bars. This option is valid only if the OverlaidType property of the current bar is exOverlaidBarsCascade. The cascade type allows you to arrange the bars on the same levels for those with the same exBarOverlaidCascade key, and on a different level for bars with a different exBarOverlaidCascade key, like in the following screen shot. The levels in the cascade is displayed in the alphabetic order like explained in the following sample.

The following screen shot shows the bars arranged on cascade, K1, K2, K3 on the first level, and the T1, T2, T3 on the second level.

The bars get arranged into a cascade/levels based on the key (exOverlaidBarsCascade). The T1, T2, T3 are shown on the same level "B", as they have the same exBarOverlaidCascade key, and does not intersect the K1, K2, K3 level "A".

exBarOverlaidKey 52

For instance the following sample:

```
Items.ItemBar(0,"<K*>",exBarOverlaidKey) = "A"
Items.ItemBar(0,"<T*>",exBarOverlaidKey) = "B"
```

specifies that the K* bars (all bars with the key starting with the K character), should be displayed on the same level "A", and the T* bars on the level "B", which indicates that the A will be displayed as the first level, and the B as the second level. In other words, the cascades/levels are being displayed in their alphabetic order.

If we exchange the "A" with "B" like in the following sample:

```
Items.ItemBar(0,"<K*>",exBarOverlaidKey) = "B"
Items.ItemBar(0,"<T*>",exBarOverlaidKey) = "A"
```

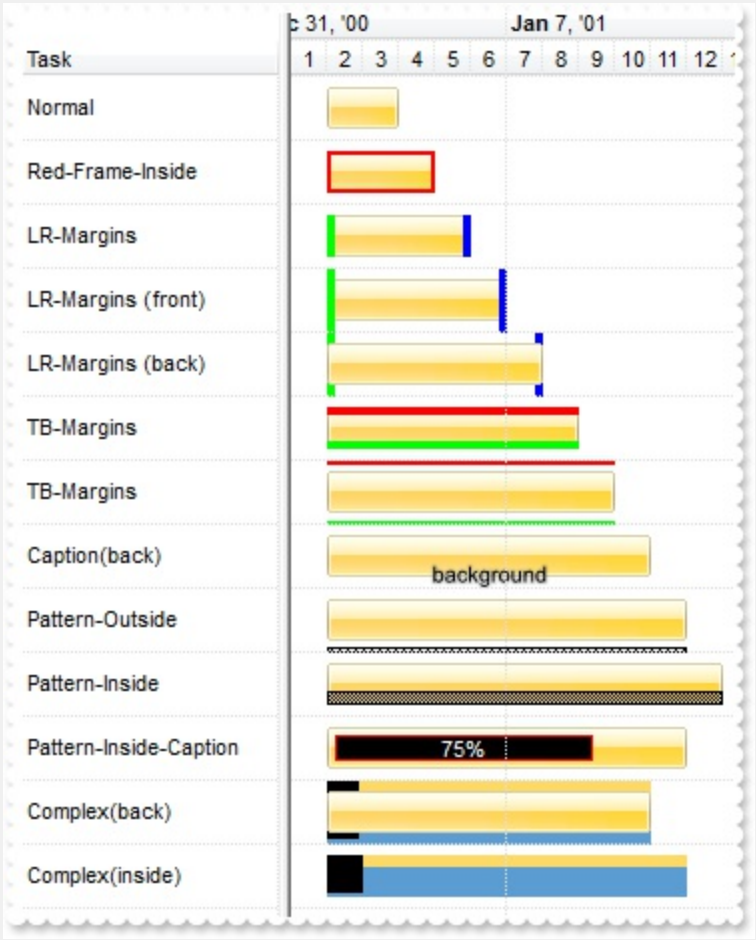
we get the following screen show

(Variant expression)

Specifies unlimited options to show any HTML text, images, colors, EBNs, patterns, frames anywhere on the bar's background, using EBN String

Format. The exBarBackgroundExtFlags, specifies whether the extension is shown on the back or the front of the bar, while exBarBackgroundExtInflate increases or decreases the margins of the portion where the exBarBackgroundExt is applied/shown.

The following screen shot shows a few options you can have by using the EBN String Format on bars/tasks:

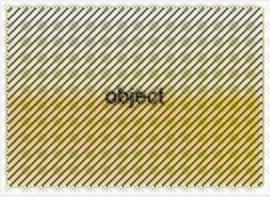
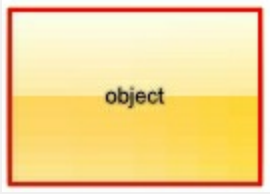
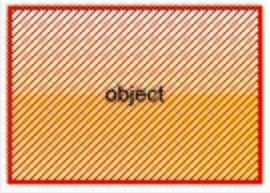
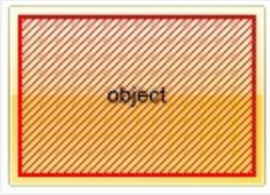


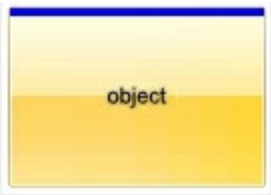
The **EBN String Format** syntax in BNF notation is defined like follows:

```
<EBN> ::= <elements> | <root> "(" [<elements>] ")"
<elements> ::= <element> [ "," <elements> ]
<root> ::= "root" [ <attributes> ] | [ <attributes> ]
<element> ::= <anchor> [ <attributes> ] [ "(" [<elements>] ")" ]
<anchor> ::= "none" | "left" | "right" | "client" | "top" | "bottom"
```

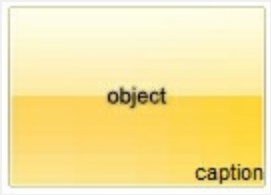
```
<attributes> ::= "[" [<client> ","] <attribute> [ ","
<attributes> ] "]"
<client> ::= <expression> | <expression> ","
<expression> "," <expression> "," <expression>
<expression> ::= <number> | <number> "%"
<attribute> ::= <backcolor> | <text> |
<wordwrap> | <align> | <pattern> |
<patterncolor> | <frame> | <framethick> |
<data> | <others>
<equal> ::= "="
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<decimal> ::= <digit><decimal>
<hexadigit> ::= <digit> | "A" | "B"  "C" | "D" | "E"
"F"
<hexa> ::= <hexadigit><hexa>
<number> ::= <decimal> | "0x" <hexa>
<color> ::= <rgbcolor> | number
<rgbcolor> ::= "RGB" "(" <number> ","
<number> "," <number> ")"
<string> ::= "`" <characters> "`" | "'" <characters>
"'" | " <characters> "
<characters> ::= <char>|<characters>
<char> ::= <any_character_excepts_null>
<backcolor> ::= "back" <equal> <color>
<text> ::= "text" <equal> <string>
<align> ::= "align" <equal> <number>
<pattern> ::= "pattern" <equal> <number>
<patterncolor> ::= "patterncolor" <equal>
<color>
<frame> ::= "frame" <equal> <color>
<data> ::= "data" <equal> <number> | <string>
<framethick> ::= "framethick"
<wordwrap> ::= "wordwrap"
```

Others like: pic, stretch, hstretch, vstretch, transparent, from, to are reserved for future use only.

exBarBackgroundExt	53	Here's a few easy samples: • "[pattern=6]", shows the BDiagonal pattern on the object's background.  • "[frame=RGB(255,0,0),framethick]", draws a red thick-border around the object.  • "[frame=RGB(255,0,0),framethick,pattern=6,patt draws a red thick-border around the object, with a patter inside.  • "[[patterncolor=RGB(255,0,0)](none[(4,4,100%-8,100%-8),pattern=0x006,patt draws a red thick-border around the object, with a patter inside, with a 4-pixels wide padding:  • "top[4,back=RGB(0,0,255)]", draws a blue line on the top side of the object's background, of 4-pixels wide.



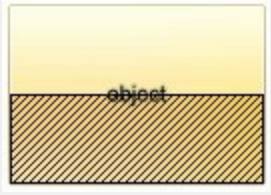
- "[text=`caption`,align=0x22]", shows the caption string aligned to the bottom-right side of the object's background.



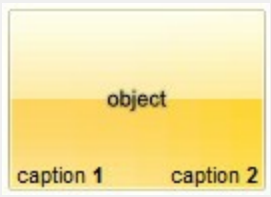
- "[text=`<img>flag</img>`,align=0x11]" shows the flag picture and the sweden string aligned to the bottom side of the object.

- "left[10,back=RGB(255,0,0)]", draws a red line on the left side of the object's background, of 10-pixels wide.

- "bottom[50%,pattern=6,frame]", shows the BDiagonal pattern with a border arround on the lower-half part of the object's background.

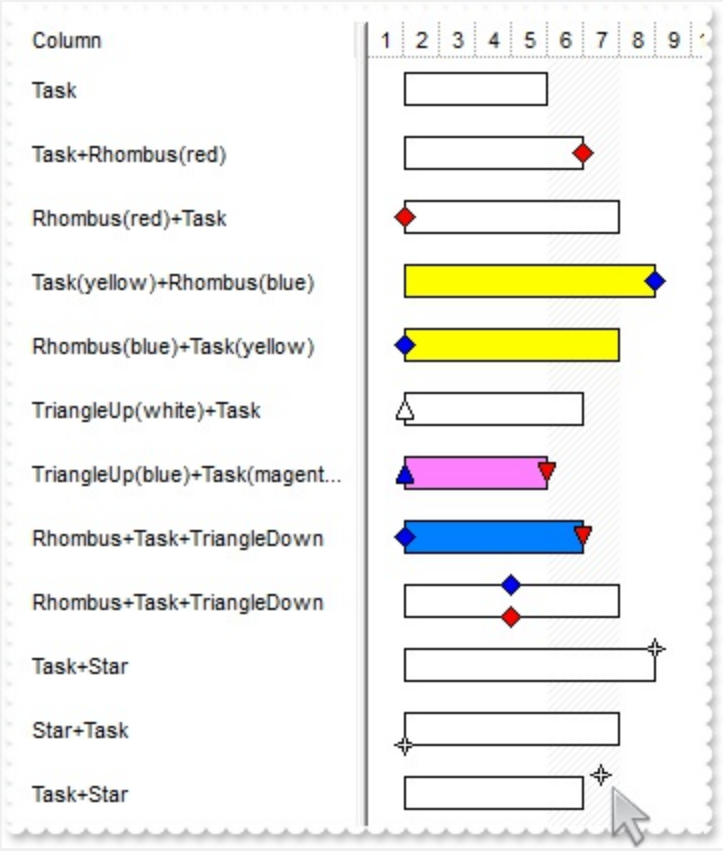


- "root[text=`caption <b>2`,align=0x22] (client[text=`caption <b>1`,align=0x20])",

		shows the caption **1** aligned to the bottom-left side, and the caption **2** to the bottom-right side  (*String expression*)
exBarBackgroundExtFlags	54	Specifies the flags to show the exBarBackgroundExt on the bar's background. By default, the exBarBackgroundExtFlags property is 0, which indicates that the exBarBackgroundExt property is applied in front of the bar's client area. The exBarBackgroundExtInflate increases or decreases the margins of the portion where the exBarBackgroundExt is applied/shown. The exBarBackgroundExtFlagssupports a combination of the following values: • **0**, the exBarBackgroundExt is applied in front, using the bar's client area. The Height of the bar specifies the height of the bar, and so it defines the bar's client area. • **1**, the exBarBackgroundExt is applied on the back of the bar. If missing, the exBarBackgroundExt is applied on front. • **2**, the exBarBackgroundExt is uses the bar's background client are to show the exBarBackgroundExt option. The Height of the item that hosts the bar specifies the client area of the bar's background. If missing, the bar's client area is used instead. *(Long expression, between 0 and 3, by default it is 0)*
		Increases or decreases the margins of the portion where the exBarBackgroundExt is applied/shown. By default, the exBarBackgroundExtInflate property

is 0, which indicates that the bar's client area is the portion where the exBarBackgroundExt is applied. For instance, "-10,0,10,0" enlarges the bar's margins by 10 pixels, to left and right, and so the exBarBackgroundExt looks wider. If the exBarBackgroundExtInflate property is of numeric type, it specifies the range to increases or decreases all the margins. The exBarBackgroundExt adds unlimited options to show any HTML text, images, colors, EBNs, patterns, frames anywhere on the bar's background. Ability to draw additional EBN/Color/Text/Patterns/Images on the bar, using the EBN String Format (create and run at runtime EBN objects)

exBarBackgroundExtInflate 55



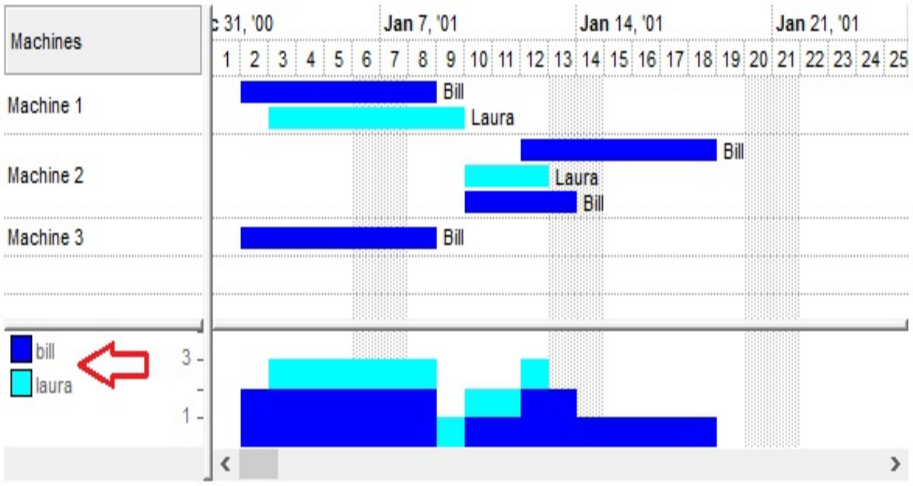
(Long expression, which indicates that all margins of the extension is increased or decreased with specified value, a String expression such as "left,top,right,bottom", to specify different margins for the portion to show the extension)

Shows or hides the bar's caption. The exBarCaption specifies the bar's caption. The exBarHAlignCaption

exBarShowCaption	56	/ exBarVAlignCaption aligns horizontally / vertically the bar's caption relative to bar's client-rectangle. *(Boolean expression)*
exBarShowExtraCaption	57	Shows or hides the bar's extra caption. The exBarExtraCaption specifies the bar's extra caption. The exBarExtraCaptionHAlign/ exBarExtraCaptionVAlignaligns horizontally / vertically the bar's extra caption relative to bar's client-rectangle. *(Boolean expression)*
exBarCaptionHOffset	58	Indicates the bar's caption horizontal offset. The exBarCaptionVOffset indicates the bar's caption vertical offset. The exBarCaption specifies the bar's caption. The exBarHAlignCaption / exBarVAlignCaption aligns horizontally / vertically the bar's caption relative to bar's client-rectangle. *(short expression)*
exBarCaptionVOffset	59	Indicates the bar's caption vertical offset. The exBarCaptionHOffset indicates the bar's caption horizontal offset. The exBarCaption specifies the bar's caption. The exBarHAlignCaption / exBarVAlignCaption aligns horizontally / vertically the bar's caption relative to bar's client-rectangle. *(short expression)*
		Specifies the description to show within the histogram's legend for the bar in the control's histogram. Use the ItemBar(exBarColor) property to specify a different color for a specified bar. The exBarHistLegend option has effect only if the bar's HistogramCumulativeOriginalColorBars property is exKeepOriginalColor. For exKeepOriginalColor, the HistogramCumulativeShowLegend property specifies whether the bar's legend (ItemBar(

exBarHistLegend	60	exBarHistLegend) property) is shown or hidden. This option supports built-in **HTML** format including the **<%=formula%>** tag. The **<%=formula%>** tag indicates the result of the giving formula. The formula supports value formatting. The **formula** supports the following keywords: • ***%0**, **%1**, **%2**, ... specifies the corresponding property of the bar, such as %0 indicates the exBarName, %1 exBarStart, %2 exBarEnd, and so on.* • ***%C0**, **%C1**, **%C2**, ... specifies the caption of the cell, or the string the cell displays in the column with the index 0, 1 2, ... The CellCaption property specifies the cell's caption. The cell's value may be different than what the cell displays as a string. For instance, let's say a cell display HTML format. The %0 returns the html format including the HTML tags, while %C0 returns the cell's content as string without HTML tags. For instance, "upper(%C1)" converts the caption of the cell with the index 1, to upper case, while "%C0 left 2" returns the leftmost two characters on the cell with the index 0.* • ***%CD0**, **%CD1**, **%CD2**, ... specifies the cell's extra data in the column with the index 0, 1 2, ... The CellData property associates any extra/user data to a cell. For instance, "%CD0 = `your user data`" specifies all cells whose CellData property is `your user data`, on the column with the index 0.* • ***%CS0**, **%CS1**, **%CS2**, ... specifies the cell's state in the column with the index 0, 1 2, ... The CellState property specifies the cell's state, and so it indicates whether the cell is checked or un-checked. For instance, "%CS0" defines all checked items on the column with the index 0, or "not %CS1" defines all un-checked items in the column with the index 1.*

For instance the Items.ItemBar(exBarHistLegend) = "<fgcolor=666666><%=lower(%3)%>" defines the bar's histogram-legend as the following screen shot:



This property/method supports predefined constants and operators/functions as described here.

(HTML String expression)

exBarsCount	256	Retrieves a value that indicates the number of bars in the item. The exBarsCount property counts the bars being displayed in the item. Use the AddBar property to add new bars to the item. This option ignores the Key parameter, so no matter what you are using for the Key parameter it gets the number of bars in the item. For instance, the Items.ItemBar(Item,Nothing,exBarsCount) property gets the number of bars inside the specified item. *(Long expression)*
exBarSelected	257	Specifies whether the bar is selected or unselected. By default, the exBarSelected is False. The AllowSelectObjects property allows users to select at runtime the bars and links in the chart area. The ChartSelectionChanged event is fired when the selection in the chart is changed. *(Boolean expression)*

exBarWorkingCount	258	Specifies the count of working units (days) in the bar. For instance, 1 indicates one working day, while 0.5 indicates 12 hours from a working day. The NonworkingDays property specifies the non-working days. Use the AddNonworkingDate property to add custom non-working days. Use the NonworkingHours property to specify the non-working hours. Use the exBarWorkingCount property to specify the number of working days for a specified bar. For instance, if your chart displays days, and the NonworkingDays is set, the exBarWorkingCount property sets or gets the count of working days in the bar. If the chart displays hours, and the NonworkingHours property is set, the exBarWorkingCount property sets or gets the count of working hours in the bar. *(Float expression)*
exBarNonWorkingCount	259	Specifies the count of non-working units (days) in the bar. For instance, 1 indicates one non-working day, while 0.5 indicates 12 hours from a non-working day. The NonworkingDays property specifies the non-working days. Use the AddNonworkingDate property to add custom non-working days. Use the NonworkingHours property to specify the non-working hours. For instance, if your chart displays days, and the NonworkingDays is set, the exBarNonworkingCount property gets the count of non-working days in the bar. If the chart displays hours, and the NonworkingHours property is set, the exBarNonWorkingCount property gets the count of non-working hours in the bar. *(Float expression)*
		Retrieves a collection of pairs (start-end) that indicates the non-working parts of the bar. You can use the exBarNonWorkingUnitsAsString property to get the non-working parts of the bar as a string. The /NET Assembly provides a

exBarNonWorkingUnits	260	get_BarNonWorkingUnits method that retrieves a collection of DateTime objects. The following VB sample lists the start and end date-time values for non-working parts of the bar (for the /COM version): Private Sub G2antt1_BarResize(ByVal item As EXG2ANTTLibCtl.HITEM, ByVal Key As Variant) Debug.Print "Non-working parts of the bar:" With G2antt1.Items Dim i As Variant, j As Long For Each i In .ItemBar(item, Key, exBarNonWorkingUnits) Debug.Print IIf(j Mod 2 = 0, "Start ", "End ") & i j = j + 1 Next End With End Sub The following VB/NET sample lists the start and end date-time values for non-working parts of the bar (for the /NET Assembly version): Private Sub Exg2antt1_BarResize(ByVal sender As System.Object, ByVal Item As System.Int32, ByVal Key As System.Object) Handles Exg2antt1.BarResize Debug.Print("Non-working parts of the bar:") With Exg2antt1.Items Dim i As Object, j As Long For Each i In .get_BarNonWorkingUnits(Item, Key) Debug.Print(IIf(j Mod 2 = 0, "Start ", "End ") & i) j = j + 1 Next End With

```
End Sub
```

(safe array of pairs of dates indicating the start and end of the non-working area)

exBarNonWorkingUnitsAsString	261

Displays as string the collection of pairs (start-end) that indicates the non-working parts of the bar. The /NET Assembly provides a get_BarNonWorkingUnitsAsString method that retrieves the non-working parts of the bar as string

The following VB sample lists the start and end date-time values for non-working parts of the bar (for the /COM version):

```
Private Sub G2antt1_BarResize(ByVal item As
EXG2ANTTLibCtl.HITEM, ByVal Key As Variant)
   Debug.Print "Non-working parts of the bar:"
   With G2antt1.Items
      Debug.Print .ItemBar(item, Key,
exBarNonWorkingUnitsAsString)
   End With
End Sub
```

The following VB/NET sample lists the start and end date-time values for non-working parts of the bar (for the /NET Assembly version):

```
Private Sub Exg2antt1_BarResize(ByVal sender As
System.Object, ByVal Item As System.Int32, ByVal
Key As System.Object) Handles
Exg2antt1.BarResize
   Debug.Print("Non-working parts of the bar:")
   With Exg2antt1.Items

Debug.Print(.get_BarNonWorkingUnitsAsString(Iter
 Key))
   End With
End Sub
```

(String expression)

exBarWorkingUnits	262	Retrieves a collection of pairs (start-end) that indicates the working parts of the bar. You can use the exBarWorkingUnitsAsString property to get the working parts of the bar as a string. The /NET Assembly provides a get_BarWorkingUnits method that retrieves a collection of DateTime objects. Use the exBarStartWorking option to get the start working date of the bar. Use the exBarEndWorking option to get the end working date of the bar. The following VB sample lists the start and end date-time values for working parts of the bar (for the /COM version): `Private Sub G2antt1_BarResize(ByVal item As EXG2ANTTLibCtl.HITEM, ByVal Key As Variant)` `  Debug.Print "Working parts of the bar:"` `  With G2antt1.Items` `  Dim i As Variant, j As Long` `  For Each i In .ItemBar(item, Key, exBarWorkingUnits)` `  Debug.Print IIf(j Mod 2 = 0, "Start ", "End ") & i` `  j = j + 1` `  Next` `  End With` `End Sub` The following VB/NET sample lists the start and end date-time values for working parts of the bar (for the /NET Assembly version): `Private Sub Exg2antt1_BarResize(ByVal sender As System.Object, ByVal Item As System.Int32, ByVal Key As System.Object) Handles Exg2antt1.BarResize` `  Debug.Print("Working parts of the bar:")` `  With Exg2antt1.Items`

```
    Dim i As Object, j As Long
    For Each i In .get_BarWorkingUnits(Item, Key)
      Debug.Print(Iif(j Mod 2 = 0, "Start ", "End
") & i)
      j = j + 1
    Next
  End With
End Sub
```

(safe array of pairs of dates indicating the start and end of the working area)

Displays as string the collection of pairs (start-end) that indicates the working parts of the bar. The /NET Assembly provides a get_BarWorkingUnitsAsString method that retrieves the working parts of the bar as string.

The following VB sample lists the start and end date-time values for working parts of the bar (for the /COM version):

```
Private Sub G2antt1_BarResize(ByVal item As
EXG2ANTTLibCtl.HITEM, ByVal Key As Variant)
  Debug.Print "Working parts of the bar:"
  With G2antt1.Items
    Debug.Print .ItemBar(item, Key,
exBarWorkingUnitsAsString)
  End With
End Sub
```

exBarWorkingUnitsAsString 263

The following VB/NET sample lists the start and end date-time values for working parts of the bar (for the /NET Assembly version):

```
Private Sub Exg2antt1_BarResize(ByVal sender As
System.Object, ByVal Item As System.Int32, ByVal
Key As System.Object) Handles
```

		`` Exg2antt1.BarResize Debug.Print("Working parts of the bar:") With Exg2antt1.Items Debug.Print(.get_BarWorkingUnitsAsString(Item, Key)) End With End Sub ``` *(String expression)*
exBarStartWorking	264	Retrieves the start working date of the bar. Use the exBarWorkingUnits option to get the list of working units in the specified bar. For instance, if the bar starts in a working area, the exBarStart and exBarStartWorking properties gets the same result. If the bar starts into a non-working portion of the chart, the exBarStartWorking gets the first working units, where the bar begins. *(Date expression)*
exBarEndWorking	265	Retrieves the end working date of the bar. Use the exBarWorkingUnits option to get the list of working units in the specified bar. For instance, if the bar ends in a working area, the exBarEnd and exBarEndWorking properties gets the same result. If the bar ends into a non-working portion of the chart, the exBarEndWorking gets the previously working units, where the bar ends. *(Date expression)*
exBarResourcesFormat	266	This option can be used with the PutRes method. Retrieves the list of bar's resources using a formatted string. The exBarResourcesFormat property (get only) returns formatted expression of the exBarResources using the exBarResourceFormat (no **s**, so it is exBarResourceFormat, not exBarResource**s**Format

		). *(String expression)*
exBarResourcesNames	267	This option can be used with the PutRes method. Retrieves the list of names of resources being assigned to the bar. exBarResourcesNames property (get only) returns the name of each resource to be used by the current bar, in the Source, as a string expression. This option returns no percent or usage of any resource. *For instance, if the exBarResources property is "R3[67.89%],R4[23.23%]", the exBarResourcesNames property gets the "R3,R4".* *(String expression)*
exBarResourcesUsages	268	This option can be used with the PutRes method. Retrieves the list of usages of resources being assigned to the bar. The exBarResourcesUsages property (get only) returns the usage (double expression from 0 to 1) of each resource to be used by the current bar, in the Source, as a string expression. This option returns no name any resource. *For instance, if the exBarResources property is "R3[67.89%],R4[23.23%]", the exBarResourcesUsages property gets the "0.6789,0.2323".* *(String expression)*
		Indicates whether the current bar is part of the critical path, if the value is not 0. The DefSchedulePDM(exPDMCriticalPathBarColor) specifies the color to display the activities (bars) in the critical path. The value of the exBarCriticalPath property is valid only after calling the SchedulePDM method, and the DefSchedulePDM(exPDMCriticalPathBarColor) is not zero. The SchedulePDM method arranges the activities (bars) in the chart based on their

relationships (links). Previously, the value of the exBarCriticalPath was 0/False (if the bar is not part of the critical path), or -1/True (indicates that the bar is part of the critical path). Currently, the exBarCriticalPath property specifies the position (1-based) of the bar in the critical path. In other words the current bar is part of the critical path, if the exBarCriticalPath property is not 0. Any positive value indicates the position of the current bar in the critical path.

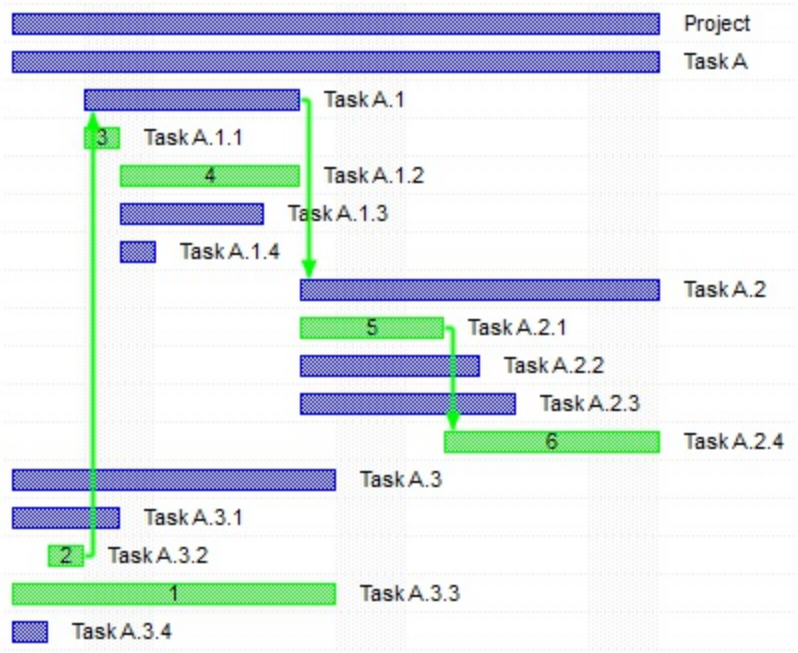
exBarCriticalPath 269

You can automatically specify the critical path position of the bar in its caption using a code like:

```
Items.ItemBar(0, "<*>", exBarCaption) = "
<%=int(%269) > 0 ? %269 : ``%>"
```

The code, changes the caption of all bars, using an expression that shows the integer (value of property 269/exBarCriticalPath) if it is positive, else displays nothing.

The following screen shot shows the critical path, including the position of the bars:



(Currently, Long expression, Previously Boolean expression)

Indicates the list of predecessor bars, separated by

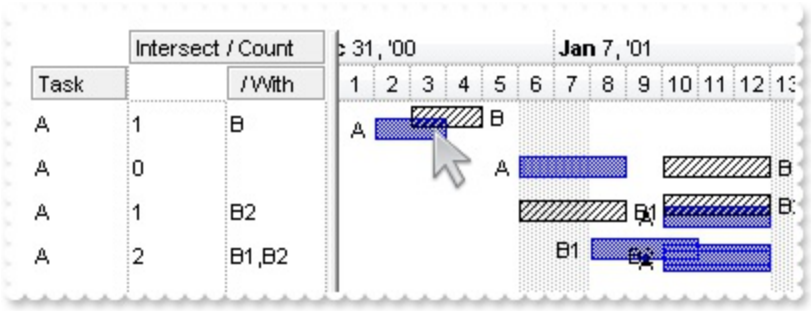
exBarPredecessor	270	comma. A dependency/link is the relationship between predecessor and successor tasks. Tasks may have multiple predecessors or multiple successors. Before you begin establishing dependencies, its important to understand that there are four types: • Finish to Start (FS), the predecessor ends before the successor can begin • Start to Start (SS), the predecessor begins before the successor can begin • Finish to Finish (FF), the predecessor ends before the successor can end • Start to Finish (SF), the predecessor begins before the successor can end The format of bar's predecessor is INDEX1["SF"|"FS"|"FF"|"SS"][KEY][:["W"]LAG|:LAG["W"]], where • INDEX1 is the 1-based index of the item that hosts the bar • followed by the type of the link which can be one any of SF(Start-Finish), FS(Finish-Start), SS(Start-Start) or FF(Finish-Finish) sequence (FS if missing) • continues with the KEY of the bar (empty is not used) • and ends with the LAG of the link (specifies the delay the activity is postponed by the link). The "W" indicates a working-lag for the link (specifies the delay in working-units the activity is postponed by the link). For instance: • "2FSZ", specifies that the current item-bar is linked with the "Z" bar of the second item (item with the index 1) using a Finish-Start link • "1SF:-2", adds a Start-Finish link with the bar '' of the first-item, using a lag of -2 days Changing the ItemBar(exBarPredecessor) property updates the links related to the current bar. The

		AddLink., RemoveLink, RemoveLinkOf methods adds, removes links. The Background(exPSLinkColorEditSel) property specifies the color to highlight the links being selected within an editable predecessor/successor column. The Background(exPSBarColorEditSel) property specifies the color to highlight the incoming/outgoing bars of the links being selected within an editable predecessor/successor column *(String expression)*
exBarSuccessor	271	Indicates the list of successor bars, separated by comma. A dependency/link is the relationship between predecessor and successor tasks. Tasks may have multiple predecessors or multiple successors. Before you begin establishing dependencies, its important to understand that there are four types: • Finish to Start (FS), the predecessor ends before the successor can begin • Start to Start (SS), the predecessor begins before the successor can begin • Finish to Finish (FF), the predecessor ends before the successor can end • Start to Finish (SF), the predecessor begins before the successor can end The format of bar's successo is INDEX1["SF"|"FS"|"FF"|"SS"][KEY][:["W"]LAG|:LAG["W"]], where • INDEX1 is the 1-based index of the item that hosts the bar • followed by the type of the link which can be one any of SF(Start-Finish), FS(Finish-Start), SS(Start-Start) or FF(Finish-Finish) sequence (FS if missing) • continues with the KEY of the bar (empty is not used) • and ends with the LAG of the link (specifies the

		delay the activity is postponed by the link). The "W" indicates a working-lag for the link (specifies the delay in working-units the activity is postponed by the link). For instance: • "3SFy", specifies that the current item-bar is linked with the "y" bar of the third item (item with the index 1) using a Start-Finish link Changing the ItemBar(exBarSuccessor) property updates the links related to the current bar. The AddLink., RemoveLink, RemoveLinkOf methods adds, removes links. *(String expression)*
exBarParent	512	Specifies the handle of the parent item that displays the bar. The Item parameter of the AddBar method indicates the handle of the item that hosts the bar. Use the exBarCanMoveToAnother option to specify whether the user can move a bar from one item to another by drag and drop. The control fires the BarParentChange event just before moving the bar to another item. Use this event to control the items where your bar can be moved. A bar can be moved to another item, ONLY if the second item does not contain a bar with the same key. The exBarKey property specifies the key of the bar. Moving a bar from an item to another (changing the Bar's parent) fails, if the new parent already contains a bar with the same key. An item can hold multiple bars, so each bar is identified by its key, so this key must be unique in the item, but could be the same on several items. *(Long expression)*
		Specifies the duration or the length of the bar in days (or hours if including the decimal point, for instance 0.5 indicates a 12 hours lenght). Gets the

exBarDuration	513	difference between exBarEnd and exBarStart as a double expression. If calling the set property, it changes the bar's duration or length. If negative the start date is computed as the end - duration, since if it is positive, the end date is start + duration. The round part indicates the number of days. Use the exBarMove property to move programmatically a bar by specified time. If you need to change both start and end points of the bar in the same time, you can call the AddBar method with the new coordinates, same item and key. Use the exBarDurationPrev property to get the length or duration of the bar before resizing it. Use the exBarMinDuration and exBarMaxDuration properties to specify the limits for the bar's duration. *(Float expression)*
exBarMove	514	Moves the bar inside the same item by specified amount of time. If you need to change both start and end points of the bar in the same time, you can call the AddBar method with the new coordinates, same item and key. The exBarParent changes the bar's parent. Use the exBarCanMoveToAnother option to specify whether the user can move a bar from one item to another by drag and drop. *(Float expression)*
exBarStartPrev	515	Retrieves the starting date of the bar before changing it ie if the user moves or resizes the bar at runtime, the exBarStartPrev gives during the BarResize event the previously starting date of the bar. *(Date expression)*
exBarEndPrev	516	Retrieves the ending date of the bar before changing it ie if the user moves or resizes the bar at runtime, the exBarEndPrev gives during the BarResize event the previously starting date of the bar.

		(Date expression)
exBarDurationPrev	517	Retrieves the duration or length of the bar before resizing it. During the BarResize event the exBarDurationPrev gives the length or duration of the bar before resizing it. You can distingue moving or resizing a specified bar by comparing the exBarDuration and exBarDurationPrev values. (Float expression)
exBarPercent100	518	 Specifies the percent from the original bar where the progress bar is displayed. Specifies the percent to display the progress on the bar, between 0 and 100. The exBarPercent100 option does the same thing as exBarPercent excepts that it works with integer values between 0 and 100, instead float expression from 0 to 1. Use the Add("A%B") to add a combination of two bars, so the exBarPercent value specifies the percent from the bar A to be displayed as bar B. For instance, the *Add("Task%Progress")* adds a combination of Task and Progress bars, so the Task shape is displayed on the full bar, and the Progress shape is displayed only on the portion determined by the exBarPercent100 value. When you resize the original bar (A), the inside bar (B) is shown proportionally. For instance, you can use the exBarPercent100 to display and edit values in the cells, when this property is associated with the cell using the AllowCellValueToItemBar property. *The 0 value corresponds to the exBarStart, while 100 corresponds to the exBarEnd, so the formula Items.ItemBar(exBarStart) + (Items.ItemBar(exBarEnd)- Items.ItemBar(exBarStart))*Items.ItemBar(exBarPer gives exactly the date time where progress bar is in the chart.* *(Long expression, between 0 and 100)*

exBarIntersectWith	519	Gets a collection of bars that intersect with the current bar. Use the exBarIntersectWithCount property to retrieve only the count of intersected bars. The result of exBarIntersectWith property is a collection of VARIANT values that indicates the keys of the bar in the current item that intersects with the current bar. The /NET or /WPF version provides a template function as public virtual List<object> get_BarIntersectWith(int Item, object Key) that returns a list of keys. The IntersectBars property specifies whether two bars intersect if returns 0, if 1 A is before B and -1 if A is after bar B. The exBarIntersectWith, exBarIntersectWithAsString, exBarIntersectWithCount checks all bars in the same item, of the same type or any other type indicated by the bar's OverlaidGroup property. The following screen shot shows the intersection of the bars of type TaskA:  *(Safe array of variant expression)*
exBarIntersectWithAsString	520	Gets a collection of bars that intersect with the current bar as string. The list is separated by comma character. The exBarIntersectWith, exBarIntersectWithAsString, exBarIntersectWithCount checks all bars in the same item, of the same type or any other type indicated by the bar's OverlaidGroup property. *(String expression)*
		Specifies the number of bars that intersects with the current bar. The IntersectBars property specifies whether two bars intersect if returns 0, if

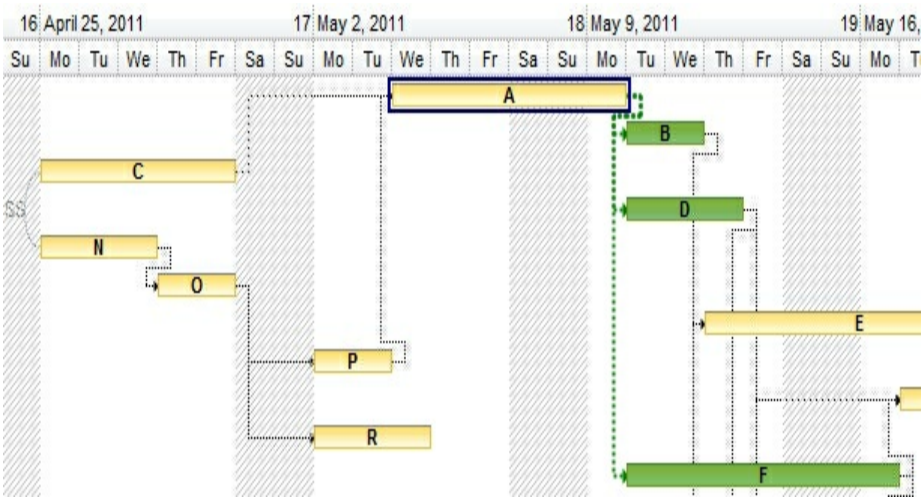
exBarIntersectWithCount	521	1 A is before B and -1 if A is after bar B. The exBarIntersectWith, exBarIntersectWithAsString, exBarIntersectWithCount checks all bars in the same item, of the same type or any other type indicated by the bar's OverlaidGroup property. *(Long expression)*
exBarsGroup	522	Retrieves a collection of bars being grouped with the current bar. The result of exBarsGroup property is a collection of VARIANT values that indicates the handle of the item and the key of the bars being grouped with specified bar. The GroupBars method group two bars, while the Use the UngroupBars method to ungroup two bars or all bars. The /NET or /WPF version provides a template function as public virtual List<SelectedBar> get_BarsGroup(int Item, object Key) that returns a list of bars. The SelectedBar structure provides two members the Item and the Key, where the Item indicates the handle of the item that hosts the bar, while the Key indicates the key of the bar. The following VB/NET sample displays the bars being grouped with selected bar(s): `With Exg2antt1.Items` `  Dim sSelected As List(Of exontrol.EXG2ANTTLib.Items.SelectedBar) = .get_SelectedBars()` `  If Not sSelected Is Nothing Then` `  Dim s As exontrol.EXG2ANTTLib.Items.SelectedBar` `  For Each s In sSelected` `  Debug.Print(s.Key & " from " & .get_CellCaption(s.Item, 0))` `  **Dim sGrouped As List(Of exontrol.EXG2ANTTLib.Items.SelectedBar) = .get_BarsGroup(s.Item, s.Key)**` `  If Not sGrouped Is Nothing Then` `  Dim b As`

```
exontrol.EXG2ANTTLib.Items.SelectedBar
            For Each b In sGrouped
                Debug.Print(vbTab & b.Key & " from "
& .get_CellCaption(b.Item, 0))
            Next
          End If
       Next
    End If
End With
```

(Safe array of variant expression)

exBarOutgoingLinks	523	Retrieves a collection of links that start from the current bar. The element in the collection indicates the name of the link that starts from the current bar. The Items.ItemBar(exBarOutgoingLinks) and Items.ItemBar(exBarOutgoingLinksAll) gives a collection of links that start on the current bar, including its descendents. The exBarOutgoingLinksAsString and exBarOutgoingLinksAllAsString gives the same information excepts that the result is returned as a string. The Link property can be used to access the link properties. The following sample displays all links that starts directly from the bar on the cursor: `With G2antt1.Items` `  Dim h As HITEM, c As Long, hit As HitTestInfoEnum` `  h = G2antt1.ItemFromPoint(-1, -1, c, hit)` `  Dim l As Variant` `  For Each l In .ItemBar(h, G2antt1.Chart.BarFromPoint(-1, -1), exBarOutgoingLinks)` `    Debug.Print "Link: " & l` `  Next` `End With`

The following sample shows the outgoing bars/links in green (B, D, F), when the current bar is A:



(Safe array of variant expression)

exBarOutgoingLinksAsString	524	Retrieves the links that start from the current bar, as string, separated by , (comma character). The Link property can be used to access the link properties. The exBarOutgoingLinksAllAsString option gives the list of links that starts from the current bar, including the descendents. The following sample displays the list of links that starts from the bar on the cursor: `With G2antt1.Items` `  Dim h As HITEM, c As Long, hit As HitTestInfoEnum` `  h = G2antt1.ItemFromPoint(-1, -1, c, hit)` `  Debug.Print .ItemBar(h, G2antt1.Chart.BarFromPoint(-1, -1), exBarOutgoingLinksAsString)` `End With` *(String expression)*
		Retrieves a collection of links that start from the current bar, including the descendents. The element in the collection indicates the name of the link that starts from the current bar. The Link property can

		be used to access the link properties. The following sample displays all links that starts from the bar on the cursor, including the descendents: `With G2antt1.Items` `  Dim h As HITEM, c As Long, hit As HitTestInfoEnum` `  h = G2antt1.ItemFromPoint(-1, -1, c, hit)` `  Dim l As Variant` `  For Each l In .ItemBar(h, G2antt1.Chart.BarFromPoint(-1, -1), exBarOutgoingLinksAll)` `  Debug.Print "Link: " & l` `  Next` `End With` *(Safe array of variant expression)*
exBarOutgoingLinksAll	525	
exBarOutgoingLinksAllAsString	526	Retrieves a collection of links that start from the current bar, including the descendents, as string. The Link property can be used to access the link properties. The following sample displays the list of links that starts from the bar on the cursor: `With G2antt1.Items` `  Dim h As HITEM, c As Long, hit As HitTestInfoEnum` `  h = G2antt1.ItemFromPoint(-1, -1, c, hit)` `  Debug.Print .ItemBar(h, G2antt1.Chart.BarFromPoint(-1, -1), exBarOutgoingLinksAllAsString)` `End With` *(String expression)*

Retrieves a collection of outgoing bars from the current bar. The element in the collection is a string that indicates the handle of the item, the : character, and the key of the bar, aka "78253912:newbar". The exBarOutgoingBars gives a collection of bars that are linked with the current bar. You can use the Split method to decompose the element to get the handle and the key, and so you can use the Items.ItemBar(Item,Key) to access the properties of the outgoing bar.

The following sample displays the bars that are linked with a FS link from the bar on the cursor:

exBarOutgoingBars 527

```
With G2antt1.Items
   Dim h As HITEM, c As Long, hit As
HitTestInfoEnum
   h = G2antt1.ItemFromPoint(-1, -1, c, hit)
   Dim b As Variant
   For Each b In .ItemBar(h,
G2antt1.Chart.BarFromPoint(-1, -1),
exBarOutgoingBars)
      Debug.Print "Outgoing Bar: " & b
   Next
End With
```

(Safe array of variant expression)

Retrieves a collection of outgoing bars from the current bar, as string. The exBarOutgoingBarsAsString displays the list of outgoing bar aka "79225544:newbar,79229032:newbar". The elements in the collections are separated by a , (comma) character. You can use the exBarOutgoingBarsDebug for debugging purpose to get displayed the caption on the hierarchy column instead displaying the handle of the item.

The following sample displays the list of outgoing bars:

exBarOutgoingBarsAsString 528

```
With G2antt1.Items
   Dim h As HITEM, c As Long, hit As
HitTestInfoEnum
   h = G2antt1.ItemFromPoint(-1, -1, c, hit)
   Debug.Print .ItemBar(h,
G2antt1.Chart.BarFromPoint(-1, -1),
exBarOutgoingBarsAsString)
End With
```

(String expression)

exBarOutgoingBarsAll 529

Retrieves a collection of all outgoing bars (including descendents) from the current bar. The element in the collection is a string that indicates the handle of the item, the : character, and the key of the bar, aka "78253912:newbar", where the 78253912 is the handle of the item and the newbar is the key of the bar. The exBarOutgoingBars gives a collection of bars that are linked with the current bar. You can use the Split method to decompose the element to get the handle and the key, and so you can use the Items.ItemBar(Item,Key) to access the properties of the outgoing bar.

The following sample displays all bars (including descendents) that are linked with a FS link from the bar on the cursor:

```
With G2antt1.Items
   Dim h As HITEM, c As Long, hit As
HitTestInfoEnum
   h = G2antt1.ItemFromPoint(-1, -1, c, hit)
   Dim b As Variant
   For Each b In .ItemBar(h,
G2antt1.Chart.BarFromPoint(-1, -1),
exBarOutgoingBarsAll)
      Debug.Print "Outgoing Bar: " & b
   Next
```

```
End With
```

(Safe array of variant expression)

exBarOutgoingBarsAllAsString530

Retrieves a collection of all outgoing bars (including descendents) from the current bar, as string. The exBarOutgoingBarsAllAsString displays the list of outgoing bars aka "79225544:newbar,79229032:newbar". The elements in the collections are separated by a , (comma) character. You can use the exBarOutgoingBarsAllDebug for debugging purpose to get displayed the caption on the hierarchy column instead displaying the handle of the item.

The following sample displays the list of outgoing bars:

```
With G2antt1.Items
   Dim h As HITEM, c As Long, hit As
HitTestInfoEnum
   h = G2antt1.ItemFromPoint(-1, -1, c, hit)
   Debug.Print .ItemBar(h,
G2antt1.Chart.BarFromPoint(-1, -1),
exBarOutgoingBarsAllAsString)
End With
```

(String expression)

Retrieves a collection of outgoing bars from the current bar, as string (the bar is indicated using the caption of the tree/hierarchy column, or TreeColumnIndex column). You can use this option for debugging purpose, instead exBarOutgoingBarsAsString, so instead displaying the handle of the item, the caption of the column is being displayed aka "Item 4:newbar,Item 5:newbar,Item 3:newbar". The element in the collection is separated by , (comma) character.

The following sample displays the direct outgoing

exBarOutgoingBarsDebug	531	bars, for debugging purpose: `With G2antt1.Items` `  Dim h As HITEM, c As Long, hit As HitTestInfoEnum` `  h = G2antt1.ItemFromPoint(-1, -1, c, hit)` `  Debug.Print .ItemBar(h, G2antt1.Chart.BarFromPoint(-1, -1), exBarOutgoingBarsDebug)` `End With` *(String expression)*
exBarOutgoingBarsAllDebug	532	Retrieves a collection of of all outgoing bars (including descendents) from the current bar, as string (the bar is indicated using the caption of the tree/hierarchy column, or TreeColumnIndex column). You can use this option for debugging purpose, instead exBarOutgoingBarsAllAsString, so instead displaying the handle of the item, the caption of the column is being displayed aka "Item 4:newbar,Item 5:newbar,Item 3:newbar". The element in the collection is separated by , (comma) character. The following sample displays the outgoing bars (including descendents), for debugging purpose: `With G2antt1.Items` `  Dim h As HITEM, c As Long, hit As HitTestInfoEnum` `  h = G2antt1.ItemFromPoint(-1, -1, c, hit)` `  Debug.Print .ItemBar(h, G2antt1.Chart.BarFromPoint(-1, -1), exBarOutgoingBarsAllDebug)` `End With` *(String expression)*
		Retrieves a collection of links that end on the

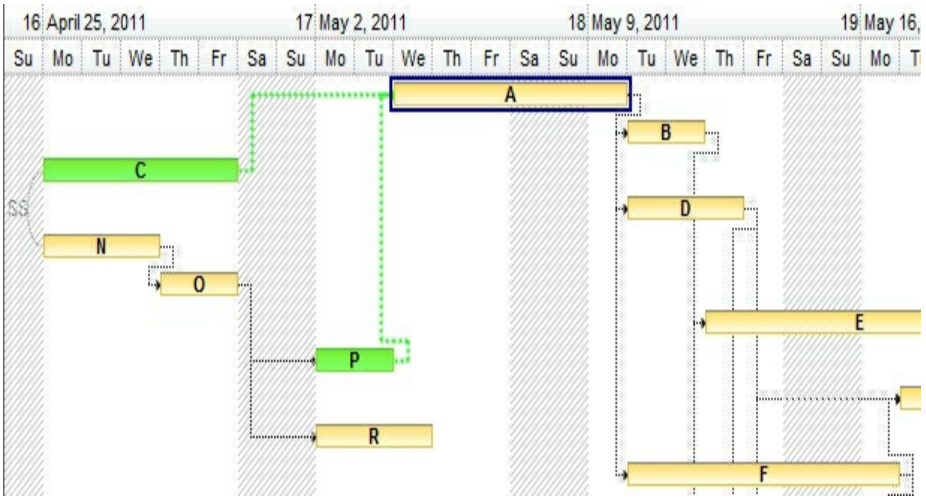
current bar. The element in the collection indicates the name of the link that ends on the current bar. The Items.ItemBar(exBarIncomingLinks) and Items.ItemBar(exBarIncomingLinksAll) gives a collection of links that ends on the current bar, including its ascendants. The exBarIncomingLinksAsString and exBarIncomingLinksAllAsString gives the same information excepts that the result is returned as a string. The Link property can be used to access the link properties.

The following sample displays all links that ends directly on the bar from the cursor:

exBarIncomingLinks 533

```
With G2antt1.Items
   Dim h As HITEM, c As Long, hit As
HitTestInfoEnum
   h = G2antt1.ItemFromPoint(-1, -1, c, hit)
   Dim l As Variant
   For Each l In .ItemBar(h,
G2antt1.Chart.BarFromPoint(-1, -1),
exBarIncomingLinks)
      Debug.Print "Incomming Link: " & l
   Next
End With
```

The following sample shows the incoming bars/links in green (C, P), when the current bar is A:



(Safe array of variant expression)

exBarIncomingLinksAsString	534	Retrieves a collection of links that end on the current bar, as string. The elements in the collection are separated by , (comma) character. The Link property can be used to access the link properties. The exBarIncomingLinksAsString option gives the list of links that ends on the current bar, including the ascendants. The following sample displays the list of links that ends on the bar from the cursor: `With G2antt1.Items` `  Dim h As HITEM, c As Long, hit As HitTestInfoEnum` `  h = G2antt1.ItemFromPoint(-1, -1, c, hit)` `  Debug.Print .ItemBar(h, G2antt1.Chart.BarFromPoint(-1, -1), exBarIncomingLinksAsString)` `End With` *(String expression)*
exBarIncomingLinksAll	535	Retrieves a collection of links that end on the current bar, including the ascendants. The element in the collection indicates the name of the link that ends on the current bar. The Link property can be used to access the link properties. The following sample displays all links that ends on the bar from the cursor, including the ascendants: `With G2antt1.Items` `  Dim h As HITEM, c As Long, hit As HitTestInfoEnum` `  h = G2antt1.ItemFromPoint(-1, -1, c, hit)` `  Dim l As Variant` `  For Each l In .ItemBar(h, G2antt1.Chart.BarFromPoint(-1, -1), exBarIncomingLinksAll)`

```
    Debug.Print "Incomming Link: " & l
  Next
End With
```

(Safe array of variant expression)

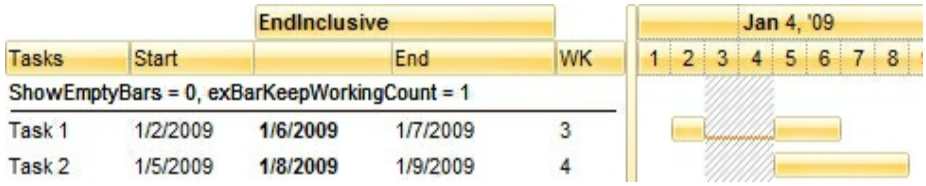
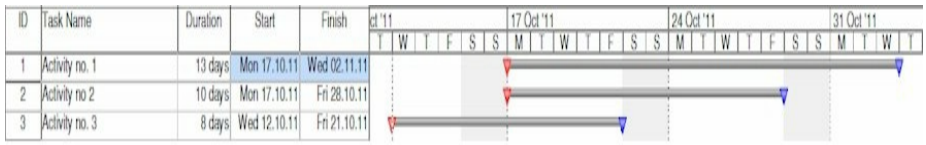
exBarIncomingLinksAllAsString 536	Retrieves a collection of links that end on the current bar, including the ascendants, as string. The Link property can be used to access the link properties. The following sample displays the list of links that ends on the bar from the cursor: `With G2antt1.Items` `  Dim h As HITEM, c As Long, hit As HitTestInfoEnum` `  h = G2antt1.ItemFromPoint(-1, -1, c, hit)` `  Debug.Print .ItemBar(h, G2antt1.Chart.BarFromPoint(-1, -1), exBarIncomingLinksAllAsString)` `End With` *(String expression)*
	Retrieves a collection of incoming bars from the current bar. The element in the collection is a string that indicates the handle of the item, the : character, and the key of the bar, aka "78253912:newbar". The exBarOutgoingBars gives a collection of bars that are linked with the current bar. You can use the Split method to decompose the element to get the handle and the key, and so you can use the Items.ItemBar(Item,Key) to access the properties of the incoming bar. The following sample displays the bars that are linked with a SF link from the bar on the cursor:

exBarIncomingBars	537	With G2antt1.Items Dim h As HITEM, c As Long, hit As HitTestInfoEnum h = G2antt1.ItemFromPoint(-1, -1, c, hit) Dim b As Variant For Each b In .ItemBar(h, G2antt1.Chart.BarFromPoint(-1, -1), exBarIncomingBars) Debug.Print "Outgoing Bar: " & b Next End With *(Safe array of variant expression)*
exBarIncomingBarsAsString	538	Retrieves a collection of incoming bars from the current bar, as string. The exBarIncomingBarsAsString displays the list of outgoing bar aka "79225544:newbar,79229032:newbar". The elements in the collections are separated by a , (comma) character. You can use the exBarIncomingBarsDebug for debugging purpose to get displayed the caption on the hierarchy column instead displaying the handle of the item. The following sample displays the list of incoming bars: With G2antt1.Items Dim h As HITEM, c As Long, hit As HitTestInfoEnum h = G2antt1.ItemFromPoint(-1, -1, c, hit) Debug.Print .ItemBar(h, G2antt1.Chart.BarFromPoint(-1, -1), exBarIncomingBarsAsString) End With *(String expression)*

exBarIncomingBarsAll	539	Retrieves a collection of all incoming bars (including ascendants) from the current bar. The element in the collection is a string that indicates the handle of the item, the : character, and the key of the bar, aka "78253912:newbar", where the 78253912 is the handle of the item and the newbar is the key of the bar. The exBarIncomingBarsAll gives a collection of bars that are linked with the current bar. You can use the Split method to decompose the element to get the handle and the key, and so you can use the Items.ItemBar(Item,Key) to access the properties of the incoming bar. The following sample displays all bars (including descendents) that are linked with a SF link from the bar on the cursor: `With G2antt1.Items` `  Dim h As HITEM, c As Long, hit As HitTestInfoEnum` `  h = G2antt1.ItemFromPoint(-1, -1, c, hit)` `  Dim b As Variant` `  For Each b In .ItemBar(h, G2antt1.Chart.BarFromPoint(-1, -1), exBarIncomingBarsAll)` `  Debug.Print "Outgoing Bar: " & b` `  Next` `End With` *(Safe array of variant expression)*
		Retrieves a collection of all incoming bars (including ascendants) from the current bar, as string. The exBarIncomingBarsAllAsString displays the list of incoming bars aka "79225544:newbar,79229032:newbar". The elements in the collections are separated by a , (comma) character. You can use the exBarIncomingBarsAllDebug for debugging purpose to get displayed the caption on the hierarchy column

		instead displaying the handle of the item. The following sample displays the list of incoming bars: `With G2antt1.Items` `  Dim h As HITEM, c As Long, hit As HitTestInfoEnum` `  h = G2antt1.ItemFromPoint(-1, -1, c, hit)` `  Debug.Print .ItemBar(h, G2antt1.Chart.BarFromPoint(-1, -1), exBarIncomingBarsAllAsString)` `End With` *(String expression)*
exBarIncomingBarsAllAsString	540	
exBarIncomingBarsDebug	541	Retrieves a collection of incoming bars from the current bar, as string (the bar is indicated using the caption of the tree/hierarchy column, or TreeColumnIndex column). You can use this option for debugging purpose, instead exBarIncomingBarsAsString, so instead displaying the handle of the item, the caption of the column is being displayed aka "Item 4:newbar,Item 5:newbar,Item 3:newbar". The element in the collection is separated by , (comma) character. The following sample displays the direct incoming bars, for debugging purpose: `With G2antt1.Items` `  Dim h As HITEM, c As Long, hit As HitTestInfoEnum` `  h = G2antt1.ItemFromPoint(-1, -1, c, hit)` `  Debug.Print .ItemBar(h, G2antt1.Chart.BarFromPoint(-1, -1), exBarIncomingBarsDebug)` `End With` *(String expression)*

exBarIncomingBarsAllDebug	542	Retrieves a collection of of all incoming bars (including ascendants) from the current bar, as string (the bar is indicated using the caption of the tree/hierarchy column, or TreeColumnIndex column). You can use this option for debugging purpose, instead exBarIncomingBarsAllAsString, so instead displaying the handle of the item, the caption of the column is being displayed aka "Item 4:newbar,Item 5:newbar,Item 3:newbar". The element in the collection is separated by , (comma) character. The following sample displays the incoming bars (including ascendants), for debugging purpose: `With G2antt1.Items` `  Dim h As HITEM, c As Long, hit As HitTestInfoEnum` `  h = G2antt1.ItemFromPoint(-1, -1, c, hit)` `  Debug.Print .ItemBar(h, G2antt1.Chart.BarFromPoint(-1, -1), exBarIncomingBarsAllDebug)` `End With` *(String expression)*
		Retrieves or sets a value that indicates the inclusive ending point of the bar. Generally, the inclusive ending point of the bar is exBarEnd - 1. You can use the exBarEndInclusive to display exBarEnd - 1 when associate a cell with a bar, using the AllowCellValueToItemBar property, so the ending point displayed on the list section is one day less. The exBarEndInclusive option was designed to be used when the chart's unit scale is exDay, so days are being displayed. *Changing the exBarEnd value may change the exBarEndInclusive value, or reverse.* For instance, a a task bar from 1/1/2001 to 1/3/2001 shows two days, the exBarEnd displays 1/3/2001, while the exBarEndInclusive displays 1/2/2001.

exBarEndInclusive	543	The following screen shot shows the values of exBarEnd and exBarEndInclusive displayed on End and EndInclusive columns:  In the previously picture you can notice that the exBarEnd of the Task 1 is 1/7/2009, while the exBarEndInclusive indicates the 1/6/2009. Using the exBarEndInclusive you can show and edit the finish column like shown in the following picture:  *(Date expression)*
exBarMoveStart	544	Retrieves or sets (by preserving the bar's length/duration) a value that indicates the start of the bar. This property returns the same value as exBarStart property. If used to change the starting point of the bar, the exBarMoveStart property moves the bar, while the exBarStart property resizes the bar, or in other words, the exBarMoveStart moves the starting point of the bar, by preserving the bar's length/duration. For instance, this property can be associated with a column of drop down calendar type, so once the user changes the date in the calendar, the associated bar is moved to start at selected date. *(Date expression)*
		Retrieves or sets (by preserving the bar's length/duration) a value that indicates the end of the bar. This property returns the same value as exBarEnd property. If used to change the ending

exBarMoveEnd	545	point of the bar, the exBarMoveEnd property moves the bar, while the exBarEnd property resizes the bar, or in other words, the exBarMoveEnd moves the ending point of the bar, by preserving the bar's length/duration. For instance, this property can be associated with a column of drop down calendar type, so once the user changes the date in the calendar, the associated bar is moved to end at selected date. *(Date expression)*
exBarMoveEndInclusive	546	Retrieves or sets (by preserving the bar's length/duration) a value that indicates the inclusive ending point of the bar. Generally, the inclusive ending point of the bar is exBarEnd - 1. This property returns the same value as exBarEndInclusive property. If used to change the inclusive ending point of the bar, the exBarMoveEndInclusive property moves the bar, while the exBarEndInclusive property resizes the bar, or in other words, the exBarMoveEndInclusive moves the inclusive ending point of the bar, by preserving the bar's length/duration. For instance, this property can be associated with a column of drop down calendar type, so once the user changes the date in the calendar, the associated bar is moved to end at selected date. *(Date expression)*

constants ItemsAllowSizingEnum

The ItemsAllowSizingEnum type specifies whether the user can resize items individuals or all items at once, at runtime. Use the ItemsAllowSizing property to specify whether the user can resize items individuals or all items at once, at runtime. Currently, the ItemsAllowSizingEnum type supports the following values:

Name	Value	Description
exNoSizing	0	The user can't resize the items at runtime.
exResizeItem	-1	Specifies whether the user resizes the item from the cursor.
exResizeAllItems	1	Specifies whether the user resizes all items at runtime.

constants UIVisualThemeEnum

The UIVisualThemeEnum expression specifies the UI parts that the control can shown using the current visual theme. The UseVisualTheme property specifies whether the UI parts of the control are displayed using the current visual theme.

Name	Value	Description
exNoVisualTheme	0	exNoVisualTheme
exDefaultVisualTheme	16777215	exDefaultVisualTheme
exHeaderVisualTheme	1	exHeaderVisualTheme
exFilterBarVisualTheme	2	exFilterBarVisualTheme
exButtonsVisualTheme	4	exButtonsVisualTheme
exCalendarVisualTheme	8	exCalendarVisualTheme
exSliderVisualTheme	16	exSliderVisualTheme
exSpinVisualTheme	32	exSpinVisualTheme
exCheckBoxVisualTheme	64	exCheckBoxVisualTheme
exProgressVisualTheme	128	exProgressVisualTheme
exCalculatorVisualTheme	256	exCalculatorVisualTheme

constants UnitEnum

The UnitEnum type specifies the time units supported. Use the UnitScale property to specify the time scale. Use the Unit property to specify the time unit in the level. The UnitEnum type includes the following time units:

Name	Value	Description
exYear	0	Indicates the year. Values: ..., 2001, 2002, 2003, ...
exHalfYear	1	A date between January 1st and June 31 indicates the first half of the year, and from July 1 to December 31, indicates the second half of the year. Values: 1 and 2
exQuarterYear	2	A date between January 1st and March 31 indicates the first quarter of the year, a date between April 1st and June 30 indicates the second quarter of the year, a date between July 1st and September 30 indicates the third quarter of the year, and if a date between October 1st and December 31 indicates the forth quarter of the year. Values: 1, 2, 3 and 4
exMonth	16	Indicates the month. Values: 1 (January), 2 (February), ..., and 12 (December). Use the MonthNames property to specify the name of the months.
exThirdMonth	17	The first ten days in a month indicates the first third of the month, the next 10 days indicates the second third of the month, and the last 10 days in the month indicates the last third of the month. Values: 1, 2 and 3.
exWeek	256	Indicates the week in the year. Values: 1,2,...,53. Use the WeekDays property to specify the name of the days in the week.
exDay	4096	Indicates the day of the date. Values: 1,2,..,31
exHour	65536	Indicates the hour.
exMinute	1048576	Indicates the minute.
exSecond	16777216	Indicates the second.

constants ValidateValueType

The ValidateValueType specifies the type of validation that control supports. The CauseValidateValue property specifies whether the ValidateValue event is fired before Change event, so the user can validate the values being entered. The ValidateValue event is not fired if the CauseValidateValue property is False ~ exNoValidate. The ValidateValue event is fired once the user tries to leaves the focused cell (exValidateCell) or focused item (exValidateItem). The ValidateValueType enumeration supports the following values:

Name	Value	Description
exValidateCell	-1	The ValidateValue event is called just before leaving the cell. Use this option to validate the values per cell.
exNoValidate	0	The ValidateValue event is not fired.
exValidateItem	1	The ValidateValue event is fired when the user leaves the focused item. Use this option to validate the values per item.

constants VAlignmentEnum

Specifies the source's vertical alignment.

Name	Value	Description
exTop	0	The source is aligned to the top.
exMiddle	1	The source is centered.
exBottom	2	The source is aligned to the bottom.
exVOutside	16	The object is displayed outside of the source.

constants ValueFormatEnum

Defines how the cell's value is shown. The CellValueFormat property indicates the way the cell displays its content. The Def(exCellValueFormat) property indicates the format for all cells within the column. The CellValue property indicates the cell's value, content or formula. The ComputedField property indicates the formula to compute all cells in the column. The FormatColumn property indicates the format to be applied for cells in the columns.The ValueFormatEnum type supports can be a combination of the following values:

Name	Value	Description
exText	0	No HTML tags are painted
		Currently, the Exontrol's built-in HTML format supports the following HTML tags: • ** ... ** displays the text in **bold** • ** ... ** displays the text in *italics* • **<u> ... </u>** underlines the text • **<s> ... </s>** ~~Strike~~-through text • **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements. The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using <a ;exp=> or <a ;e64=> anchor tags. The exp/e64 field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption. ◦ exp, stores the plain text to be shown once the user clicks the anchor, such as <a ;exp=show lines> ◦ e64, encodes in BASE64 the HTML text to

be shown once the user clicks the anchor, such as <a ;e64=gA8ABmABnABjABvABshIAOQAEAA that displays show lines- in gray when the user clicks the + anchor. The gA8ABmABnABjABvABshIAOQAEAAHAAG string encodes the <fgcolor 808080>show lines<a>-</fgcolor> The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, <solidline> Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3 shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The show lines is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** displays portions of text with a different font and/or different size. For instance, the bit draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, bit displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or

exHTML	1	<solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values. • **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values. • **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>). • **<r>** right aligns the text • **<c>** centers the text • ** ** forces a line-break • **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels. • **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter

indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.

- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: Text with <**off** 6>subscript displays the text such as: Text with $_{\text{subscript}}$ The Text with <**off** -6>superscript displays the text such as: Text with $^{\text{subscript}}$
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the <**gra** FFFFFF;1;1>gradient-center</**gra**> generates the following picture:

- **<out rrggbb;width> ... </out>** shows the text

with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the <**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**> generates the following picture:

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the <**sha**>shadow</**sha**> generates the following picture:

or *<**sha** 404040;5;0> <fgcolor=FFFFFF>outline anti-aliasing</fgcolor></**sha**>* gets:

For instance, the following HTML caption

```
<font Segoe Print>This is a bit of text with a
<b>different</b> font</font>
<upline><dotline>left 1<r><b>right</b>
2
<img>1</img><c><a><s>center
<img>pic1:64</img> picture</s><r></a>
```

```
<img>2</img>
left 3 <c>center<r><b>right</b> 4
```

generates the following screen shot:

exComputedField	2	Indicates a computed field. The CellValue property indicates the formula to compute the field. A computed field can display its content using the values from any other cell in the same item/row. For instance %1 + %2 indicates that the cell displays the addition from the second and third cells in the same item (cells are 0 based). For instance, if the cells are of numeric format the result is the sum of two values, while if any of the cell is of string type it performs a concatenation of the specified cells. The ComputedField property indicates the formula to compute all cells in the column. The exComputedField can be combined with exText or exHTML. For instance, the exComputedField + exHTML indicates that the computed field may display HTML tags. The syntax for the CellValue property should be: **formula** where %n indicates the cell from the n-index. The operation being supported are listed bellow. For instance %1 + %2 indicates the sum of all cells in the second and third column from the current item.
		Indicates a total/subtotal field. The CellValue property indicates the formula for total field that includes an aggregate function such as: sum, min, max, count, avg. The exTotalField can be combined with exText or exHTML. For instance, the exTotalField + exHTML indicates that the total field

		may display HTML tags. The syntax for the CellValue property should be: **aggregate**(**list**,**direction**,**formula**) where: **aggregate** must be one of the following: • *sum* - calculates the sum of values. • *min* - retrieves the minimum value. • *max* - retrieves the maximum value. • *count* - counts the number of items. • *avg* - calculates the average of values. **list** must be one of the following: • a *long* expression that specifies the index of the item being referred. • a predefined string expression as follows: ◦ *all* - indicates all items, so the formula is being applied to all items. The direction has no effect. ◦ *current* - the current item. ◦ *parent* - the parent item. ◦ *root* - the root item. **direction** must be one of the following: • *dir* - collects the direct descendents. • *rec* - collects the leaf descendents (leaf items). A leaf item is an item with no child items. • *all* - collects all descendents. Currently, the following items are excluded by aggregate functions:
exTotalField	4	• *not-sortable items*. The SortableItem property specifies whether the item can be sorted (a sortable item can change its position after sorting, while a not-sortable item keeps its position after sorting. • *not-selectable items*. The SelectableItem property specifies whether the user can selects/focus the specified item. • *divider items*. The ItemDivider property

specifies whether the item displays a single cell, instead displaying whole cells.

In conclusion, aggregate functions counts ONLY items that are:

- *sortable*, SortableItem is True, by default.
- *selectable*, SelectableItem is True, by default.
- *not divider*, ItemDivider is -1, by default.

Shortly, by setting to a different value to any of these properties, makes the item to be ignored by the aggregate functions.

For instance

- count(current,dir,1) counts the number of child items (not implies recursively child items).
- count(current,all,1) counts the number of all child items (implies recursively child items).
- count(current,rec,1) counts the number of leaf items (implies recursively leaf items).
- count(current,rec,1) counts the number of leaf items (a leaf item is an item with no child items).
- sum(parent,dir,%1=0?0:1) counts the not-zero values in the second column (%1)
- sum(parent,dir,%1 + %2) indicates the sum of all cells in the second (%1) and third (%2) column that are directly descendent from the parent item.
- sum(all,rec,%1 + %2) sums all leaf cells in the second (%1) and third (%2) columns.

The **formula** on the CellValue property (if the CellValueFormat property indicates the exComputedField or exTotalField) may include the formatting operators as follows:

The expression supports cell's identifiers as follows:

- ***%0**, **%1**, **%2**, ... specifies the value of the cell in the column with the index 0, 1 2, ... The CellValue property specifies the cell's value. For instance, "%0 format ``" formats the value on the cell with the index 0, using current regional setting, while "int(%1)" converts the value of the column with the index 1, to integer.*

- *%C0, %C1, %C2, ... specifies the caption of the cell, or the string the cell displays in the column with the index 0, 1 2, ... The CellCaption property specifies the cell's caption. The cell's value may be different than what the cell displays as a string. For instance, let's say a cell display HTML format. The %0 returns the html format including the HTML tags, while %C0 returns the cell's content as string without HTML tags. For instance, "upper(%C1)" converts the caption of the cell with the index 1, to upper case, while "%C0 left 2" returns the leftmost two characters on the cell with the index 0.*
- *%CD0, %CD1, %CD2, ... specifies the cell's extra data in the column with the index 0, 1 2, ... The CellData property associates any extra/user data to a cell. For instance, "%CD0 = `your user data`" specifies all cells whose CellData property is `your user data`, on the column with the index 0.*
- *%CS0, %CS1, %CS2, ... specifies the cell's state in the column with the index 0, 1 2, ... The CellState property specifies the cell's state, and so it indicates whether the cell is checked or un-checked. For instance, "%CS0" defines all checked items on the column with the index 0, or "not %CS1" defines all un-checked items in the column with the index 1.*

This property/method supports predefined constants and operators/functions as described here.

Usage examples:

1. "**1**", the cell displays 1
2. "**%0 + %1**", the cell displays the sum between cells in the first and second columns.
3. "**%0 + %1 - %2**", the cell displays the sum between cells in the first and second columns minus the third column.
4. "**(%0 + %1)*0.19**", the cell displays the sum between cells in the first and second columns multiplied with 0.19.
5. "**(%0 + %1 + %2)/3**", the cell displays the arithmetic average for the first three columns.
6. "**%0 + %1 < %2 + %3**", displays 1 if the sum between cells in the first two columns is less than the sum of third and forth columns.
7. "**proper(%0)**'" formats the cells by capitalizing first letter in each word
8. "**currency(%1)**'" displays the second column as currency using the format in the control panel for money
9. "**len(%0) ? currency(dbl(%0)) : ''**" displays the currency only for not empty/blank cells.
10. "**int(date(%1)-date(%2)) + 'D ' + round(24*(date(%1)-date(%2) - floor(date(%1)-date(%2)))) + 'H'**'" displays interval between two dates in days and hours, as xD yH
11. **"2:=((1:=int(0:= date(%1)-date(%0))) = 0 ? '' : str(=:1) + ' day(s)') + (3:=round(24*(=:0-floor(=:0))) ? (len(=:2) ? ' and ' : '') + =:3 + ' hour(s)' : '')"** displays the interval

between two dates, as x day(s) [and y hour(s)], where the x indicates the number of days, and y the number of hours. The hour part is missing, if 0 hours is displayed, or nothing is displayed if dates are identical.

constants WeekDayEnum

The WeekDayEnum type indicates the days in the week. The WeekDays property indicates the name of the days in the week. The WeekDayEnum type includes the following values.

Name	Value	Description
exSunday	0	Sunday
exMonday	1	Monday
exTuesday	2	Tuesday
exWednesday	3	Wednesday
exThursday	4	Thursday
exFriday	5	Friday
exSaturday	6	Saturday

constants WeekNumberAsEnum

The WeekNumberAsEnum type specifies the ways the control displays the week number for dates. The WeekNumberAs property specifies the way the control displays the week number. The FirstWeekDay property specifies the first day of the week where the week begins. The WeekNumberAsEnum type supports the following values:

Name	Value	Description
exISO8601WeekNumber	0	Indicates that the week number is displayed according to the ISO8601 standard, which specifies that the first week of the year is the one that includes the January the 4th
exSimpleWeekNumber	1	The first week starts on January 1st of a given year, week n+1 starts 7 days after week n (default)

constants ZoomOnFlyEnum

The ZoomOnFlyEnum type indicates the flags that can be combined to show the control's Zoom-OnFly view. The Zoom-OnFly view was provided to let you magnify a portion of the chart without affecting the chart's scale. The Zoom-OnFly view displays the item and its neighbors from the cursor, and additional information about the bar from the cursor. The Chart.AllowZoomOnFly property indicates whether the Zoom-OnFly view is shown once the user presses the CTRL or SHIFT key over a bar. The ZoomOnFlyEnum type supports the following values:

Name	Value	Description
exNoZoomOnFly	0	(Default) The Zoom-OnFly view is not available.
exZoomOnFlyShift	1	If this flag is present, the Zoom-OnFly view can be shown if pressing the SHIFT, not requiring the CTRL + SHIFT keys combination.
exZoomOnFlyCtrl	2	If this flag is present, the Zoom-OnFly view can be shown if pressing the CTRL, not requiring the CTRL + SHIFT keys combination.
exAllowRefineOnFly	8	Specifies whether the Zoom-OnFly view is visible when the user clicks the mouse.
exAllowInfoOnFly	16	Specifies whether the Zoom-OnFly view is visible when the user hovers the mouse.
exZoomOnFlyBarsOnly	32	Specifies whether the Zoom-OnFly view is visible if the cursor hovers a bar, and show nothing, if there is no bar at the cursor position.
exZoomOnFly	24	Specifies that the Zoom-OnFly view is visible when the user clicks or hovers the mouse on the chart area, while pressing the CTRL + SHIFT keys combination. For instance, if the Chart.AllowZoomOnFly property is exZoomOnFly + exZoomOnFlyShift (25), the view can be shown if the user presses the SHIFT, not requiring the CTRL + SHIFT keys combination.
exZoomOnFlyIncludeNeighborItems	256	Specifies whether the Zoom-OnFly view displays the neighbors items (previously visible item and next visible items). For instance, if the Chart.AllowZoomOnFly property is exZoomOnFly + exZoomOnFlyIncludeNeighborItems (280), the view shows the previously and next visible item of the one from the cursor. In other words, the view

	can show up to 3 items including the item from the cursor, the previously visible item and the next visible items. The last two are shown only if they exists. Use this option to display more than one item in the view for better alignment of the bars based on their neighbors.
exZoomOnFlyIncludeSelectedItems 768	Indicates that the view can show the item from the cursor, including the previously selected item and the next selected item. You can use this flag to show the current item among with any other selected item. Use exZoomOnFly + exZoomOnFlyIncludeSelectedItems (793) option to include the next and previously selected items for a better comparing between selection and the item from the cursor. For instance, click or selects an item to be compared and next go to other item, and so the view will include these items. Use the SelectOnClick property to prevent selecting a row / item when clicking the chart portion of the control.

Appearance object

The component lets the user changes its visual appearance using **skins**, each one providing an additional visual experience that enhances viewing pleasure. Skins are relatively easy to build and put on any part of the control. The Appearance object holds a collection of skins. The Appearance object supports the following properties and methods:

Name	Description
Add	Adds or replaces a skin object to the control.
Clear	Removes all skins in the control.
Remove	Removes a specific skin from the control.
RenderType	Specifies the way colored EBN objects are displayed on the component.

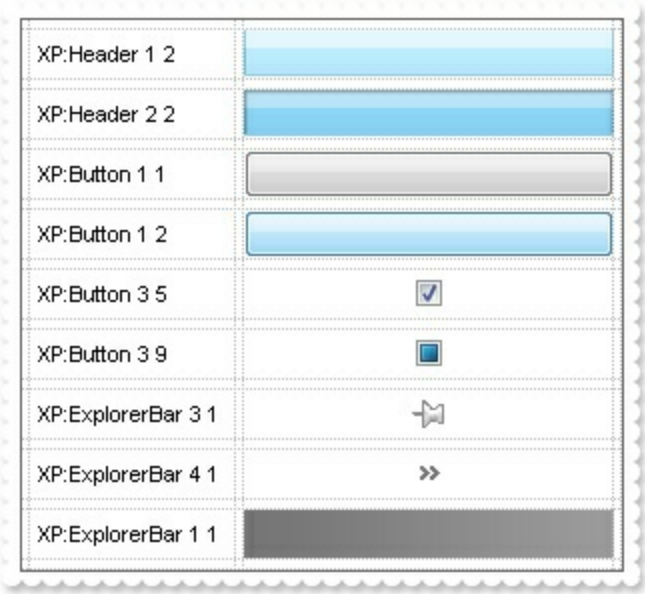
method Appearance.Add (ID as Long, Skin as Variant)

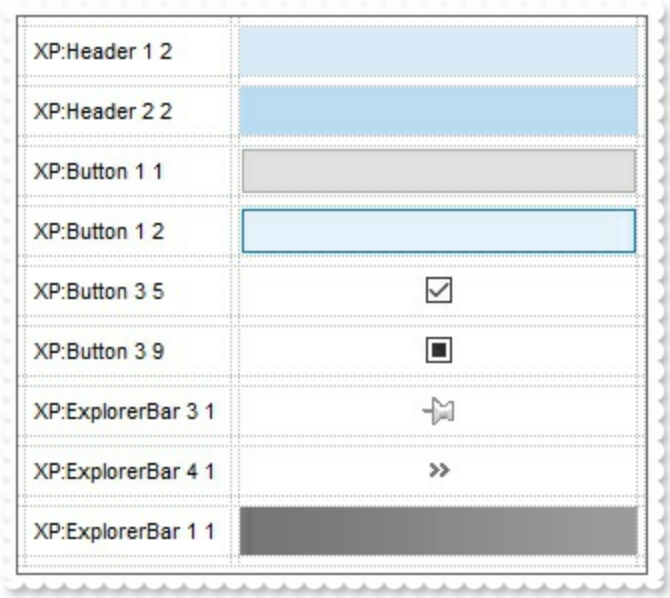
Adds or replaces a skin object to the control.

Type	Description
ID as Long	A Long expression that indicates the index of the skin being added or replaced. The value must be between 1 and 126, so Appearance collection should holds no more than 126 elements.
	The Skin parameter of the Add method can a STRING as explained bellow, a BYTE[] / safe arrays of VT_I1 or VT_UI1 expression that indicates the content of the EBN file. You can use the BYTE[] / safe arrays of VT_I1 or VT_UI1 option when using the EBN file directly in the resources of the project. For instance, the VB6 provides the LoadResData to get the safe array o bytes for specified resource, while in VB/NET or C# the internal class Resources provides definitions for all files being inserted. (ResourceManager.GetObject("ebn", resourceCulture)) If the Skin parameter points to a string expression, it can be one of the following: • A path to the skin file (*.EBN). The ExButton component or ExEBN tool can be used to create, view or edit EBN files. For instance, "C:\Program Files\Exontrol\ExButton\Sample\EBN\MSOffice-Ribbon\msor_frameh.ebn" • A BASE64 encoded string that holds the skin file (*.EBN). Use the ExImages tool to build BASE 64 encoded strings of the skin file (*.EBN). The BASE64 encoded string starts with "gBFLBCJw..." • An Windows XP theme part, if the Skin parameter starts with "**XP:**". Use this option, to display any UI element of the Current Windows XP Theme, on any part of the control. In this case, the syntax of the Skin parameter is: "XP:ClassName Part State" where the ClassName defines the window/control class name in the Windows XP Theme, the Part indicates a long expression that defines the part, and the State indicates the state of the part to be shown. All known values for window/class, part and start are defined at

the end of this document. For instance the "XP:Header 1 2" indicates the part 1 of the Header class in the state 2, in the current Windows XP theme.

The following screen shots show a few Windows XP Theme Elements, running on Windows Vista and Windows 10:



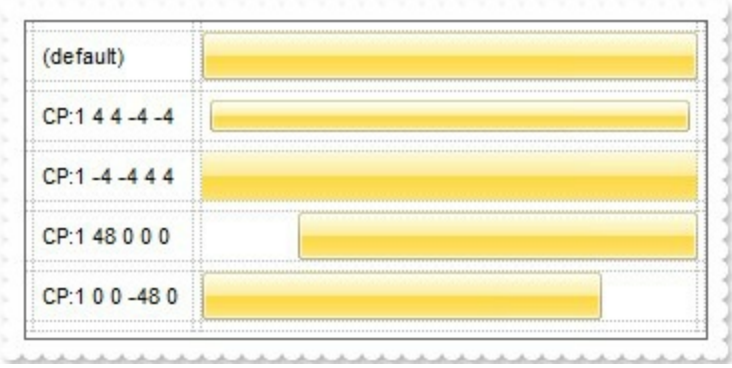


Skin as Variant

- A copy of another skin with different coordinates (position, size), if the Skin parameter starts with "**CP**:". Use this option, to display the EBN, using different coordinates (position, size). By default, the EBN skin object is rendered on the part's client area. Using this option, you can display the same EBN, on a different position / size. In this case, the syntax of the Skin parameter is: "CP:ID Left Top Right Bottom"

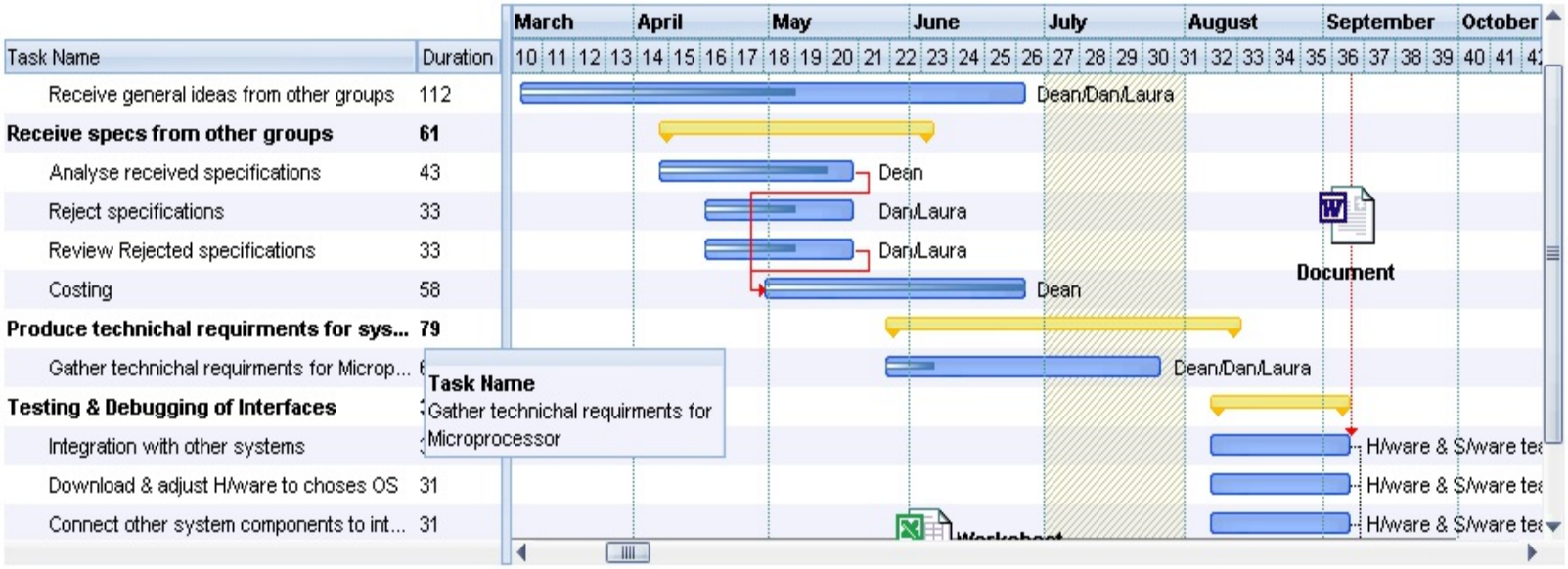
where the ID is the identifier of the EBN to be used (it is a number that specifies the ID parameter of the Add method), Left, Top, Right and Bottom parameters/numbers specifies the relative position to the part's client area, where the EBN should be rendered. The Left, Top, Right and Bottom parameters are numbers (negative, zero or positive values, with no decimal), that can be followed by the D character which indicates the value according to the current DPI settings. For instance, "CP:1 -2 -2 2 2", uses the EBN with the identifier 1, and displays it on a 2-pixels wider rectangle no matter of the DPI settings, while "CP:1 -2D -2D 2D 2D" displays it on a 2-pixels wider rectangle if DPI settings is 100%, and on on a 3-pixels wider rectangle if DPI settings is 150%.

The following screen shot shows the same EBN being displayed, using different CP: options:



Return	Description
Boolean	A Boolean expression that indicates whether the new skin was added or replaced.

Use the Add method to add or replace skins to the control. The skin method, in it's simplest form, uses a single graphic file (*.ebn) assigned to a part of the control. By using a collection of objects laid over the graphic, it is possible to define which sections of the graphic will be used as borders, corners and other possible elements, fixing them to their proper position regardless of the size of the part. Use the Remove method to remove a specific skin from the control. Use the Clear method to remove all skins in the control. Use the BeginUpdate and EndUpdate methods to maintain performance while init the control. Use the Refresh method to refresh the control.



The identifier you choose for the skin is very important to be used in the background properties like explained bellow. Shortly, the color properties uses 4 bytes (DWORD, double WORD, and so on) to hold a RGB value. More than that, the first byte (most significant byte in the color) is used only to specify system color. if the first bit in the byte is 1, the rest of bits indicates the index of the system color being used. So, we use the last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. So, since the 7 bits can cover 127 values, excluding 0, we have 126 possibilities to store an identifier in that byte. This way, a DWORD expression indicates the background color stored in RRGGBB format and the index of the skin (ID parameter) in the last 7 bits in the high significant byte of the color. For instance, the BackColor = BackColor Or &H2000000 indicates that we apply the skin with the index 2 using the old color, to the object that BackColor is applied.

The skin method may change the visual appearance for the following parts in the control:

- control's border, Appearance property
- **levels** on the chart area, BackColor property, BackColorLevelHeader property
- bar's background, ItemBar(exBarBackColor) property
- control's **header bar**, BackColorHeader property
- control's **filter bar**, FilterBarBackColor property
- control's **sort bar**, BackColorSort property
- the caption of the control's sort bar, BackColorSortCaption property
- **selected item** or cell, SelBackColor property
- **item**, ItemBackColor property
- **cell**, CellBackColor property
- **bars**, Color property
- cell's **button**, "**drop down**" filter bar button, "close" filter bar button, tooltips, and so on, Background property
- 'focus' box in the chart's overview area, OverviewSelBackColor property.

For instance, the following VB sample changes the visual appearance for the selected item. The SelBackColor property indicates the selection background color. Shortly, we need to add a skin to the Appearance object using the Add method, and we need to set the last 7 bits in the SelBackColor property to indicates the index of the skin that we want to use. The sample applies the "  " to the selected item(s):

```
With G2antt1
   With .VisualAppearance
      .Add &H23, App.Path + "\selected.ebn"
   End With
   .SelForeColor = RGB(0, 0, 0)
   .SelBackColor = &H23000000
End With
```

The sample adds the skin with the index 35 (Hexa 23), and applies to the selected item using the SelBackColor property.

The following C++ sample applies a new appearance to the selected item(s):

```
#include "Appearance.h"
m_g2antt.GetVisualAppearance().Add( 0x23,
COleVariant(_T("D:\\Temp\\ExG2antt_Help\\selected.ebn")) );
m_g2antt.SetSelBackColor( 0x23000000 );
m_g2antt.SetSelForeColor( 0 );
```

The following VB.NET sample applies a new appearance to the selected item(s):

```
With AxG2antt1
   With .VisualAppearance
      .Add(&H23, "D:\Temp\ExG2antt_Help\selected.ebn")
   End With
   .SelForeColor = Color.Black
   .Template = "SelBackColor = 587202560"
End With
```

The VB.NET sample uses the Template property to assign a new value to the SelBackColor property. The 587202560 value represents &23000000 in hexadecimal.

The following C# sample applies a new appearance to the selected item(s):

```
axG2antt1.VisualAppearance.Add(0x23, "D:\\Temp\\ExG2antt_Help\\selected.ebn");
```

```
axG2antt1.Template = "SelBackColor = 587202560";
```

The following VFP sample applies a new appearance to the selected item(s):

```
With thisform.G2antt1
   With .VisualAppearance
      .Add(35, "D:\Temp\ExG2antt_Help\selected.ebn")
   EndWith
   .SelForeColor = RGB(0, 0, 0)
   .SelBackColor = .587202560
EndWith
```

The 587202560 value represents &23000000 in hexadecimal. The 32 value represents &23 in hexadecimal

The screen shot was generated using the following template:

```
BeginUpdate

Images("gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjIgjIBAEijUlk8plUrlktl

Images("gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjIgjIBAEijUlk8plUrlktl

Images("gBJJgBggAAkGAAQhIAf8Nf4hhkOiRCJo2AEXjAAi0XFEYIEYhUXAIAEEZi8hk0plUrlkt

Images("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olUrlktl



Font
{
   Name = "Tahoma"
}

VisualAppearance
{
   ' Header

Add(1,"gBFLBCJwBAEHhEJAEGg4BawDg6AADACAxRDAMgBQKAAzQFAYZhoHKGAAGEYxR
```

```
    ' HeaderFilterBarButton

Add(2,"gBFLBCJwBAEHhEJAEGg4BAQEg6AADACAxRDAMgBQKAAzQFAYZhoHKGAAGEYxR

Add(3,"gBFLBCJwBAEHhEJAEGg4BBAEg6AADACAxRDAMgBQKAAzQFAYZhoHKGAAGEYxR

    ' SelectedItem
    Add(4,
"gBFLBCJwBAEHhEJAEGg4BV4Fg6AABACAxWgKBADQKAAyDIKsEQGGIZRhhGIwAgaFIXQK

    ' Marks a cell

Add(5,"gBFLBCJwBAEHhEJAEGg4BF4Gg6AABACAxWgKBADQKAAyDIKsEQGGIZRhhGIwAga

Add(6,"gBFLBCJwBAEHhEJAEGg4BaAFg6AADACAxRDAMgBQKAAzQFAYZhxBaERiGIZ4JhUA

}
BackColorHeader = 16777216          '0x01BBGGRR
BackColorSortBarCaption = 33488896     '0x01BBGGRR
FilterBarBackColor = 16777216       '0x01BBGGRR
Background(0) = 33554432            '0x02BBGGRR
Background(1) = 50331648            '0x03BBGGRR
Background(2) = 67108864            '0x04BBGGRR
Background(3) = 100663296           '0x06BBGGRR
Background(8) = 67108864            '0x04BBGGRR
Background(9) = 67108864            '0x04BBGGRR
Background(10) = 100663296             '0x06BBGGRR
Background(11) = 100663296             '0x06BBGGRR
Background(12) = 100663296             '0x06BBGGRR
Background(13) = 100663296             '0x06BBGGRR
Background(14) = 100663296             '0x06BBGGRR
Background(15) = 16777216           '0x01BBGGRR
SelBackColor = 67108864             '0x04BBGGRR
BackColorSortBar = RGB(61,101,183)
FilterBarForeColor = RGB(255,255,255)
```

```
ForeColorHeader = RGB(255,255,255)
ForeColorSortBar = RGB(255,255,255)
SelForeColor = 0

SortBarVisible = True
MarkSearchColumn = False
LinesAtRoot = 1
ForeColor = RGB(0,0,255)
BackColor = RGB(255,255,255)
BackColorLevelHeader = RGB(255,255,255)
DrawGridLines = -1
ScrollBySingleLine = True
HasLines = 2
HasButtons = 3
CheckImage(1) = 4
CheckImage(0) = 5
CheckImage(2) = 6
Chart
{
   DrawGridLines = -1
   BackColor = RGB(255,255,255)
   BackColorLevelHeader = 16777216      '0x01BBGGRR
   ForeColorLevelHeader = RGB(255,255,255)
   ScrollBar = False
   Bars
   {
      AddShapeCorner(1234,1)
      AddShapeCorner(1235,2)
      Add("Custom")
      {
         Color = RGB(255,0,0)
         Shape = 19
         Pattern = 2
         StartShape = 1234
         StartColor = RGB(255,0,0)
         EndShape = 1235
         EndColor = RGB(255,0,0)
```

```
      }
   }
}
Columns
{
   "Task"
   {
      HeaderBold = True
      DisplayFilterButton = True
      DisplayFilterDate = True
      Width = 196
   }
   1
   {
      AllowSizing = False
      HTMLCaption = "1 First"
      Def(0) = True
      LevelKey = 1
      Width = 25
      Alignment = 1
   }
   2
   {
      AllowSizing = False
      HTMLCaption = "2 Second"
      Def(0) = True
      LevelKey = 1
      Width = 25
      Alignment = 1
   }
   3
   {
      AllowSizing = False
      HTMLCaption = "3 Third"
      Def(0) = True
      LevelKey = 1
      Width = 25
```

```
    PartialCheck = True
    Alignment = 1
  }
  ""
  {
    LevelKey = 1
    Width = 20
  }
  ""
  {
    Position = 0
    Def(2) = True
    Width = 16
  }

}
Chart
{
  FirstVisibleDate = "5/29/2005"
}
Items
{
  Dim h, h1,hx
  h = AddItem(" exG2antt Add an advanced g2antt chart to your application.")
  CellTooltip(h,0) = "You can have a HTML multiple lines tooltip for any cell in the tree."
  CellValueFormat(h,0) = 1
  CellSingleLine(h,0) = False
  CellImages(h,1) = "1,2,3,"
  CellHAlignment(h,1) = 2
  CellMerge(h,0) = 1
  CellMerge(h,0) = 2
  CellMerge(h,0) = 3
  AddBar(h,"Progress","5/30/2005","6/4/2005",1,"
- TODO: project -")
  AddBar(h,"Deadline","5/29/2005 16:00","6/2/2005",2)
  AddBar(h,"Deadline","6/4/2005 07:00","6/10/2005",3)
```

```
    h1 = InsertItem(h,,"Project Sumarry1")
    CellHasCheckBox(h1,0) = True
    CellImage(h1,0) = 1
    AddBar(h1,"Project Summary","5/31/2005","6/15/2005"")
    AddBar(h1,"Milestone","5/30/2005","5/31/2005","M")
    AddBar(h1,"Milestone","6/16/2005","6/17/2005","E")

    h1 = InsertItem(h,,"Task...Split")
    CellHasCheckBox(h1,0) = True
    CellState(h1,0) = 1
    CellImage(h1,0) = 2
    AddBar(h1,"Task","6/1/2005","6/4/2005","S")
    AddBar(h1,"Split","6/4/2005","6/6/2005","Split")
    AddBar(h1,"Task","6/6/2005","6/12/2005","E")

    ExpandItem(h) = True

    h = AddItem("")
    CellValue(h,1) = "Custom icons ..."
    CellValueFormat(h,1) = 1
    ItemDivider(h) = 1
    ItemHeight(h) = 28
    ItemDividerLine(h) = 3
    CellHAlignment(h,1) = 1
    SelectableItem(h) = False
    CellPicture(h,1) =
"gBHJJGHA5MIqAAXAD3AENhozhpmhqZhrMhr/h0QGcQM0QTMQZkQf8QAESGcSM0STM

    AddBar(h,"Custom","5/31/2005","6/4/2005")
    ItemBackColor(h) = 100663296

    h = AddItem("Root 2")
    CellImages(h,0) = "2,3"
    ItemBold(h) = True
    CellMerge(h,0) = 1
    CellMerge(h,0) = 2
    CellMerge(h,0) = 3
```

```
    h1 = InsertItem(h,,"Task 1")
    AddBar(h1,"Task","6/4/2005","6/5/2005","S")
    AddBar(h1,"Split","6/5/2005","6/8/2005","Split")
    AddBar(h1,"Task","6/8/2005","6/10/2005","E")
    AddBar(h1,"","5/30/2005 12:00","6/3/2005 11:00",,"some text")
    ItemBar(h1,,7) = 83886080
'   ItemBar(h1,,8) = RGB(255,255,255)
    ItemBar(h1,,6) = "This is a bit of text that should occur when the cursor hovers the bar or the text."

    h1 = InsertItem(h,,"Task 2")

    ExpandItem(h) = true

}

EndUpdate
```

On **Windows XP**, the following table shows how the common controls are broken into parts and states:

Control/ClassName	Part	States
BUTTON	BP_CHECKBOX = 3	CBS_UNCHECKED 1 CBS_UNCHECKE CBS_UNCHECKED = 3 CBS_UNCHECKED = 4 CBS_CHECKE 5 CBS_CHECKEDH CBS_CHECKEDPR CBS_CHECKEDDIS CBS_MIXEDNORM CBS_MIXEDHOT = CBS_MIXEDPRESS CBS_MIXEDDISAB
	BP_GROUPBOX = 4	GBS_NORMAL = 1 GBS_DISABLED =
		PBS_NORMAL = 1 = 2 PBS_PRESSED

	BP_PUSHBUTTON = 1	PBS_DISABLED = PBS_DEFAULTED =
		RBS_UNCHECKED 1 RBS_UNCHECKE RBS_UNCHECKED = 3
	BP_RADIOBUTTON = 2	RBS_UNCHECKED = 4 RBS_CHECKED 5 RBS_CHECKEDH RBS_CHECKEDPR RBS_CHECKEDDIS
	BP_USERBUTTON = 5	
CLOCK	CLP_TIME = 1	CLS_NORMAL = 1
COMBOBOX	CP_DROPDOWNBUTTON = 1	CBXS_NORMAL = CBXS_HOT = 2 CBXS_PRESSED = CBXS_DISABLED =
EDIT	EP_CARET = 2	
	EP_EDITTEXT = 1	ETS_NORMAL = 1 2 ETS_SELECTED ETS_DISABLED = ETS_FOCUSED = 5 ETS_READONLY = ETS_ASSIST = 7
EXPLORERBAR	EBP_HEADERBACKGROUND = 1	
	EBP_HEADERCLOSE = 2	EBHC_NORMAL = EBHC_HOT = 2 EBHC_PRESSED =
	EBP_HEADERPIN = 3	EBHP_NORMAL = EBHP_HOT = 2 EBHP_PRESSED = EBHP_SELECTEDN 4 EBHP_SELECTE EBHP_SELECTEDF 6
	EBP_IEBARMENU = 4	EBM_NORMAL = 1 = 2 EBM_PRESSE
	EBP_NORMALGROUPBACKGROUND = 5	
	EBP_NORMALGROUPCOLLAPSE = 6	EBNGC_NORMAL = EBNGC_HOT = 2 EBNGC_PRESSED

		EBNGE_NORMAL =
	EBP_NORMALGROUPEXPAND = 7	EBNGE_HOT = 2
		EBNGE_PRESSED
	EBP_NORMALGROUPHEAD = 8	
	EBP_SPECIALGROUPBACKGROUND = 9	
		EBSGC_NORMAL =
	EBP_SPECIALGROUPCOLLAPSE = 10	EBSGC_HOT = 2
		EBSGC_PRESSED
		EBSGE_NORMAL =
	EBP_SPECIALGROUPEXPAND = 11	EBSGE_HOT = 2
		EBSGE_PRESSED
	EBP_SPECIALGROUPHEAD = 12	
HEADER	HP_HEADERITEM = 1	HIS_NORMAL = 1 H
		2 HIS_PRESSED =
	HP_HEADERITEMLEFT = 2	HILS_NORMAL = 1
		= 2 HILS_PRESSEI
	HP_HEADERITEMRIGHT = 3	HIRS_NORMAL = 1
		= 2 HIRS_PRESSE
	HP_HEADERSORTARROW = 4	HSAS_SORTEDUP
		HSAS_SORTEDDO
LISTVIEW	LVP_EMPTYTEXT = 5	
	LVP_LISTDETAIL = 3	
	LVP_LISTGROUP = 2	
		LIS_NORMAL = 1 L
		2 LIS_SELECTED =
	LVP_LISTITEM = 1	LIS_DISABLED = 4
		LIS_SELECTEDNO
		5
	LVP_LISTSORTEDDETAIL = 4	
		MS_NORMAL = 1
MENU	MP_MENUBARDROPDOWN = 4	MS_SELECTED = 2
		MS_DEMOTED = 3
		MS_NORMAL = 1
	MP_MENUBARITEM = 3	MS_SELECTED = 2
		MS_DEMOTED = 3
		MS_NORMAL = 1
	MP_CHEVRON = 5	MS_SELECTED = 2
		MS_DEMOTED = 3
		MS_NORMAL = 1
		MS_SELECTED = 2

	MP_MENUDROPDOWN = 2	MS_DEMOTED = 3
	MP_MENUITEM = 1	MS_NORMAL = 1 MS_SELECTED = 2 MS_DEMOTED = 3
	MP_SEPARATOR = 6	MS_NORMAL = 1 MS_SELECTED = 2 MS_DEMOTED = 3
MENUBAND	MDP_NEWAPPBUTTON = 1	MDS_NORMAL = 1 = 2 MDS_PRESSE MDS_DISABLED = MDS_CHECKED = MDS_HOTCHECKE
	MDP_SEPERATOR = 2	
PAGE	PGRP_DOWN = 2	DNS_NORMAL = 1 = 2 DNS_PRESSED DNS_DISABLED =
	PGRP_DOWNHORZ = 4	DNHZS_NORMAL = DNHZS_HOT = 2 DNHZS_PRESSED DNHZS_DISABLED
	PGRP_UP = 1	UPS_NORMAL = 1 = 2 UPS_PRESSED UPS_DISABLED =
	PGRP_UPHORZ = 3	UPHZS_NORMAL = UPHZS_HOT = 2 UPHZS_PRESSED UPHZS_DISABLED
PROGRESS	PP_BAR = 1	
	PP_BARVERT = 2	
	PP_CHUNK = 3	
	PP_CHUNKVERT = 4	
REBAR	RP_BAND = 3	
	RP_CHEVRON = 4	CHEVS_NORMAL = CHEVS_HOT = 2 CHEVS_PRESSED
	RP_CHEVRONVERT = 5	
	RP_GRIPPER = 1	
	RP_GRIPPERVERT = 2	
		ABS_DOWNDISAB

SCROLLBAR	SBP_ARROWBTN = 1	ABS_DOWNHOT, ABS_DOWNNORM ABS_DOWNPRESS ABS_UPDISABLED ABS_UPHOT, ABS_UPNORMAL, ABS_UPPRESSED, ABS_LEFTDISABLE ABS_LEFTHOT, ABS_LEFTNORMA ABS_LEFTPRESSE ABS_RIGHTDISAB ABS_RIGHTHOT, ABS_RIGHTNORM ABS_RIGHTPRESS
	SBP_GRIPPERHORZ = 8 SBP_GRIPPERVERT = 9	
	SBP_LOWERTRACKHORZ = 4	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_LOWERTRACKVERT = 6	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_THUMBBTNHORZ = 2	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_THUMBBTNVERT = 3	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_UPPERTRACKHORZ = 5	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_UPPERTRACKVERT = 7	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
		SZB_RIGHTALIGN

	SBP_SIZEBOX = 10	SZB_LEFTALIGN =
SPIN	SPNP_DOWN = 2	DNS_NORMAL = 1 = 2 DNS_PRESSED DNS_DISABLED =
	SPNP_DOWNHORZ = 4	DNHZS_NORMAL = DNHZS_HOT = 2 DNHZS_PRESSED DNHZS_DISABLED
	SPNP_UP = 1	UPS_NORMAL = 1 = 2 UPS_PRESSED UPS_DISABLED =
	SPNP_UPHORZ = 3	UPHZS_NORMAL = UPHZS_HOT = 2 UPHZS_PRESSED UPHZS_DISABLED
STARTPANEL	SPP_LOGOFF = 8	
	SPP_LOGOFFBUTTONS = 9	SPLS_NORMAL = SPLS_HOT = 2 SPLS_PRESSED =
	SPP_MOREPROGRAMS = 2	
	SPP_MOREPROGRAMSARROW = 3	SPS_NORMAL = 1 = 2 SPS_PRESSED
	SPP_PLACESLIST = 6	
	SPP_PLACESLISTSEPARATOR = 7	
	SPP_PREVIEW = 11	
	SPP_PROGLIST = 4	
	SPP_PROGLISTSEPARATOR = 5	
	SPP_USERPANE = 1	
	SPP_USERPICTURE = 10	
STATUS	SP_GRIPPER = 3	
	SP_PANE = 1	
	SP_GRIPPERPANE = 2	
TAB	TABP_BODY = 10	
	TABP_PANE = 9	
	TABP_TABITEM = 1	TIS_NORMAL = 1 2 TIS_SELECTED = TIS_DISABLED = 4 TIS_FOCUSED = 5

	TABP_TABITEMBOTHEDGE = 4	TIBES_NORMAL = TIBES_HOT = 2 TIBES_SELECTED TIBES_DISABLED TIBES_FOCUSED =
	TABP_TABITEMLEFTEDGE = 2	TILES_NORMAL = TILES_HOT = 2 TILES_SELECTED TILES_DISABLED = TILES_FOCUSED =
	TABP_TABITEMRIGHTEDGE = 3	TIRES_NORMAL = TIRES_HOT = 2 TIRES_SELECTED TIRES_DISABLED TIRES_FOCUSED
	TABP_TOPTABITEM = 5	TTIS_NORMAL = 1 = 2 TTIS_SELECTE TTIS_DISABLED = TTIS_FOCUSED =
	TABP_TOPTABITEMBOTHEDGE = 8	TTIBES_NORMAL = TTIBES_HOT = 2 TTIBES_SELECTE TTIBES_DISABLED TTIBES_FOCUSED
	TABP_TOPTABITEMLEFTEDGE = 6	TTILES_NORMAL = TTILES_HOT = 2 TTILES_SELECTE TTILES_DISABLED TTILES_FOCUSED
	TABP_TOPTABITEMRIGHTEDGE = 7	TTIRES_NORMAL TTIRES_HOT = 2 TTIRES_SELECTE TTIRES_DISABLED TTIRES_FOCUSED
TASKBAND	TDP_GROUPCOUNT = 1	
	TDP_FLASHBUTTON = 2	
	TDP_FLASHBUTTONGROUPMENU = 3	
TASKBAR	TBP_BACKGROUNDBOTTOM = 1	
	TBP_BACKGROUNDLEFT = 4	
	TBP_BACKGROUNDRIGHT = 2	

	TBP_BACKGROUNDTOP = 3	
	TBP_SIZINGBARBOTTOM = 5	
	TBP_SIZINGBARBOTTOMLEFT = 8	
	TBP_SIZINGBARRIGHT = 6	
	TBP_SIZINGBARTOP = 7	
TOOLBAR	TP_BUTTON = 1	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED
	TP_DROPDOWNBUTTON = 2	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED
	TP_SPLITBUTTON = 3	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED
	TP_SPLITBUTTONDROPDOWN = 4	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED
	TP_SEPARATOR = 5	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED
	TP_SEPARATORVERT = 6	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED
TOOLTIP	TTP_BALLOON = 3	TTBS_NORMAL = TTBS_LINK = 2
	TTP_BALLOONTITLE = 4	TTBS_NORMAL = TTBS_LINK = 2 TTCS_NORMAL =

	TTP_CLOSE = 5	TTCS_HOT = 2 TTCS_PRESSED =
	TTP_STANDARD = 1	TTSS_NORMAL = TTSS_LINK = 2
	TTP_STANDARDTITLE = 2	TTSS_NORMAL = TTSS_LINK = 2
TRACKBAR	TKP_THUMB = 3	TUS_NORMAL = 1 2 TUS_PRESSED = TUS_FOCUSED = 4 TUS_DISABLED =
	TKP_THUMBBOTTOM = 4	TUBS_NORMAL = TUBS_HOT = 2 TUBS_PRESSED = TUBS_FOCUSED = TUBS_DISABLED =
	TKP_THUMBLEFT = 7	TUVLS_NORMAL = TUVLS_HOT = 2 TUVLS_PRESSED TUVLS_FOCUSED TUVLS_DISABLED
	TKP_THUMBRIGHT = 8	TUVRS_NORMAL = TUVRS_HOT = 2 TUVRS_PRESSED TUVRS_FOCUSED TUVRS_DISABLED
	TKP_THUMBTOP = 5	TUTS_NORMAL = TUTS_HOT = 2 TUTS_PRESSED = TUTS_FOCUSED = TUTS_DISABLED =
	TKP_THUMBVERT = 6	TUVS_NORMAL = TUVS_HOT = 2 TUVS_PRESSED = TUVS_FOCUSED = TUVS_DISABLED =
	TKP_TICS = 9	TSS_NORMAL = 1
	TKP_TICSVERT = 10	TSVS_NORMAL =
	TKP_TRACK = 1	TRS_NORMAL = 1
	TKP_TRACKVERT = 2	TRVS_NORMAL =
TRAYNOTIFY	TNP_ANIMBACKGROUND = 2	
	TNP_BACKGROUND = 1	

TREEVIEW	TVP_BRANCH = 3	
	TVP_GLYPH = 2	GLPS_CLOSED = 1 GLPS_OPENED = 2
	TVP_TREEITEM = 1	TREIS_NORMAL = 1 TREIS_HOT = 2 TREIS_SELECTED TREIS_DISABLED TREIS_SELECTED = 5
WINDOW	WP_CAPTION = 1	CS_ACTIVE = 1 CS = 2 CS_DISABLED
	WP_CAPTIONSIZINGTEMPLATE = 30	
	WP_CLOSEBUTTON = 18	CBS_NORMAL = 1 = 2 CBS_PUSHED CBS_DISABLED =
	WP_DIALOG = 29	
	WP_FRAMEBOTTOM = 9	FS_ACTIVE = 1 FS = 2
	WP_FRAMEBOTTOMSIZINGTEMPLATE = 36	
	WP_FRAMELEFT = 7	FS_ACTIVE = 1 FS = 2
	WP_FRAMELEFTSIZINGTEMPLATE = 32	
	WP_FRAMERIGHT = 8	FS_ACTIVE = 1 FS = 2
	WP_FRAMERIGHTSIZINGTEMPLATE = 34	
	WP_HELPBUTTON = 23	HBS_NORMAL = 1 = 2 HBS_PUSHED HBS_DISABLED =
	WP_HORZSCROLL = 25	HSS_NORMAL = 1 = 2 HSS_PUSHED HSS_DISABLED =
	WP_HORZTHUMB = 26	HTS_NORMAL = 1 2 HTS_PUSHED = 3 HTS_DISABLED = 4
	WP_MAX_BUTTON	MAXBS_NORMAL = MAXBS_HOT = 2 MAXBS_PUSHED = MAXBS_DISABLED MXCS_ACTIVE = 1

WP_MAXCAPTION = 5	MXCS_INACTIVE = MXCS_DISABLED
WP_MDICLOSEBUTTON = 20	CBS_NORMAL = 1 = 2 CBS_PUSHED CBS_DISABLED =
WP_MDIHELPBUTTON = 24	HBS_NORMAL = 1 = 2 HBS_PUSHED HBS_DISABLED =
WP_MDIMINBUTTON = 16	MINBS_NORMAL = MINBS_HOT = 2 MINBS_PUSHED = MINBS_DISABLED
WP_MDIRESTOREBUTTON = 22	RBS_NORMAL = 1 = 2 RBS_PUSHED RBS_DISABLED =
WP_MDISYSBUTTON = 14	SBS_NORMAL = 1 = 2 SBS_PUSHED SBS_DISABLED =
WP_MINBUTTON = 15	MINBS_NORMAL = MINBS_HOT = 2 MINBS_PUSHED = MINBS_DISABLED
WP_MINCAPTION = 3	MNCS_ACTIVE = 1 MNCS_INACTIVE = MNCS_DISABLED
WP_RESTOREBUTTON = 21	RBS_NORMAL = 1 = 2 RBS_PUSHED RBS_DISABLED =
WP_SMALLCAPTION = 2	CS_ACTIVE = 1 CS = 2 CS_DISABLED
WP_SMALLCAPTIONSIZINGTEMPLATE = 31	
WP_SMALLCLOSEBUTTON = 19	CBS_NORMAL = 1 = 2 CBS_PUSHED CBS_DISABLED =
WP_SMALLFRAMEBOTTOM = 12	FS_ACTIVE = 1 FS = 2
WP_SMALLFRAMEBOTTOMSIZINGTEMPLATE = 37	
WP_SMALLFRAMELEFT = 10	FS_ACTIVE = 1 FS = 2

WP_SMALLFRAMELEFTSIZINGTEMPLATE = 33	
WP_SMALLFRAMERIGHT = 11	FS_ACTIVE = 1 FS = 2
WP_SMALLFRAMERIGHTSIZINGTEMPLATE = 35	
WP_SMALLHELPBUTTON	HBS_NORMAL = 1 = 2 HBS_PUSHED HBS_DISABLED =
WP_SMALLMAXBUTTON	MAXBS_NORMAL MAXBS_HOT = 2 MAXBS_PUSHED = MAXBS_DISABLED
WP_SMALLMAXCAPTION = 6	MXCS_ACTIVE = 1 MXCS_INACTIVE = MXCS_DISABLED
WP_SMALLMINCAPTION = 4	MNCS_ACTIVE = 1 MNCS_INACTIVE = MNCS_DISABLED
WP_SMALLRESTOREBUTTON	RBS_NORMAL = 1 = 2 RBS_PUSHED RBS_DISABLED =
WP_SMALLSYSBUTTON	SBS_NORMAL = 1 = 2 SBS_PUSHED SBS_DISABLED =
WP_SYSBUTTON = 13	SBS_NORMAL = 1 = 2 SBS_PUSHED SBS_DISABLED =
WP_VERTSCROLL = 27	VSS_NORMAL = 1 = 2 VSS_PUSHED VSS_DISABLED =
WP_VERTTHUMB = 28	VTS_NORMAL = 1 2 VTS_PUSHED = VTS_DISABLED =

method Appearance.Clear ()

Removes all skins in the control.

Type	Description

Use the Clear method to clear all skins from the control. Use the Remove method to remove a specific skin. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

The skin method may change the visual appearance for the following parts in the control:

- control's border, Appearance property
- **levels** on the chart area, BackColor property, BackColorLevelHeader property
- bar's background, ItemBar(exBarBackColor) property
- control's **header bar**, BackColorHeader property
- control's **filter bar**, FilterBarBackColor property
- control's **sort bar**, BackColorSort property
- the caption of the control's sort bar, BackColorSortCaption property
- **selected item** or cell, SelBackColor property
- **item**, ItemBackColor property
- **cell**, CellBackColor property
- cell's **button**, "**drop down**" filter bar button, "close" filter bar button, tooltips, and so on, Background property
- 'focus' box in the chart's overview area, OverviewSelBackColor property.

method Appearance.Remove (ID as Long)

Removes a specific skin from the control.

Type	Description
ID as Long	A Long expression that indicates the index of the skin being removed.

Use the Remove method to remove a specific skin. The identifier of the skin being removed should be the same as when the skin was added using the Add method. Use the Clear method to clear all skins from the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

The skin method may change the visual appearance for the following parts in the control:

- control's border, Appearance property
- **levels** on the chart area, BackColor property, BackColorLevelHeader property
- bar's background, ItemBar(exBarBackColor) property
- control's **header bar**, BackColorHeader property
- control's **filter bar**, FilterBarBackColor property
- control's **sort bar**, BackColorSort property
- the caption of the control's sort bar, BackColorSortCaption property
- **selected item** or cell, SelBackColor property
- **item**, ItemBackColor property
- **cell**, CellBackColor property
- cell's **button**, "**drop down**" filter bar button, "close" filter bar button, tooltips, and so on, Background property
- 'focus' box in the chart's overview area, OverviewSelBackColor property.

property Appearance.RenderType as Long

Specifies the way colored EBN objects are displayed on the component.

Type	Description
Long	A long expression that indicates how the EBN objects are shown in the control, like explained bellow.

By default, the RenderType property is 0, which indicates an A-color scheme. The RenderType property can be used to change the colors for the entire control, for parts of the controls that uses EBN objects. The RenderType property is not applied to the currently XP-theme if using.

The RenderType property is applied to all parts that displays an EBN object. The properties of color type may support the EBN object if the property's description includes "*A color expression that indicates the cell's background color. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.*" In other words, a property that supports EBN objects should be of format 0xIDRRGGBB, where the ID is the identifier of the EBN to be applied, while the BBGGRR is the (Red,Green,Blue, RGB-Color) color to be applied on the selected EBN. For instance, the 0x1000000 indicates displaying the EBN as it is, with no color applied, while the 0x1FF0000, applies the Blue color (RGB(0x0,0x0,0xFF), RGB(0,0,255) on the EBN with the identifier 1. You can use the EBNColor tool to visualize applying EBN colors.

Click here to watch a movie on how you can change the colors to be applied on EBN objects.

For instance, the following sample changes the control's header appearance, by using an EBN object:

```
With Control
   .VisualAppearance.Add 1,"c:\exontrol\images\normal.ebn"
   .BackColorHeader = &H1000000
End With
```

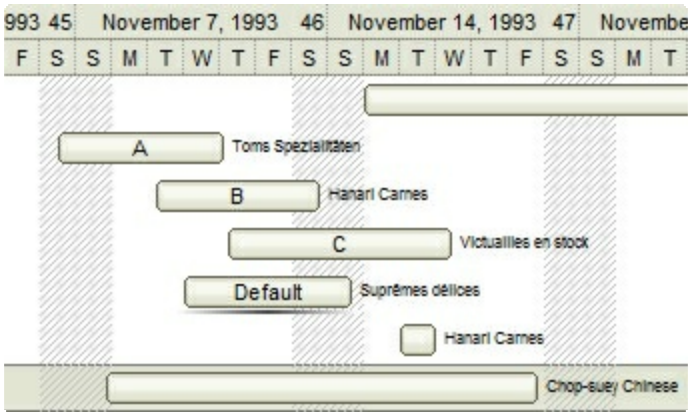
In the following screen shot the following objects displays the current EBN with a different color:

- "A" in Red (RGB(255,0,0), for instance the bar's property exBarColor is 0x10000FF
- "B" in Green (RGB(0,255,0), for instance the bar's property exBarColor is 0x100FF00

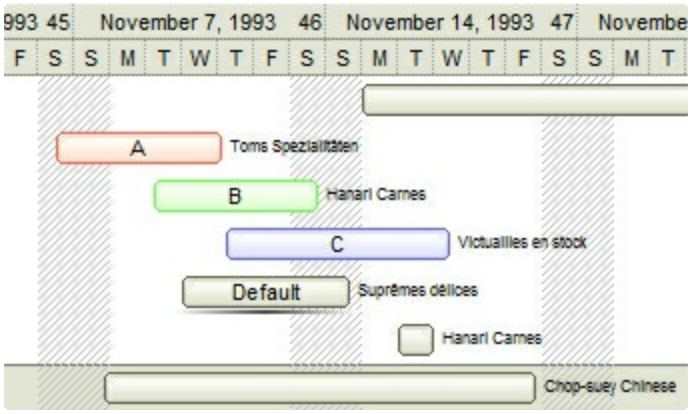
- "C" in Blue (RGB(0,0,255), for instance the bar's property exBarColor is 0x1FF0000
- "Default", no color is specified, for instance the bar's property exBarColor is 0x1000000

The RenderType property could be one of the following:

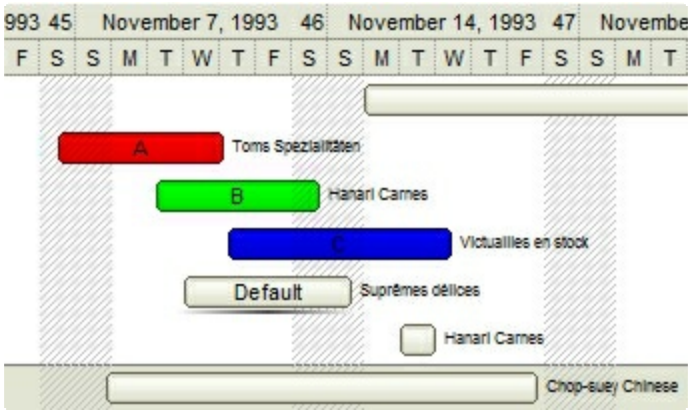
- ***-3**, no color is applied*. For instance, the BackColorHeader = &H1FF0000 is displayed as would be .BackColorHeader = &H1000000, so the 0xFF0000 color (Blue color) is ignored. You can use this option to allow the control displays the EBN colors or not.



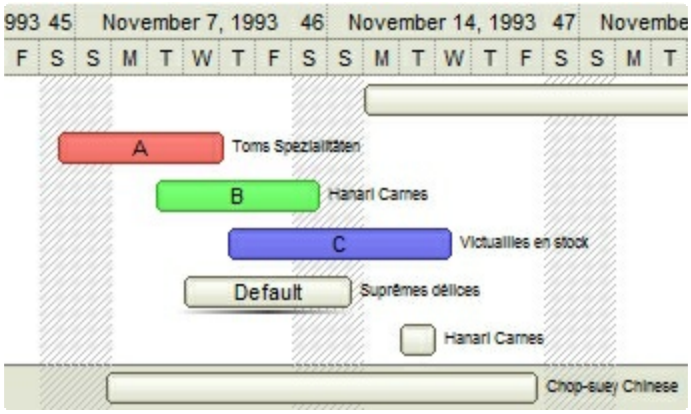
- ***-2**, OR-color scheme*. The color to be applied on the part of the control is a OR bit combination between the original EBN color and the specified color. For instance, the BackColorHeader = &H1FF0000, applies the OR bit for the entire Blue channel, or in other words, it applies a less Blue to the part of the control. This option should be used with solid colors (RGB(255,0,0), RGB(0,255,0), RGB(0,0,255), RGB(255,255,0), RGB(255,0,255), RGB(0,255,255),RGB(127,0,0),RGB(0,127,0), ...)



- ***-1**, AND-color scheme*, The color to be applied on the part of the control is an AND bit combination between the original EBN color and the specified color. For instance, the BackColorHeader = &H1FF0000, applies the AND bit for the entire Blue channel, or in other words, it applies a more Blue to the part of the control. This option should be used with solid colors (RGB(255,0,0), RGB(0,255,0), RGB(0,0,255), RGB(255,255,0), RGB(255,0,255), RGB(0,255,255),RGB(127,0,0),RGB(0,127,0), ...)

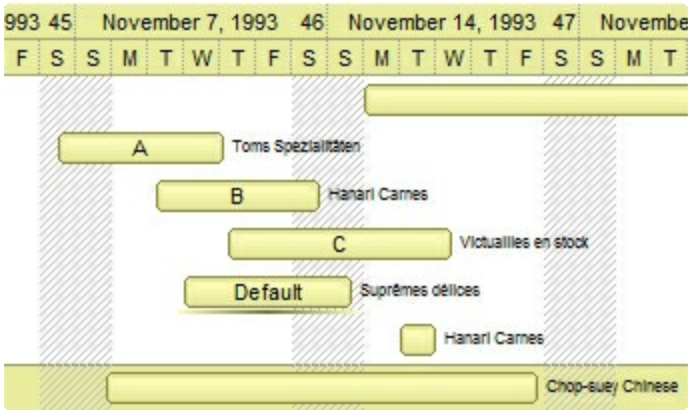


- ***0**, default*, the specified color is applied to the EBN. For instance, the BackColorHeader = &H1FF0000, applies a Blue color to the object. This option could be used to specify any color for the part of the components, that support EBN objects, not only solid colors.

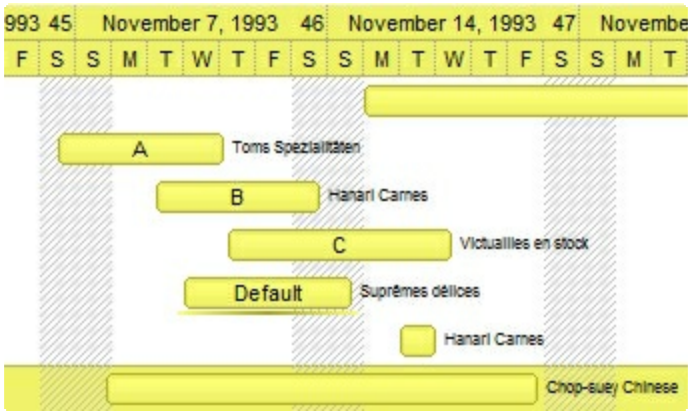


- **0xAABBGGRR**, where the AA a value between 0 to 255, which indicates the transparency, and RR, GG, BB the red, green and blue values. This option applies the same color to all parts that displays EBN objects, whit ignoring any specified color in the color property. For instance, the RenderType on 0x4000FFFF, indicates a 25% Yellow on EBN objects. The 0x40, or 64 in decimal, is a 25 % from in a 256 interal, and the 0x00FFFF, indicates the Yellow (RGB(255,255,0)). The same could be if the RenderType is 0x40000000 + vbYellow, or &H40000000 + RGB(255, 255, 0), and so, the RenderType could be the 0xAA000000 + Color, where the Color is the RGB format of the color.

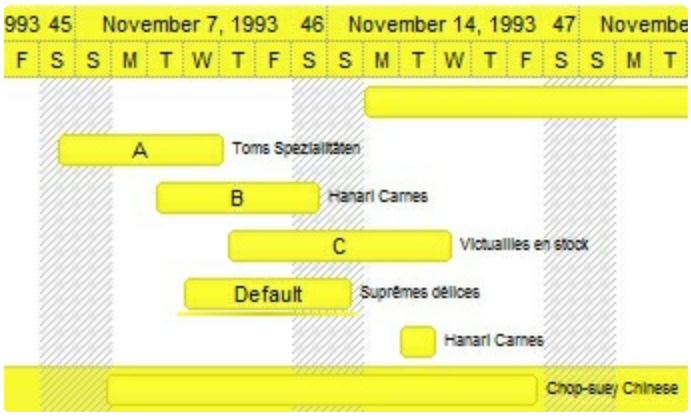
*The following picture shows the control with the RenderType property on 0x**40**00FFFF (25% Yellow, 0x40 or 64 in decimal is 25% from 256):*



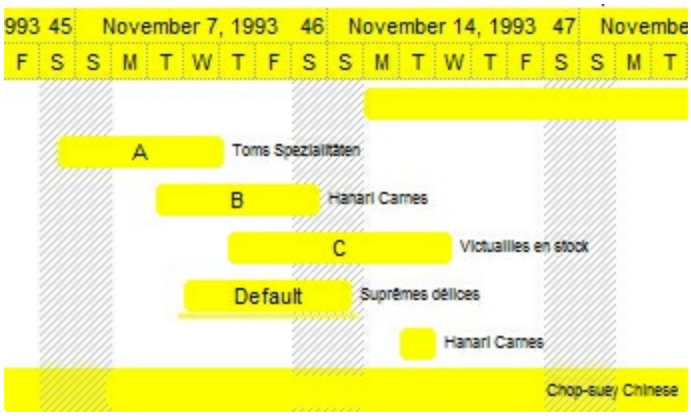
*The following picture shows the control with the RenderType property on 0x**80**00FFFF (50% Yellow, 0x80 or 128 in decimal is 50% from 256):*



*The following picture shows the control with the RenderType property on 0x**C0**00FFFF (75% Yellow, 0xC0 or 192 in decimal is 75% from 256):*



*The following picture shows the control with the RenderType property on 0x**FF**00FFFF (100% Yellow, 0xFF or 255 in decimal is 100% from 255):*


993 45
November 7, 1993
46
November 14, 1993
47
Novembe
F S S M T W T F S S M T W T F S S M T
A
Toms Spezialitäten
B
Hanari Carnes
C
Victuailles en stock
Default
Suprêmes délices
Hanari Carnes
Chop-suey Chinese

Bar object

The Bar object identifies a bar in the chart. A Bar object contains three parts: the start part and end part identifies the corners of the bar, and the middle part of the bar. The look and feel of the middle part of the bar are defined by the properties: Color, Pattern and Shape. The StartShape and StartColor properties defines the start part of the bar. The EndShape and EndColor properties defines the end part of the bar. Use the Bars property to access the Bars collection. Use the Chart object property to access the control's chart. Use the AddBar method to add a bar to an item. Use the Add and Copy methods to add new Bar objects. The Bar object supports the following properties and methods:

Name	Description
Color	Specifies the color of the bar.
Def	Specifies the default value of the given property for bars of the same type.
EndColor	Returns or sets a value that indicates the color for the right side corner.
EndShape	Retrieves or sets a value that indicates the shape of the right side corner.
FormatHistogramValues	Specifies the format to show bar's value in the histogram.
Height	Retrieves or sets a value that indicates the height in pixels of the bar.
HistogramBorderColor	Retrieves or sets a value that indicates the color to show the histogram's border.
HistogramBorderSize	Specifies the size of the border in pixels to show the bar's histogram.
HistogramColor	Retrieves or sets a value that indicates the color to be used in the histogram.
HistogramCriticalColor	Retrieves or sets a value that indicates the color to paint the overallocations in the histogram.
HistogramCriticalValue	Specifies the histogram's critical value.
HistogramCumulativeColor	Retrieves or sets a value that indicates a cumulative color to be shown in the histogram.
HistogramCumulativeColors	Specifies the number of colors that the histogram may show when it displays bars using cumulative type.
HistogramCumulativeOriginalColorBars	Specifies whether the original bar's color is changed accordingly to the cumulative histogram.
	Specifies the index of the column to display the legend for

HistogramCumulativeShowLegend	the cumulative bars in the histogram.
HistogramGridLinesColor	Retrieves or sets a value that indicates the color to show the histogram's grid lines.
HistogramItems	Specifies the number of items being represented in the histogram when overload is shown.
HistogramPattern	Retrieves or sets a value that indicates the pattern to be used in the histogram.
HistogramRulerLinesColor	Retrieves or sets a value that indicates the color to show the histogram's ruler lines.
HistogramType	Retrieves or sets a value that indicates the type of the histogram.
Name	Retrieves the name of the bar.
Overlaid	Retrieves or sets a value that indicates options for the specified overlaid type.
OverlaidGroup	Specifies the list of bars beside the current bar that may cover each other.
OverlaidType	Specifies how the overlaid bars are shown.
OverviewColor	Retrieves or sets a value that indicates the color to show the bars of this type in the control's overview panel.
Pattern	Retrieves or sets a value that indicates the pattern being used to fill the bar.
Shape	Retrieves or sets a value that indicates the shape of the bar.
Shortcut	Specifies a value that indicates a shortcut for the current bar.
ShowHistogramValues	Specifies the formula that returns the color to display the selected values in the histogram for specified type of bar.
StartColor	Returns or sets a value that indicates the color for the left side corner.
StartShape	Retrieves or sets a value that indicates the shape of the left side corner.

property Bar.Color as Color

Specifies the color of the bar.

Type	Description
Color	A Color expression that indicates the color of the bar. The last 7 bits in the high significant byte of the color indicates the identifier of the skin being used to paint the bar. Use the Add method to add new skins to the control. The skin object is used to draw the bar in the chart area.

Use the Color property to specify the color to fill the bar. This color is applied to all bars of the same type. The Color property specifies the color to show all bars of the same type. For instance, Chart.Bars("Task").Color = vbRed indicates that all "Task" bars will be shown in red.

The following properties may be used to change the color for a particular bar:

- The *ItemBar(exBarColor)* property specifies a different color/skin for a particular bar. If the ItemBar(exBarColor) property indicates using an EBN object (the last 7 bits in the high significant byte is not 0), it indicates using another EBN object to display the bar. If the ItemBar(exBarColor) property indicates no EBN object (the last 7 bits in the high significant byte is 0) but the Color property indicates using an EBN object, the ItemBar(exBarColor) color is applied over the bar's default EBN color.

- The *ItemBar(exBarBackColor)* property specifies the background color for the area being occupied by the bar. The exBarBackColor fills the item's background color for the area being delimited by the bar.

- The *ItemBar(exBarPercentColor)* specifies the color to show inside percent bar. The option is valid for bars that displays inside a percent bar. A bar can display a percent bar if it was creates using the Chart.Bars.Add("A%B") syntax. For instance, the Add("Task%Progress") adds a combination of Task  and Progress bars, so the Task shape is displayed on the full bar, and the Progress shape is displayed only on the portion determined by the Items.ItemBar(,,exBarPercent) value as 

. The ItemBar(exBarPercent) / ItemBar(exBarPercent100) property specifies the value of the percent to be displayed inside the bar.

- The *ItemBar(exBarNonWorkingColor)* specifies the color to show non-working parts of the bar. A bar may show different shape, pattern for non-working parts of the bar if it was previously created using the Bars.Add("A:B"). For instance, the Add("Task:Split") property adds a combination of Task  and Split  bars, so the Task bar is displayed in working area, and the Split bar is displayed in the non-working area

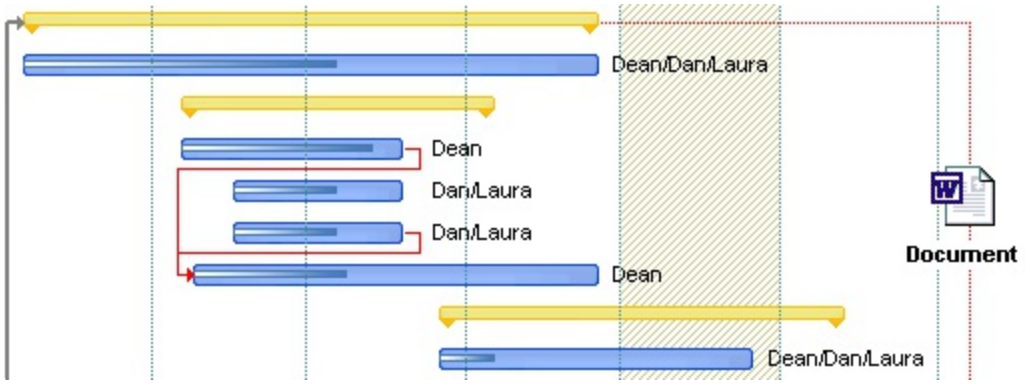
as . The NonworkingDays, NonworkingHours, ItemNonworkingUnits and related indicate the non-working parts of the chart.

- The *ItemBar(exSummaryBarBackColor)* specifies the item's background color for child bars owned by the summary bar. The DefineSummaryBars property defines bars that belongs to a summary bar. The UndefineSummaryBars method does the reverse operation, as it removes a bar from a summary bar.

- The *ItemBar(exBarOverviewColor)* property indicates the color for a particular bar in the overview part of the control. If not specified, the exBarColor property indicates the color of the bar in the overview part of the control. If the exBarColor property is not specified, the Color property indicates the color to display the bars in the overview part of the control. The OverviewVisible property specifies whether the control displays the overview/layout map of bars within the chart.

If the bar's Pattern is exPatternBox, the Color property indicates the color to show the bar's frame. In the same context, the StartColor and EndColor properties indicates the color to show the bar in gradient.

Use the Pattern property to specify the brush being used to fill the bar. Use the Shape property to specify the height and the vertical alignment of the middle part of the bar. Use the StartColor property to specify the color for the beginning part of the bar, if the StartShape property is not exShapeIconEmpty. Use the EndColor property to specify the color for the ending part of the bar, if the EndShape property is not exShapeIconEmpty.



In VB.NET or C# you require the following functions until the .NET framework will provide:

You can use the following VB.NET function:

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
   Dim i As Long
i = c.R
i = i + 256 * c.G
i = i + 256 * 256 * c.B
```

```
    ToUInt32 = Convert.ToUInt32(i)
    End Function
```

You can use the following C# function:

```
private UInt32 ToUInt32(Color c)
{
    long i;
    i = c.R;
    i = i + 256 * c.G;
    i = i + 256 * 256 * c.B;
    return Convert.ToUInt32(i);
}
```

The following VB sample creates a new bar called "Task2", that's similar with the "Task" bar excepts that we change the color to fill the bar:

```
With G2antt1.Chart.Bars
    With .Copy("Task", "Task2")
        .Color = RGB(255, 0, 0)
    End With
End With
```

The following C++ sample creates a new bar called "Task2", that's similar with the "Task" bar excepts that we change the color to fill the bar:

```
CBars bars = m_g2antt.GetChart().GetBars();
CBar bar = bars.Copy( "Task", "Task2" );
bar.SetColor( RGB(255,0,0) );
```

The following VB.NET sample creates a new bar called "Task2", that's similar with the "Task" bar excepts that we change the color to fill the bar:

```
With AxG2antt1.Chart.Bars
    With .Copy("Task", "Task2")
        .Color = ToUInt32(Color.Red)
    End With
End With
```

The following C# sample creates a new bar called "Task2", that's similar with the "Task" bar

excepts that we change the color to fill the bar:

```
EXG2ANTTLib.Bar bar = axG2antt1.Chart.Bars.Copy("Task", "Task2");
bar.Color = ToUInt32(Color.Red);
```

The following VFP sample creates a new bar called "Task2", that's similar with the "Task" bar excepts that we change the color to fill the bar:

```
with thisform.G2antt1.Chart.Bars
   with .Copy("Task", "Task2" )
      .Color = RGB(255,0,0)
   endwith
endwith
```

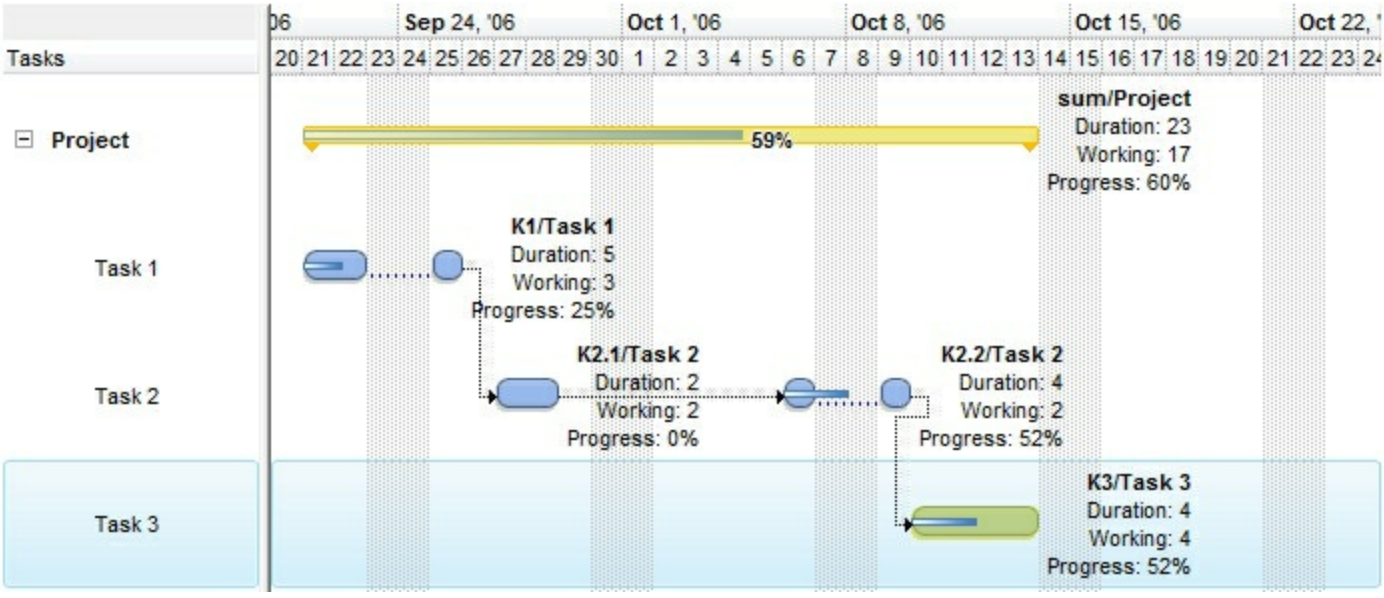
property Bar.Def(Property as ItemBarPropertyEnum) as Variant

Specifies the default value of the given property for bars of the same type.

Type	Description
Property as ItemBarPropertyEnum	An ItemBarPropertyEnum expression that indicates the property to access or set the default value.
Variant	A VARIANT expression that indicates the default value for giving property.

The Def property can be used to define your type of bars, or specify the values for ItemBar property when a new bar is added. The Def property has no effect for already added bars. For instance, you can set the Def(exBarCaption) so all added bars will display the caption giving a specified format. You can always use the ItemBar property to specify any property of the existing bar, while the Def property can define the default values for specified properties. Let's say you need to specify the ItemBar(exBarKeepWorkingCount) property for your bars, so they will keep their working units, while moving, so instead calling the ItemBar(exBarKeepWorkingCount) for each added bar, you can have the Def(exBarKeepWorkingCount) = True, and so any new bar to be added, will have this property set, so the ItemBar(exBarKeepWorkingCount) will gets True, for all new bars.

For instance the Bar.Def(exBarCaption) = "*<%=%9 + '/' + %C0%>
Duration: <%=(%2-%1)%>
Working: <%=%258%>
Progress: <%=round(100*%12)+'%'%>*" specifies that the bar's caption as in the following screen shot:



In the same manner, you can define the bar's tooltip such as: Bar.Def(exBarToolTip) = "*<%=%9 + '/' + %C0%>
Duration: <%=(%2-%1)%>
Working: <%=%258%>
Progress: <%=round(100*%12)+'%'%>*"

property Bar.EndColor as Color

Returns or sets a value that indicates the color for the right side corner.

Type	Description
Color	A Color expression that indicates the color for the ending part of the bar.

Use the EndColor property to specify the color to fill the end part of the bar, if the EndShape property is not exShapeIconEmpty or Pattern is exPatternBox. Use the Color property to specify the color to fill the middle part of the bar. Use the StartColor and StartShape properties to define the look and feel for the starting part of the bar. Use the AddShapeCorner property to add custom icons to the bars. In this case, the icon is processed before displaying based on the StartColor/ EndColor property. For instance, if you add an black and white icon, and the StartColor/EndColor is red, the icon will be painted in red. Instead, if the StartColor/EndColor property is -1 (0xFFFFFFFF, not white which is 0x00FFFFFF), the icon is painted as it was. If the Pattern property is exPatternBox, the StartColor and EndColor properties defines the start and ending color to show a gradient bar.

The following VB sample changes the "Task" bar visual appearance using liner gradient with margins as shown :

```
With G2antt1.Chart.Bars.Item("Task")
   .Color = vbWhite
   .Pattern = exPatternBox
   .StartShape = exShapeIconCircleDot
   .StartColor = vbRed
   .EndShape = exShapeIconCircleDot
   .EndColor = vbBlue
End With
```

The following VB sample changes the "Task" bar visual appearance using liner gradient with solid border as shown :

```
With G2antt1.Chart.Bars.Item("Task")
   .Color = vbRed
   .Pattern = exPatternBox
   .StartColor = vbRed
   .EndColor = vbBlue
End With
```

The following VB sample defines a new bar that looks like this  :

```
With G2antt1.Chart.Bars.Add("Task2")
   .Pattern = exPatternShadow
   .Color = RGB(0, 0, 255)
   .EndShape = exShapeIconCircleDot
   .EndColor = RGB(255, 0, 0)
End With
```

The following C++ sample defines a bar that looks like this above:

```
CBar bar = m_g2antt.GetChart().GetBars().Add("Task2");
bar.SetPattern( 3 /*exPatternShadow*/ );
bar.SetColor( RGB(0, 0, 255) );
bar.SetEndShape( 4 /* exShapeIconCircleDot*/ );
bar.SetEndColor( RGB(255, 0, 0) );
```

The following VB.NET sample defines a bar that looks like this above:

```
With AxG2antt1.Chart.Bars.Add("Task2")
   .Pattern = EXG2ANTTLib.PatternEnum.exPatternShadow
   .Color = RGB(0, 0, 255)
   .EndShape = EXG2ANTTLib.ShapeCornerEnum.exShapeIconCircleDot
   .EndColor = RGB(255, 0, 0)
End With
```

The following VB.NET sample adds a custom icon to the start of all Task bars:

```
With AxG2antt1.Chart.Bars
   .AddShapeCorner(12345, 1)
   .Item("Task").StartShape = 12345
   .Item("Task").StartColor = UInteger.MaxValue
End With
```

The following C# sample defines a bar that looks like this above:

```
EXG2ANTTLib.Bar bar = axG2antt1.Chart.Bars.Add("Task2");
bar.Pattern = EXG2ANTTLib.PatternEnum.exPatternShadow;
bar.Color = ToUInt32(Color.FromArgb(0, 0, 255));
```

```
bar.EndShape = EXG2ANTTLib.ShapeCornerEnum.exShapeIconCircleDot;
bar.EndColor = ToUInt32(Color.FromArgb(255, 0, 0));
```

The following C# sample adds a custom icon to the start of all Task bars:

```
EXG2ANTTLib.Bars bars = axG2antt1.Chart.Bars;
bars.AddShapeCorner(12345, 1);
bars["Task"].StartShape = EXG2ANTTLib.ShapeCornerEnum.exShapeIconEmpty + 12345;
bars["Task"].StartColor = 0xFFFFFFFF;
```

The following VFP sample defines a bar that looks like this above:

```
with thisform.G2antt1.Chart.Bars.Add("Task2")
   .Pattern = 3 && exPatternShadow
   .Color = RGB(0, 0, 255)
   .EndShape = 4 && exShapeIconCircleDot
   .EndColor = RGB(255, 0, 0)
EndWith
```

In VB.NET or C# you require the following functions until the .NET framework will provide:

You can use the following VB.NET function:

```
    Shared Function ToUInt32(ByVal c As Color) As UInt32
        Dim i As Long
    i = c.R
    i = i + 256 * c.G
    i = i + 256 * 256 * c.B
    ToUInt32 = Convert.ToUInt32(i)
    End Function
```

You can use the following C# function:

```
private UInt32 ToUInt32(Color c)
{
   long i;
   i = c.R;
   i = i + 256 * c.G;
   i = i + 256 * 256 * c.B;
   return Convert.ToUInt32(i);
```

```
}
```

property Bar.EndShape as ShapeCornerEnum

Retrieves or sets a value that indicates the shape of the right side corner.

Type	Description
ShapeCornerEnum	A ShapeCornerEnum expression that defines the shape of the icon being used to draw the corner.

By default, the EndShape property is exShapeIconEmpty. If the EndShape property is exShapeIconEmpty the bas has no ending part. Use the Color property to specify the color to fill the middle part of the bar. Use the Pattern property to specify the brush being used to fill the bar. Use the Shape property to specify the height and the vertical alignment of the middle part of the bar. Use the AddShapeCorner method to add a custom icon to be used as a starting or ending part of the bar. Use the Images or ReplaceIcon method to update the list of control's icons.

The following VB sample adds a custom shape and defines a bar like this  :

```
With G2antt1.Chart.Bars
   .AddShapeCorner 12345, 1
   With .Add("Task2")
      .Pattern = exPatternDot
      .Shape = exShapeThinDown
      .EndShape = 12345
      .EndColor = RGB(255, 0, 0)
      .Color = .EndColor
   End With
End With
```

The following C++ sample adds a custom shape and defines a bar like above:

```
CBars bars = m_g2antt.GetChart().GetBars();
bars.AddShapeCorner( COleVariant( (long)12345 ), COleVariant( (long)1 ) );
CBar bar = bars.Add("Task2");
bar.SetPattern( 2 /*exPatternDot*/ );
bar.SetShape( 20 /*exShapeThinDown*/ );
bar.SetEndShape( 12345 );
bar.SetEndColor( RGB(255, 0, 0) );
bar.SetColor( bar.GetEndColor() );
```

The following VB.NET sample adds a custom shape and defines a bar like above:

```
With AxG2antt1.Chart.Bars
   .AddShapeCorner(12345, 1)
   With .Add("Task2")
      .Pattern = EXG2ANTTLib.PatternEnum.exPatternDot
      .Shape = EXG2ANTTLib.ShapeBarEnum.exShapeThinDown
      .EndShape = 12345
      .EndColor = RGB(255, 0, 0)
      .Color = .EndColor
   End With
End With
```

The following C# sample adds a custom shape and defines a bar like above:

```
axG2antt1.Chart.Bars.AddShapeCorner(12345, 1);
EXG2ANTTLib.Bar bar = axG2antt1.Chart.Bars.Add("Task2");
bar.Pattern = EXG2ANTTLib.PatternEnum.exPatternDot;
bar.Shape = EXG2ANTTLib.ShapeBarEnum.exShapeThinDown;
bar.EndShape = (EXG2ANTTLib.ShapeCornerEnum)12345;
bar.EndColor = ToUInt32(Color.FromArgb(255, 0, 0));
bar.Color = bar.EndColor;
```

The following VFP sample adds a custom shape and defines a bar like above:

```
With thisform.G2antt1.Chart.Bars
   .AddShapeCorner(12345, 1)
   With .Add("Task2")
      .Pattern = 2 && exPatternDot
      .Shape = 20 && exShapeThinDown
      .EndShape = 12345
      .EndColor = RGB(255, 0, 0)
      .Color = .EndColor
   EndWith
EndWith
```

The following VB sample defines a new bar that looks like this  :

```
With .Chart.Bars.Add("Task2")
   .Pattern = exPatternShadow
```

```
    .Color = RGB(0, 0, 255)
    .EndShape = exShapeIconCircleDot
    .EndColor = RGB(255, 0, 0)
End With
```

The following C++ sample defines a bar that looks like this above:

```
CBar bar = m_g2antt.GetChart().GetBars().Add("Task2");
bar.SetPattern( 3 /*exPatternShadow*/ );
bar.SetColor( RGB(0, 0, 255) );
bar.SetEndShape( 4 /* exShapeIconCircleDot*/ );
bar.SetEndColor( RGB(255, 0, 0) );
```

The following VB.NET sample defines a bar that looks like this above:

```
With AxG2antt1.Chart.Bars.Add("Task2")
    .Pattern = EXG2ANTTLib.PatternEnum.exPatternShadow
    .Color = RGB(0, 0, 255)
    .EndShape = EXG2ANTTLib.ShapeCornerEnum.exShapeIconCircleDot
    .EndColor = RGB(255, 0, 0)
End With
```

The following C# sample defines a bar that looks like this above:

```
EXG2ANTTLib.Bar bar = axG2antt1.Chart.Bars.Add("Task2");
bar.Pattern = EXG2ANTTLib.PatternEnum.exPatternShadow;
bar.Color = ToUInt32(Color.FromArgb(0, 0, 255));
bar.EndShape = EXG2ANTTLib.ShapeCornerEnum.exShapeIconCircleDot;
bar.EndColor = ToUInt32(Color.FromArgb(255, 0, 0));
```

The following VFP sample defines a bar that looks like this above:

```
with thisform.G2antt1.Chart.Bars.Add("Task2")
    .Pattern = 3 && exPatternShadow
    .Color = RGB(0, 0, 255)
    .EndShape = 4 && exShapeIconCircleDot
    .EndColor = RGB(255, 0, 0)
EndWith
```

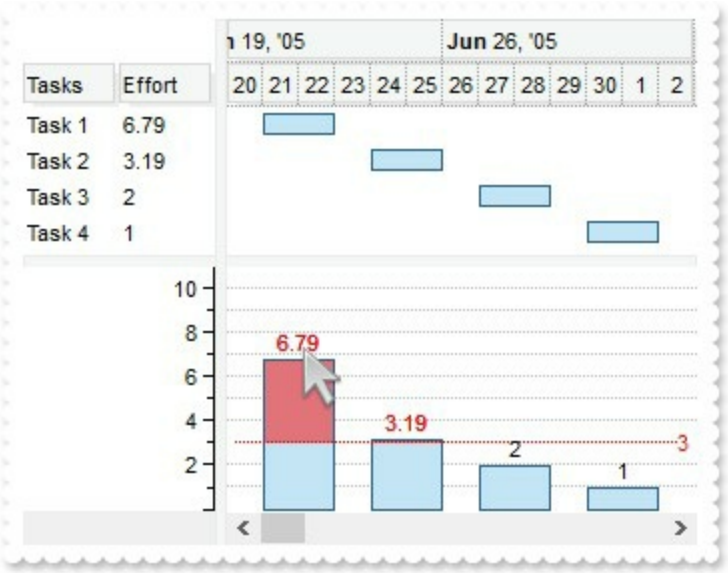
property Bar.FormatHistogramValues as String

Specifies the format to show bar's value in the histogram.

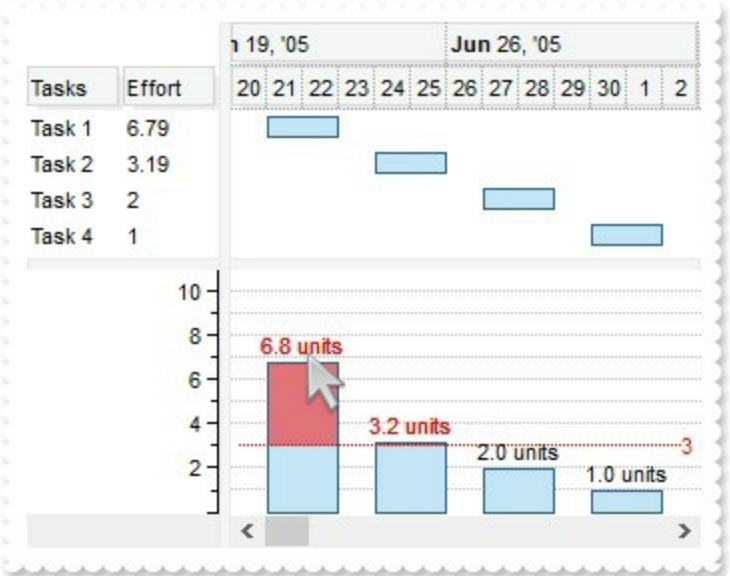
Type	Description
String	A String that specifies the expression to display the values in the bar's histogram.

By default, the FormatHistogramValues property is empty. The bar's histogram value gets formatted, if the FormatHistogramValues expression is not empty and valid. You can use the FormatHistogramValues property to customize the bar's values in the histogram. The bar is represented in the control's histogram, if its HistogramPattern / HistogramColor property is defined. Use the HistogramType property to specify the type of the graph to be displayed in the histogram for specified bar. The ResizeUnitScale property determines the refinement for the histogram part.

The following screen show shows how the bar's value/effort is displayed in the control's histogram by default:



The following screen show shows how the formatted bar's value/effort is displayed in the control's histogram (FormatHistogramValues property is "(value format `1`) + ` units`"):



The **value** keyword in the FormatHistogramValues property indicates the value to be formatted (ItemBar(exBarEffort)).

This property/method supports predefined constants and operators/functions as described here.

property Bar.Height as Long

Retrieves or sets a value that indicates the height in pixels of the bar.

Type	Description
Long	A Long expression that indicates the height of the bar, in pixels.

Use the Height property to change the heights for your bars. If the Height property is 0, the bar is not displayed. If the Height property is negative, the height of the bar is specified by the height of the item that displays the bar. If the Height property is positive it indicates the height of the bar to be displayed, in pixels. Use the DefaultItemHeight property to specify the default height for all items in the control. Use the ItemHeight property to specify the height for a specified item. The CellSingleLine property specifies whether a cell displays its caption using multiple lines. If you require a single bar with a different height, you can use the Copy method to copy a new bar, and use the Height property to specify a different height.

The control provides several predefined bars as follows:

- "Deadline":
- "Project Summary":
- "Summary":
- "Milestone":
- "Progress":
- "Split":
- "Task":

For instance, the following VB sample changes the height of the "Task" bar:

```
G2antt1.Chart.Bars("Task").Height = 18
```

The following VC++ sample changes the height of the "Task" bar:

```
m_g2antt.GetChart().GetBars().GetItem( COleVariant( "Task" ) ).SetHeight( 18 );
```

The following VFP sample changes the height of the "Task" bar:

```
With thisform.G2antt1.Chart.Bars
   .Item("Task").Height = 18
endwith
```

The following C# sample changes the height of the "Task" bar:

```
axG2antt1.Chart.Bars["Task"].Height = 18;
```

The following VB.NET sample changes the height of the "Task" bar:

```
AxG2antt1.Chart.Bars("Task").Height = 18
```

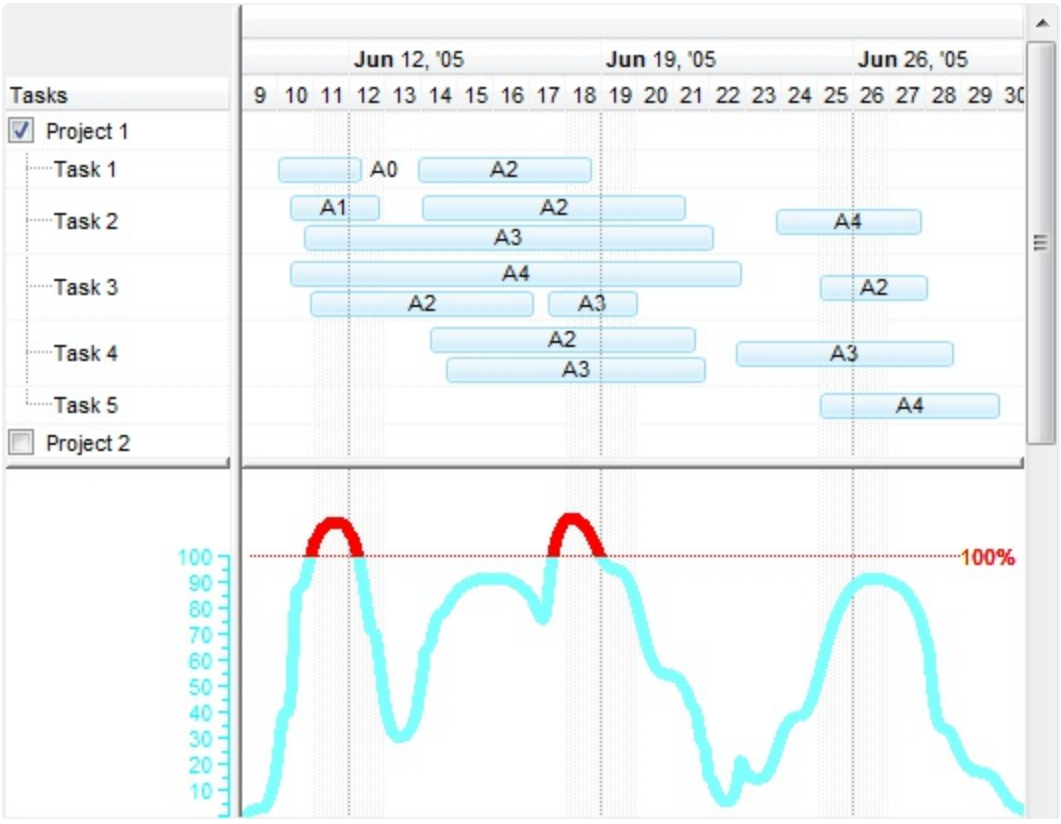
property Bar.HistogramBorderColor as Color

Retrieves or sets a value that indicates the color to show the histogram's border.

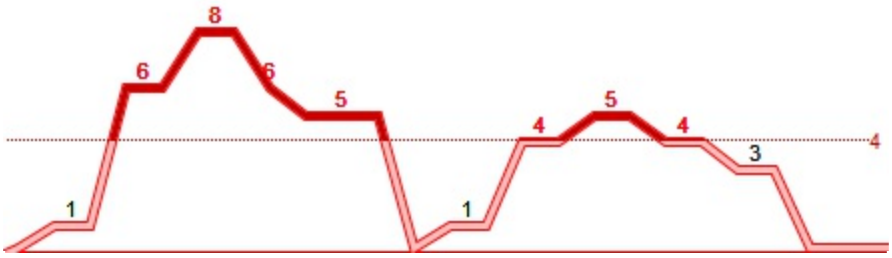
Type	Description
Color	A Color expression that specifies the color for the frame being shown in the histogram. If 0, the property is ignored.

By default the HistogramBorderColor property is 0 which means that it has no effect. In this case, if the HistogramPattern property points to a predefined value, the color for the frame in the histogram is automatically determined by the HistogramColor property. Use the HistogramBorderSize property to specify the width of the frame being shown in the histogram, only when curves are shown (the HistogramPattern property is not a predefined value, 256, 512, 1024, and so on).

The following screen shot shows the histogram of bars using different size and color for the frame:

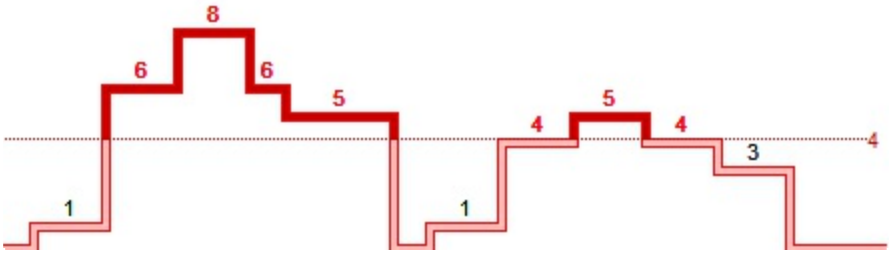


The following screen shot shows the bar's diagram if the HistogramPattern property is **exPolygonCurve** + **exPatternEmpty**, or simple **exPolygonCurve**, the AntiAliasing property is True, and the HistogramColor and HistogramBorderColor propertis have different values.



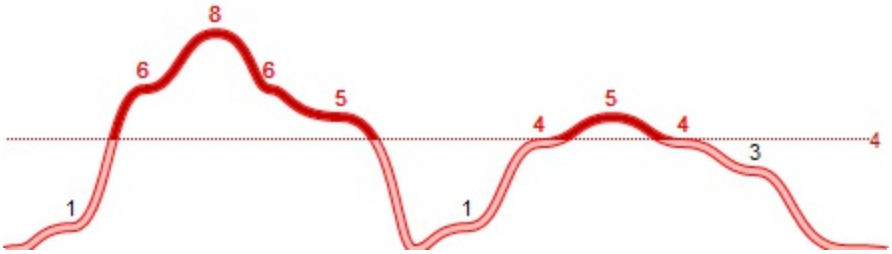
(HistogramPattern value is 256, AntiAliasing = True, HistogramColor != HistogramBorderColor)

The following screen shot shows the bar's diagram if the HistogramPattern property is **exRectangularCurve** + **exPatternEmpty**, or simple **exRectangularCurve**, the AntiAliasing property is True, and the HistogramColor and HistogramBorderColor propertis have different values.



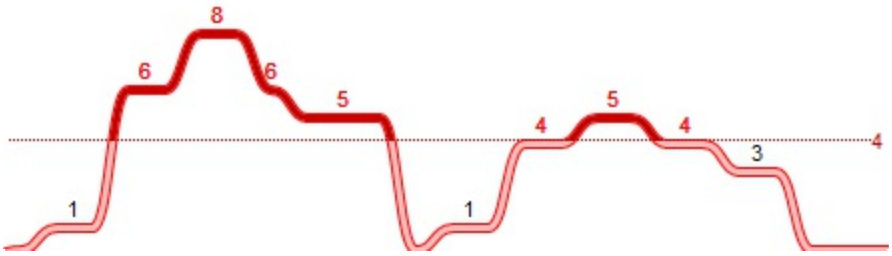
(HistogramPattern value is 2048, AntiAliasing = True, HistogramColor != HistogramBorderColor)

The following screen shot shows the bar's diagram if the HistogramPattern property is **exBezierCurve** + **exPatternEmpty**, or simple **exBezierCurve**, the AntiAliasing property is True, and the HistogramColor and HistogramBorderColor propertis have different values.



(HistogramPattern value is 512, AntiAliasing = True, HistogramColor != HistogramBorderColor)

The following screen shot shows the bar's diagram if the HistogramPattern property is **exRoundCurve** + **exPatternEmpty**, or simple **exRoundCurve**, the AntiAliasing property is True, and the HistogramColor and HistogramBorderColor propertis have different values.



(HistogramPattern value is 1024, AntiAliasing = True, HistogramColor != HistogramBorderColor)

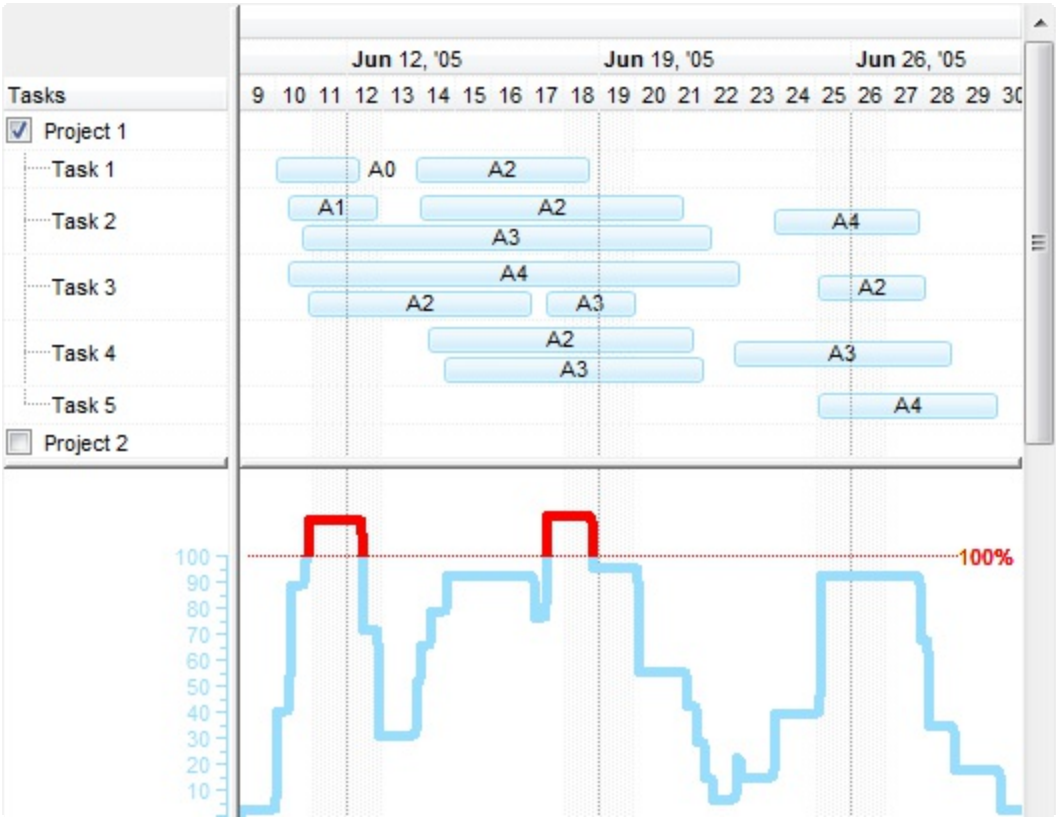
property Bar.HistogramBorderSize as Long

Specifies the size of the border in pixels to show the bar's histogram.

Type	Description
Long	A long expression that specifies the size of the frame being shown in the histogram.

By default, the HistogramBorderSize property is 3. Use the HistogramBorderSize property to specify the width of the frame being shown in the histogram, only when curves are shown (the HistogramPattern property is not a predefined value, 256, 512, 1024, and so son). Use the HistogramBorderColor property to change the color for the frame being shown in the histogram. For instance, if the HistogramPattern property points to a predefined value, the color for the frame in the histogram is automatically determined by the HistogramColor property.

The following screen shot shows the histogram of bars using different size and color for the frame:



property Bar.HistogramColor as Color

Retrieves or sets a value that indicates the color to be used in the histogram.

Type	Description
Color	A Color expression that specifies the color of the pattern being displayed for the bar in the histogram. The last 7 bits in the high significant byte of the color indicates the identifier of the skin being used. Use the Add method to add new skins to the control. The skin object is used to draw the histogram.

By default, the HistogramColor property is identical with the Color property. By default, no bar is represented in the histogram. A bar is represented in the histogram only if HistogramPattern or HistogramColor property is set. Use the HistogramColor property to define the color of the pattern or the skin object to be displayed in the histogram. Use the HistogramType property to specify the type of the graph to be displayed in the histogram for specified bar. Use the HistogramBackColor property to specify the histogram's background color. Use the HistogramVisible property to show or hide the histogram. Use the HistogramHeight property to specify at runtime the height of the histogram. The ResizeUnitScale property determines the refinement for the histogram part. For instance, if the chart displays days, while the bars are represented up to hours, the ResizeUnitScale property on exHour, will determine the histogram to show up to hours.

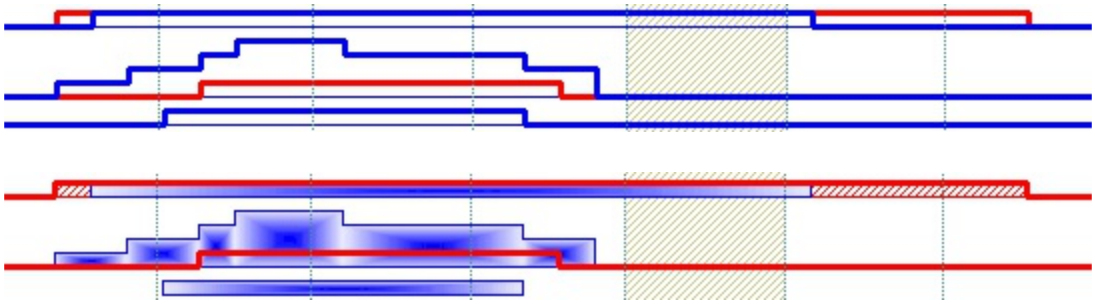
Please follow the steps in order to view your bars in the histogram.

1. Changes the **HistogramVisible** property on True (by default, it is False). After setting the HistogramVisible property on True, the control shows a horizontal splitter in the bottom side of the control.
2. Adjusts the height of the histogram view using the **HistogramHeight** property (by default it is 0). After setting the HistogramHeight property on a value greater than 0, the control shows a the histogram view in the bottom side of the control.
3. Changes the **HistogramPattern** **or/and HistogramColor** property, else no bars will be shown in the histogram. The HistogramPattern/HistogramColor properties belong to a Bar object. For instance the Chart.Bars("Task").HistogramPattern = exPatternDot, specifies that the Task bars will be represented in the histogram using the exPatternDot pattern ()

The followings are optional properties that you can set in order to customize your histogram:

- The **HistogramType** property indicates the type of the histogram being displayed for a specified bar.

- Use the **HistogramView** property to specify the items being represented in the histogram view. By default, only visible items are displayed in the histogram. For instance, using the HistogramView property you can select the items being represented in the histogram
- Use the HistogramBackColor property to specify the histogram's background color.



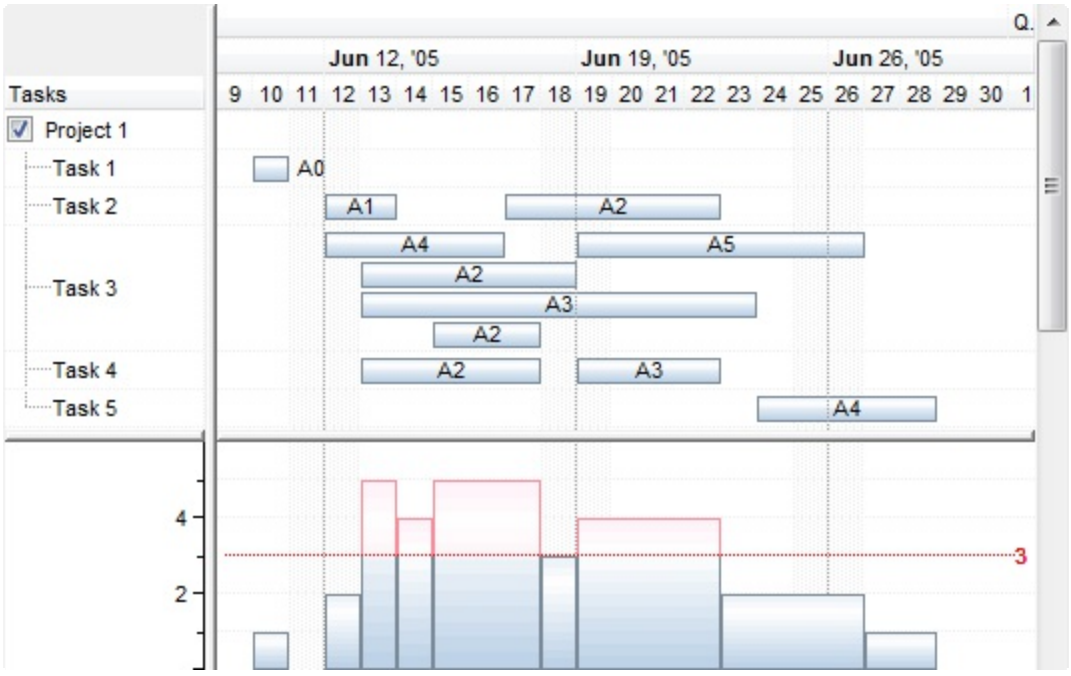
property Bar.HistogramCriticalColor as Color

Retrieves or sets a value that indicates the color to paint the overallocations in the histogram.

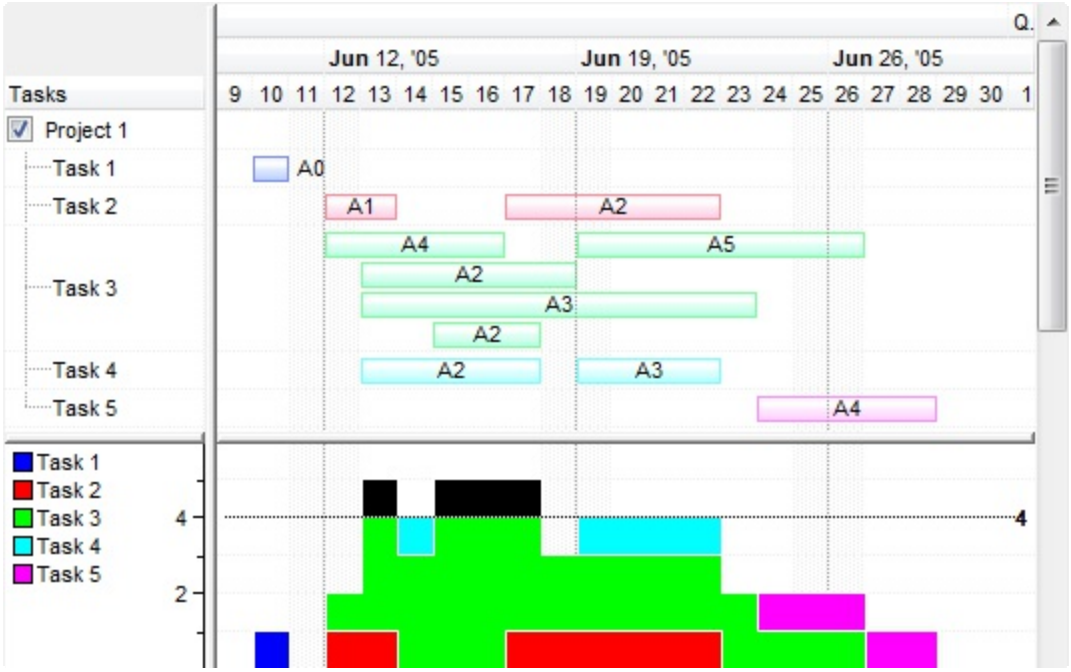
Type	Description
Color	A Color expression that specifies the color to display the histogram-graph when the allocations exceeds 100%

By default, the HistogramCriticalColor is red (RGB(255,0,0)). The histogram-chart shows the critical part for the bar ONLY if the HistogramCriticalColor is different that the HistogramColor property. Use the HistogramCriticalValue property to specify a critical value. The critical value is interpreted differently based on the HistogramType property. For instance, if the HistogramType property is exHistOverload, the critical value represents the count of cumulative bars since if the HistogramType property is exHistOverAllocation the critical value represents a percent value. Use HistogramRulerLinesColor property to specify the color to show the ruler in the left part of the histogram. Use the HistogramGridLinesColor specifies the color to show the grid lines when the HistogramType property is exHistOverload.

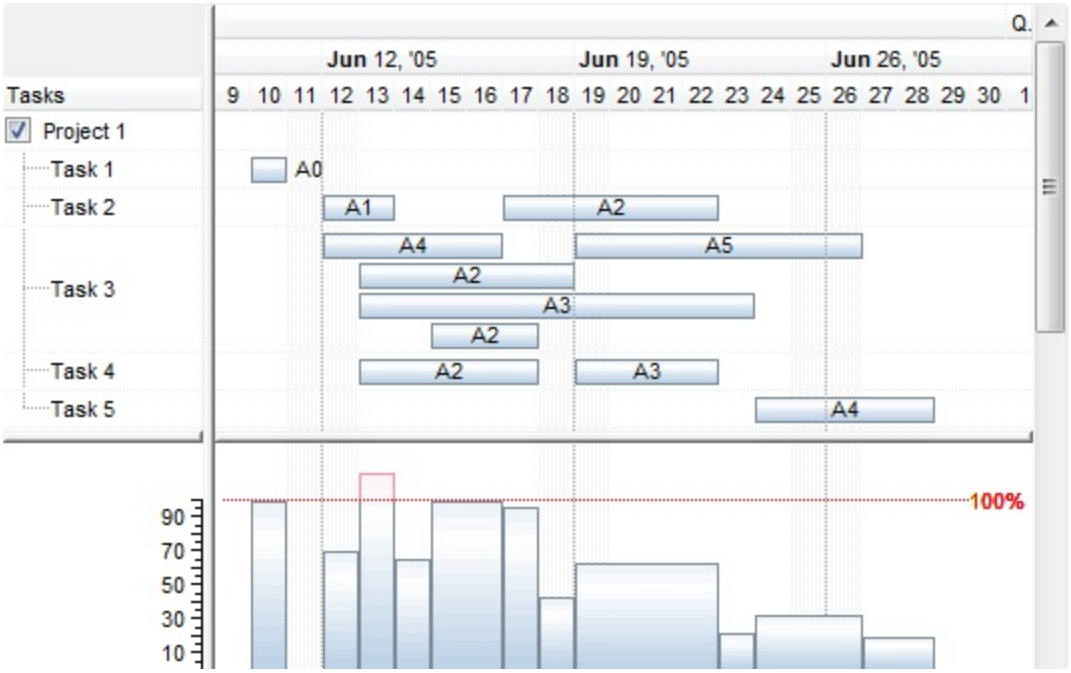
The following screen shot shows the critical part when HistogramType property is *exHistOverload*, and the HistogramCriticalValue property is 3. The bars over value 3 gets colored in red. The grid lines are shown and the ruler shows the count of the bars.



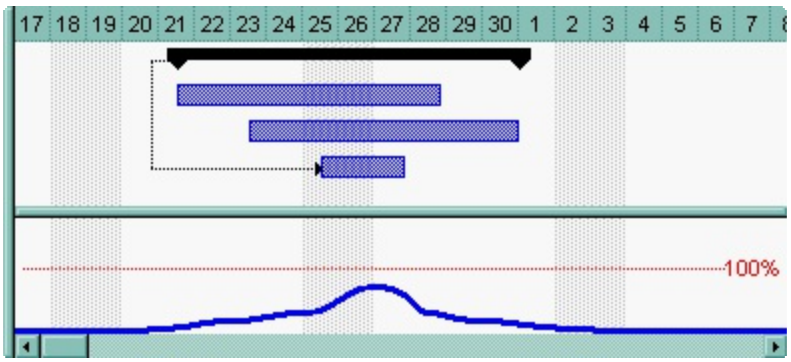
The following screen shot shows the critical part when HistogramType property is *exHistOverload* + *exHistCumulative*, and the HistogramCriticalValue property is 4. The bars over value 4 gets colored in black. The grid lines are shown and the ruler shows the count of the bars and the legend of the bars being colored.



The following screen shot shows the critical part when HistogramType property is *exHistOverallocation*, and the HistogramCriticalValue property is 100%. The bars over 100% gets colored in red. The grid lines are NOT shown and the ruler shows the percents.



The following screen shot shows how the histogram curve is changed once the user resizes or moves bars (in this case the HistogramType property is exHistOverAllocation)



property Bar.HistogramCriticalValue as Double

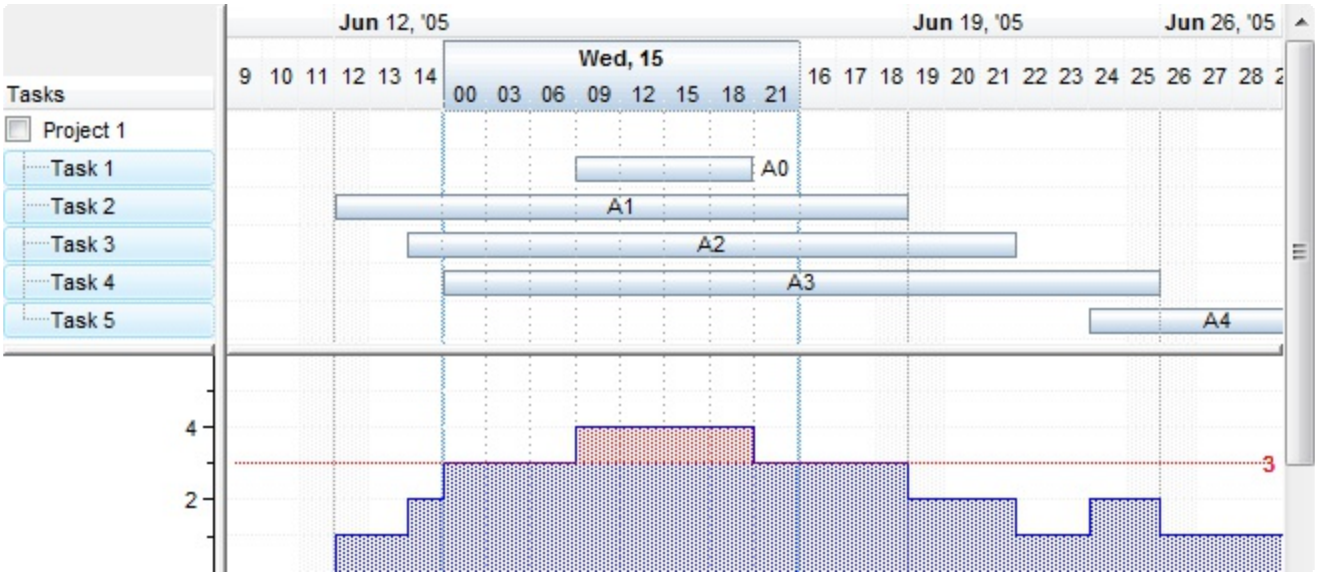
Specifies the histogram's critical value.

Type	Description
Double	A double expression that specifies the critical value. The critical value is interpreted in different way based on the HistogramType property. For instance, if the HistogramType property is *exHistOverload* the HistogramCriticalValue property specifies the count of bars while if the HistogramType property is *exHistOverallocation* it indicates a percent value.

By default, the HistogramCriticalValue property is 100. The HistogramCriticalColor property specifies the color to show the bars that are exceed the critical value. The histogram-chart shows the critical part for the bar ONLY if the HistogramCriticalColor is different that the HistogramColor property.

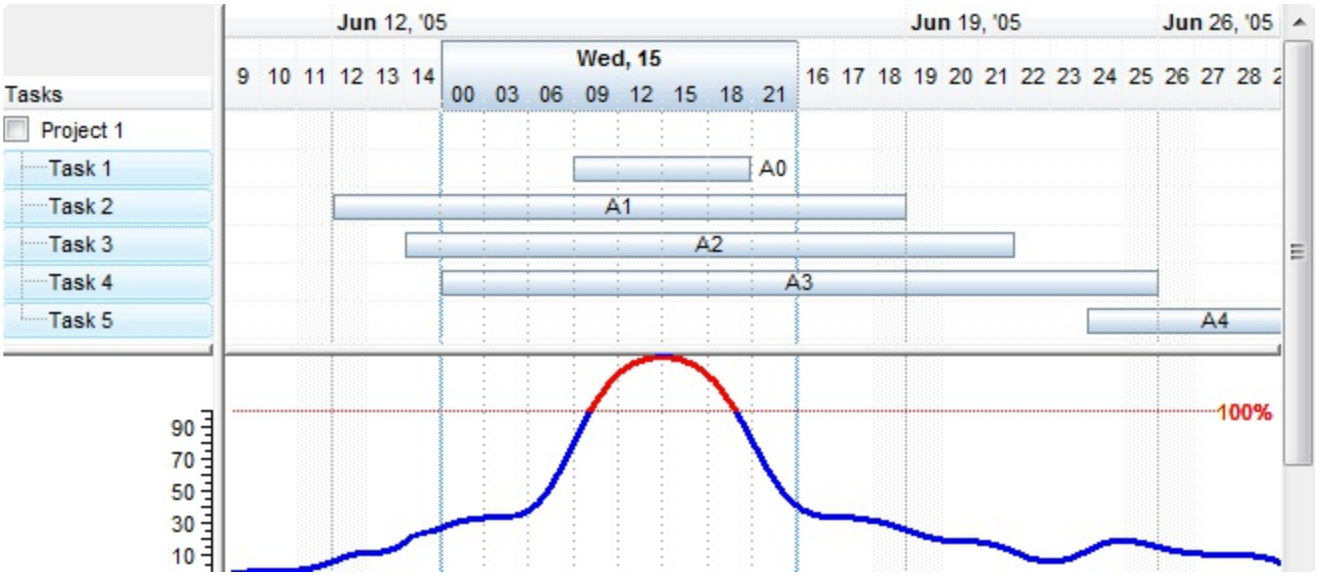
The critical value is interpreted in different way based on the HistogramType property as follows:

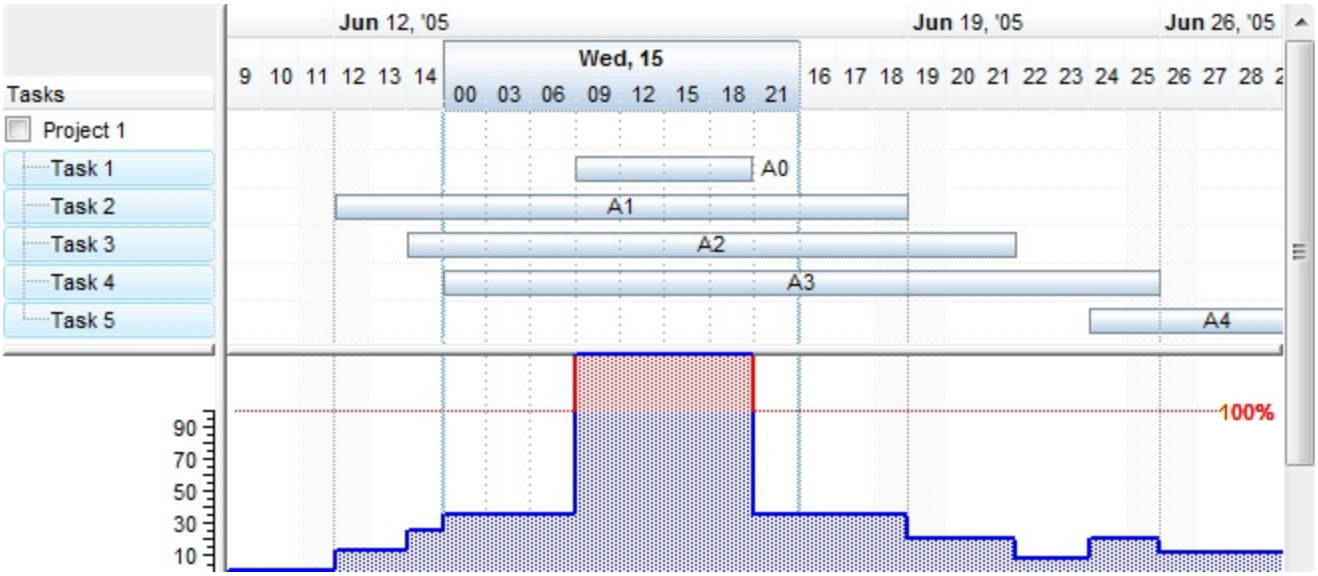
- if the HistogramType property is ***exHistOverload***, the critical value indicates a count of bars. So, if the count of bars in the histogram exceeds the critical value they are shown using the critical color. When using *exHistOverload* you can specify the number of maximum bars being displayed in the histogram using the HistogramItems property. The following screen shot shows the overload histogram, critical value 3, and the ruler displayed on the left side of the histogram displays the count of bars . The red part indicates a portion that has more than 3 bars over.



- if the HistogramType property is ***exHistOverallocation***, the critical value indicates a percent value. So, if the allocation of the bar per unit exceeds the critical value the bars

are show in the critical color. The work-load for a task is computed as ItemBar(exBarEffort) / length of the bar. The work-load for the task is the work effort / task duration. (i.e. If item.exBarEffort = 1 and gantt bar length is 10 days, then the work-load = 0.1 or 10%). The following screen shots show the critical part in red when exHistOverallocation type is displayed:





property Bar.HistogramCumulativeColor(Index as Long) as Color

Retrieves or sets a value that indicates a cumulative color to be shown in the histogram.

Type	Description
Index as Long	A long expression that specifies the index of the color being requested
Color	A Color expression that specifies a cumulative color

A cumulative histogram shows bars that generated the histogram using different colors. The cumulative histogram shows overloads or work-loads as well. The HistogramCumulativeColors property specifies the number of colors that the histogram may show when it displays bars using cumulative type. *The histogram shows cumulative values only if the HistogramCumulativeColors property is greater than 1, the HistogramType property includes the exHistCumulative flag.* Use the HistogramCumulativeOriginalColorBars property to specify whether the bars that generated the cumulative histogram change their original colors. The HistogramCumulativeShowLegend property specifies the index of the column to show the legend for the items being displayed in the cumulative histogram.

By default, the HistogramCumulativeColor values are:

1. RGB(000, 000, 255)
2. RGB(255, 000, 000)
3. RGB(000, 255, 000)
4. RGB(000, 255, 255)
5. RGB(255, 000, 255)
6. RGB(255, 255, 000)

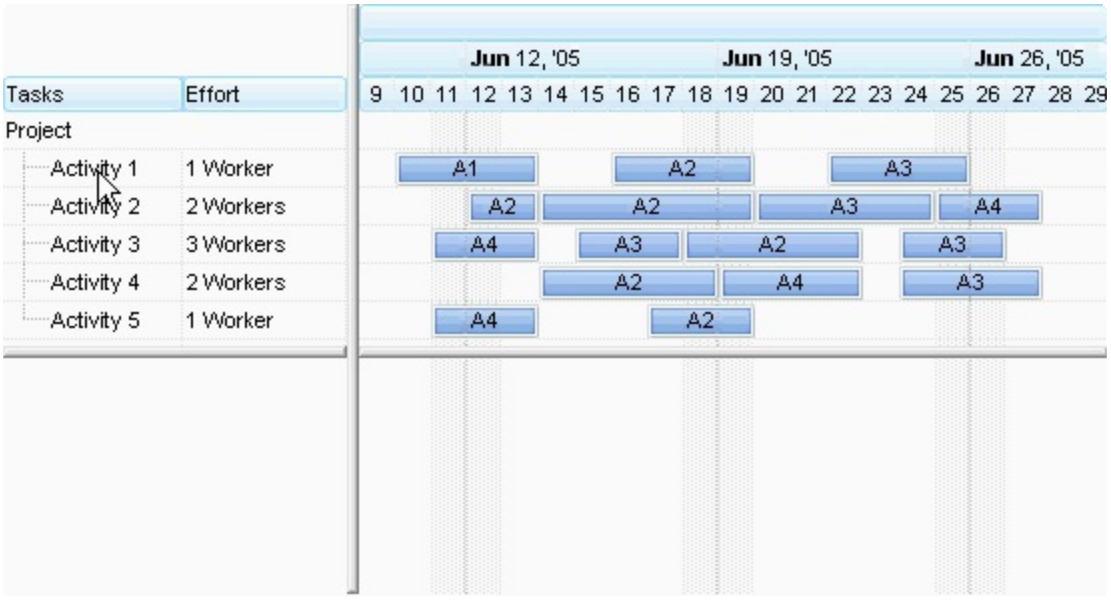
Use the HistogramCumulativeColor property to define different colors for your cumulative histogram.

property Bar.HistogramCumulativeColors as Long

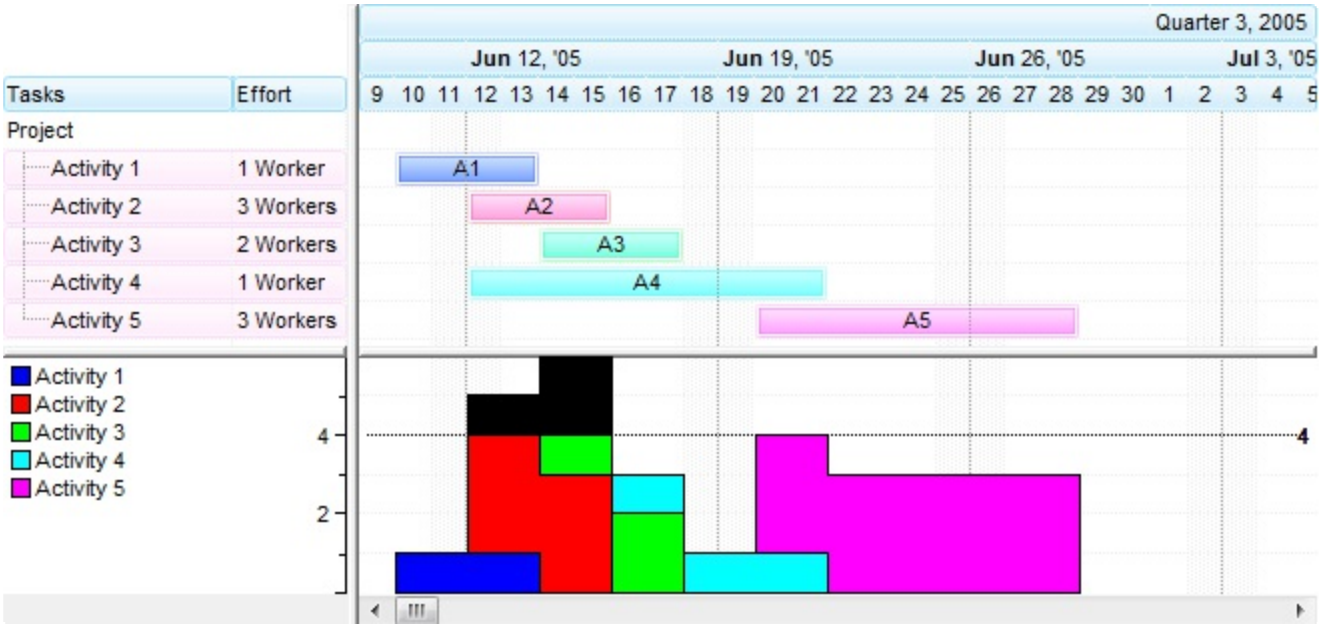
Specifies the number of colors that the histogram may show when it displays bars using cumulative type.

Type	Description
Long	A long expression that specifies the number of colors being used when a cumulative histogram is shown for the current bar. The value should be greater than 1, else the property has no effect.

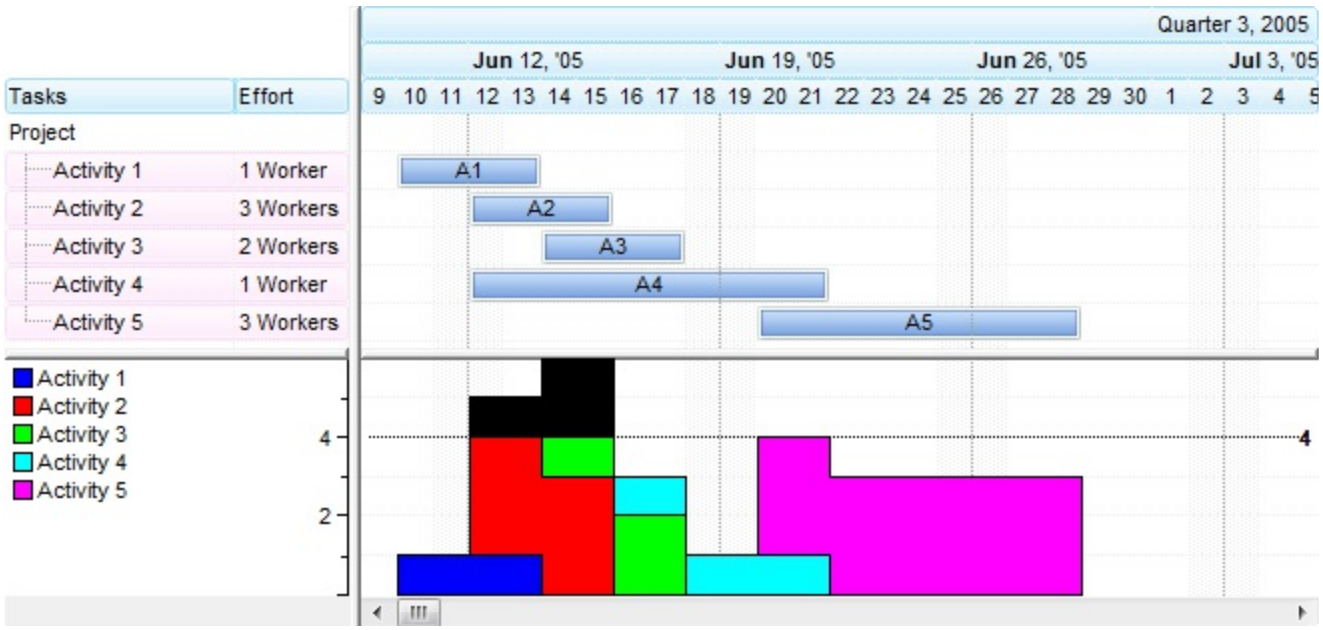
By default, the HistogramCumulativeColors property is 6. A cumulative histogram shows bars that generated the histogram using different colors. The cumulative histogram shows overloads or work-loads as well. *The histogram shows cumulative values only if the HistogramCumulativeColors property is greater than 1, the HistogramType property includes the exHistCumulative flag.* The HistogramCumulativeColor property specifies a color being used when showing cumulative histogram. Use the HistogramCumulativeOriginalColorBars property to specify whether the bars that generated the cumulative histogram change their original colors. The HistogramCumulativeShowLegend property specifies the index of the column to show the legend for the items being displayed in the cumulative histogram.



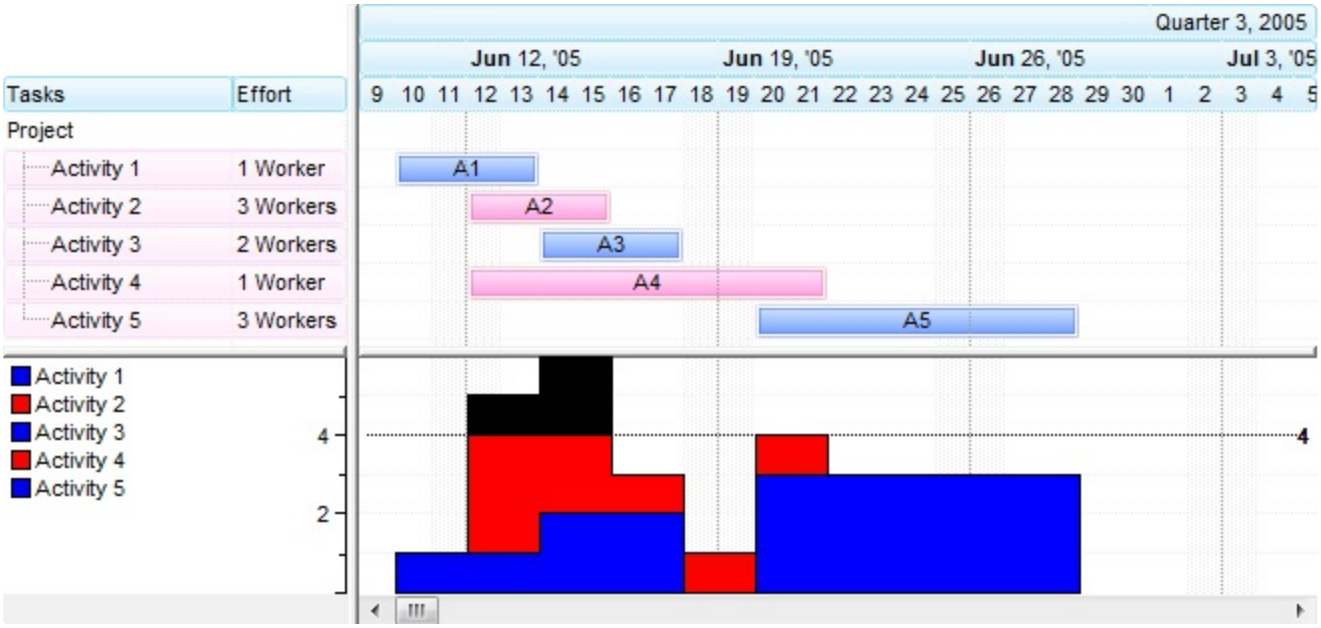
The following screen shot shows the cumulative histogram for HistogramCumulativeColors = 6, HistogramCumulativeOriginalColorBars = False



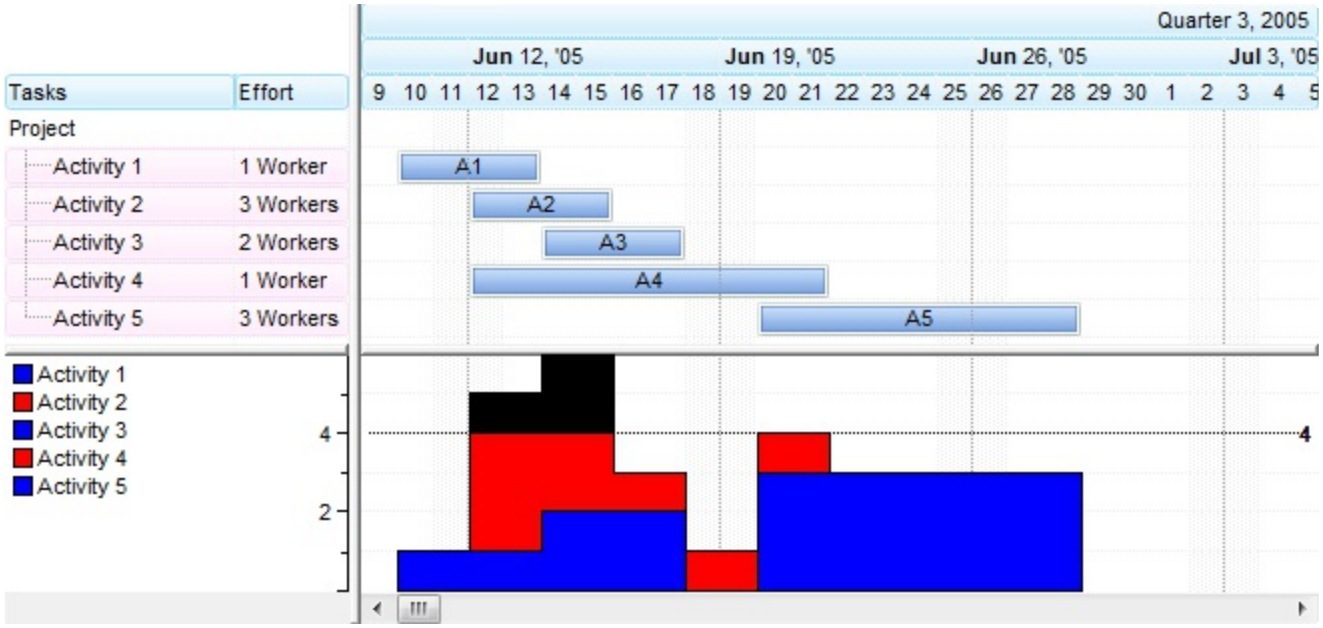
The following screen shot shows the cumulative histogram for HistogramCumulativeColors = 6, HistogramCumulativeOriginalColorBars = True



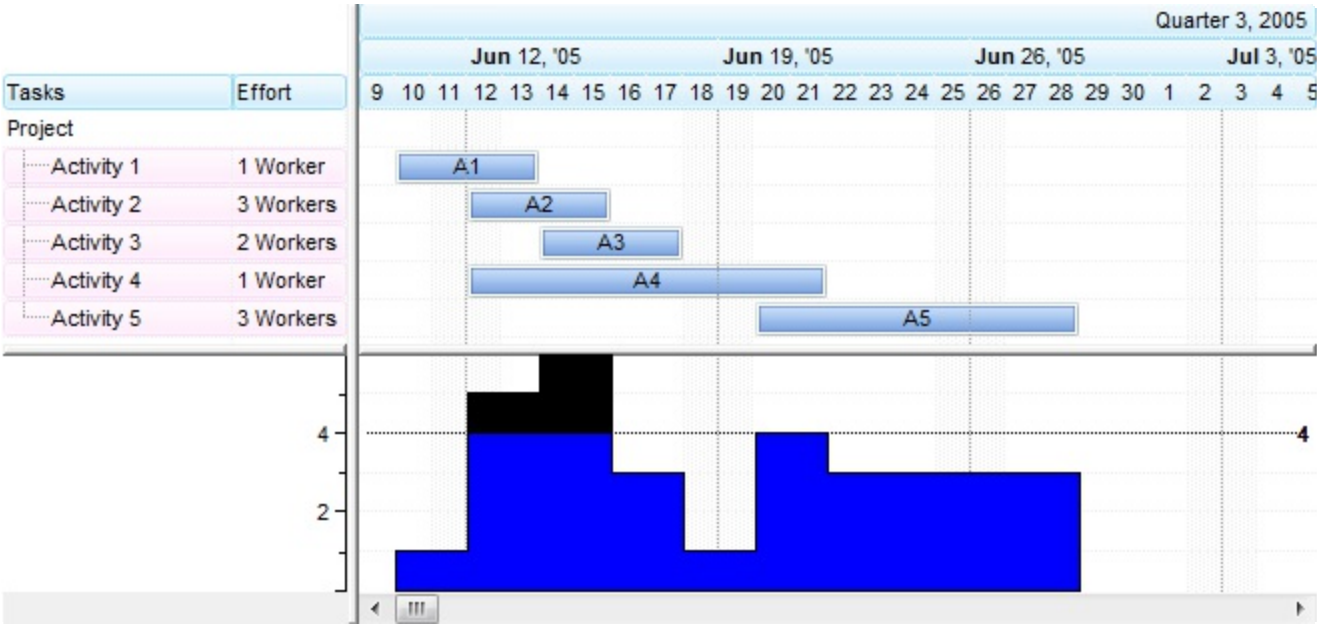
The following screen shot shows the cumulative histogram for HistogramCumulativeColors = 2, HistogramCumulativeOriginalColorBars = True



The following screen shot shows the cumulative histogram for HistogramCumulativeColors = 2, HistogramCumulativeOriginalColorBars = False



The following screen shot shows the cumulative histogram for HistogramCumulativeColors = 1 (as it would not have effect)


Quarter 3, 2005
Jun 12, '05
Jun 19, '05
Jun 26, '05
Jul 3, '05
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 1 2 3 4 5
Tasks
Effort
Project
Activity 1
1 Worker
Activity 2
3 Workers
Activity 3
2 Workers
Activity 4
1 Worker
Activity 5
3 Workers
A1
A2
A3
A4
A5
4
2
4

property Bar.HistogramCumulativeOriginalColorBars as HistogramCumulativeOriginalColorBarsEnum

Specifies whether the original bar's color is changed accordingly to the cumulative histogram.

Type	Description
HistogramCumulativeOriginalColorBarsEnum	A HistogramCumulativeOriginalColorBarsEnum expression that specifies whether the color of the bars that generated the histogram are changed.

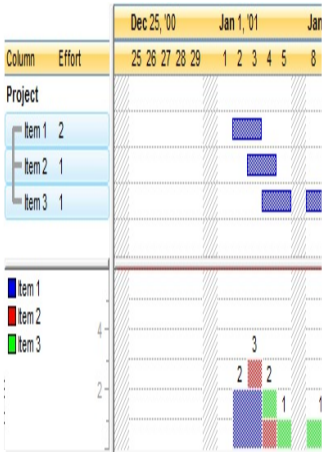
By default, the HistogramCumulativeOriginalColorBars property is -1 (True). If the HistogramCumulativeOriginalColorBars property is 0 (False), the bars that generate the cumulative histogram shows with a different color being specified by the HistogramCumulativeColor property. The HistogramCumulativeShowLegend property specifies the index of the column to show the legend for the items being displayed in the cumulative histogram. You can specify a non-existing column so the legend will not be shown in the left part of the histogram. Use the ItemBar(exBarColor) property to specify a different color for a specified bar. If the HistogramCumulativeOriginalColorBars property is 0 only bars that has the ItemBar(exBarColor) on 0 show with a different cumulative color. The ItemBar(exBarHistLegend) property specifies the description to show within the histogram's legend for the bar in the control's histogram (exKeepOriginalColor only).

If the HistogramCumulativeOriginalColorBars property is:

- -1, exShowCumulativeColor (default), the bars in the same item are represented in the histogram with a cumulative color. The color of the bar in the chart is not changed.
- 0, exChangeColor, the bars and their reflections in the histogram use a cumulative color to be shown. The color of the bar in the chart is changed accordingly to the cumulative color.
- 1, exKeepOriginalColor, the bars keeps their original color in the chart as in the histogram. The color of the bar in the chart is not changed. The ItemBar(exBarColor) property indicates the bar's color in histogram-representation as well. The ItemBar(exBarHistLegend) property specifies the description to show within the histogram's legend for the bar in the control's histogram (exKeepOriginalColor only).

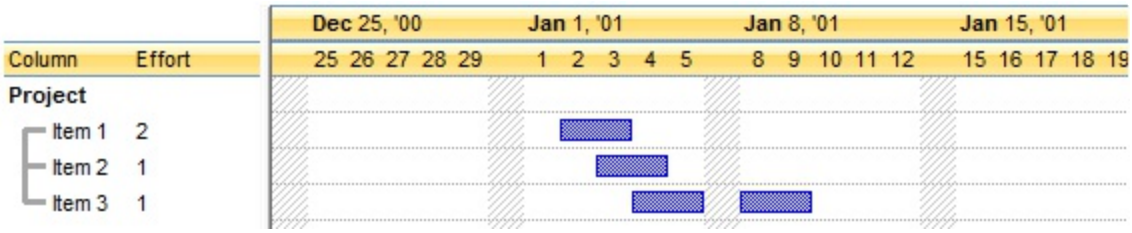
The following screen shots shows the histogram using different values for HistogramCumulativeOriginalColorBars property:

Original Layout	Histogram Represent **HistogramCumulativeOri** is
	-1 (True), by de

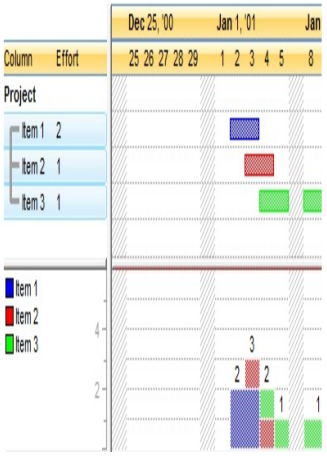


The bars in the same item represented in the histogra cumulative color. The color the chart is not changed.

This is the original layout, so no items are selected, and the effort for the bar in the "Item 1" is 2, and all bars has the same color.



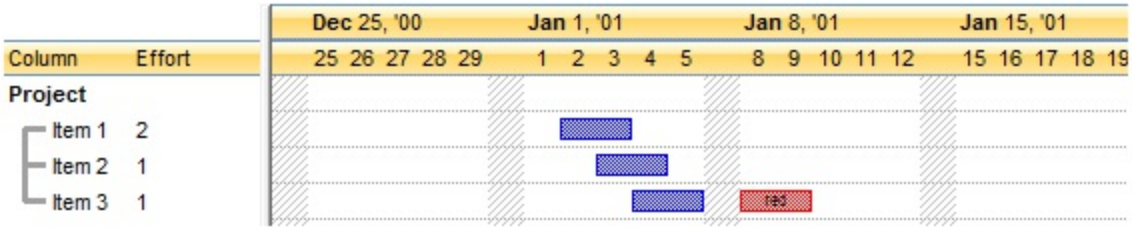
0 (False) is



The bars and their reflectio histogram use a cumulative shown. The color of the ba changed accordingly to the color.

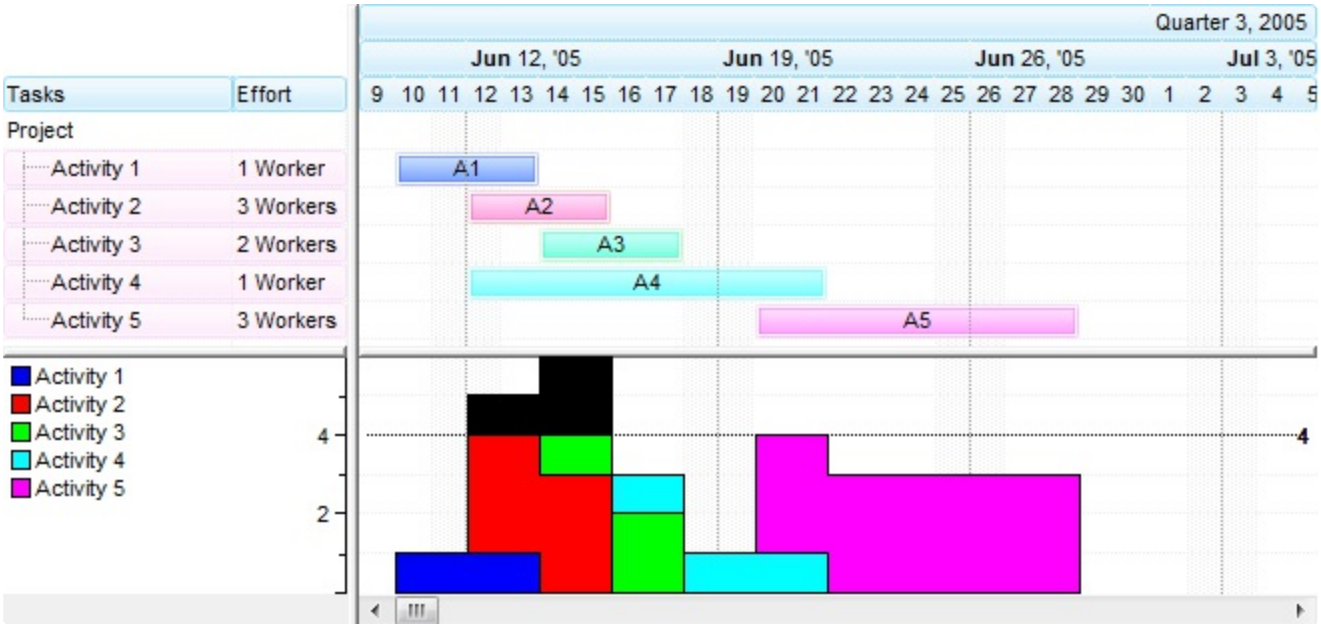
1 (exKeepOriginalC

This is the original layout, so no items are selected, and the effort for the bar in the "Item 1" is 2, and the item "Item 3" contains a red bar

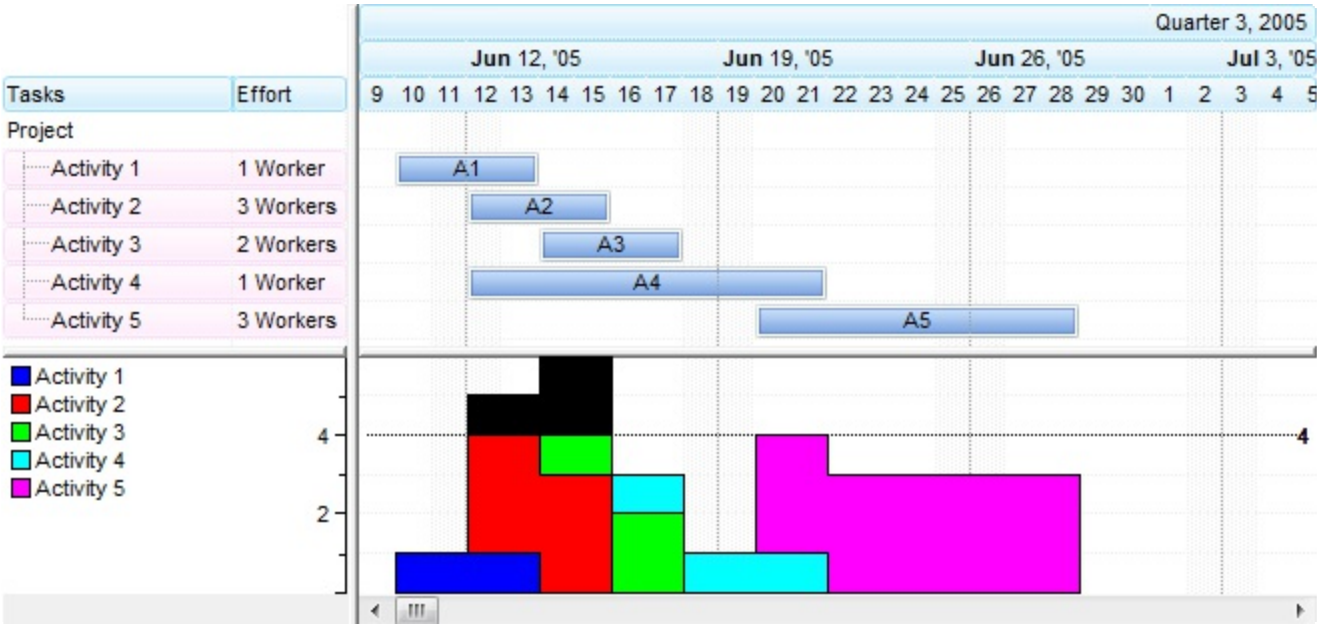


The bars keeps their origir
chart as in the histogram.
bar in the chart is not chan
ItemBar(exBarColor) prop
the local bar's color. This c
available for newer versior

The following screen shot shows the cumulative histogram for HistogramCumulativeColors = 8, HistogramCumulativeOriginalColorBars = 0 (False). In this case the A1, A2, A3, A4 and A5 bars are shown in the chart using a different color as specified by the cumulative histogram.



The following screen shot shows the cumulative histogram for HistogramCumulativeColors = 8, HistogramCumulativeOriginalColorBars = -1 (True). In this case the A1, A2, A3, A4 and A5 bars shows in the chart with the color to show the cumulative histogram.


Quarter 3, 2005
Jun 12, '05
Jun 19, '05
Jun 26, '05
Jul 3, '05
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 1 2 3 4 5
Tasks
Effort
Project
Activity 1
1 Worker
Activity 2
3 Workers
Activity 3
2 Workers
Activity 4
1 Worker
Activity 5
3 Workers
A1
A2
A3
A4
A5
Activity 1
Activity 2
Activity 3
Activity 4
Activity 5
4
2
4

property Bar.HistogramCumulativeShowLegend as Long

Specifies the index of the column to display the legend for the cumulative bars in the histogram.

Type	Description
Long	A long expression that specifies the index of the column to show the legend for items being displayed in the cumulative histogram.

By default, the HistogramCumulativeShowLegend property is 0 which means that the column with the index 0 is used to display the legend for the items being included in the histogram. The HistogramCumulativeShowLegend property has effect only if the HistogramType property includes the exHistCumulative flag. The HistogramCumulativeColors property specifies the number of colors being used to display the cumulative histogram for a bar. The HistogramCumulativeColor property specifies a color being used when showing cumulative histogram. Use the HistogramCumulativeOriginalColorBars property to specify whether the bars that generated the cumulative histogram change their original colors. Use the Item property to access a column in the columns collection.

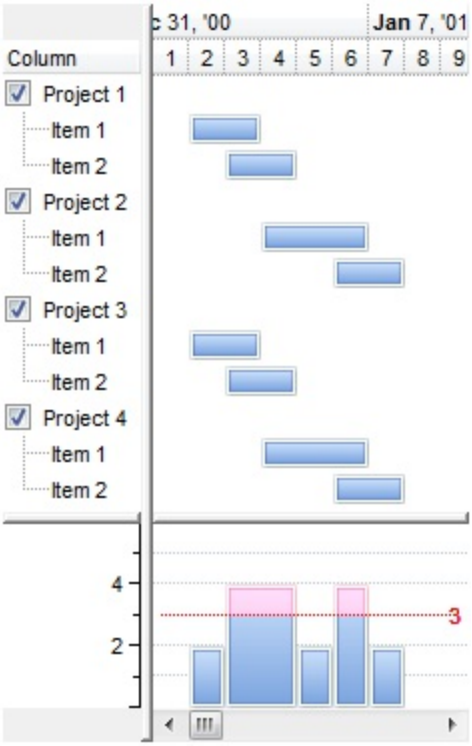
property Bar.HistogramGridLinesColor as Color

Retrieves or sets a value that indicates the color to show the histogram's grid lines.

Type	Description
Color	A Color expression that specifies whether the grid lines are shown in the histogram. Use the value on 1, incase you actually need a black grid lines color.

By default, the HistogramGridLinesColor property is 0, which means that it has no effect. If the HistogramGridLinesColor property is not 0, it indicates the color to show the grid lines in the histogram. The grid lines for the histogram are shown only if the HistogramType property is exOverload and they are shown only in the right part of the histogram. Instead, you can use the HistogramRulerLinesColor property to specify the color to show the bar's histogram ruler that are always shown in the left part of the histogram. You can always use the BeforeDrawPart/AfterDrawPart events to provide your custom drawing in the histogram.

The following screen show shows the grid lines color in the right side of the histogram:



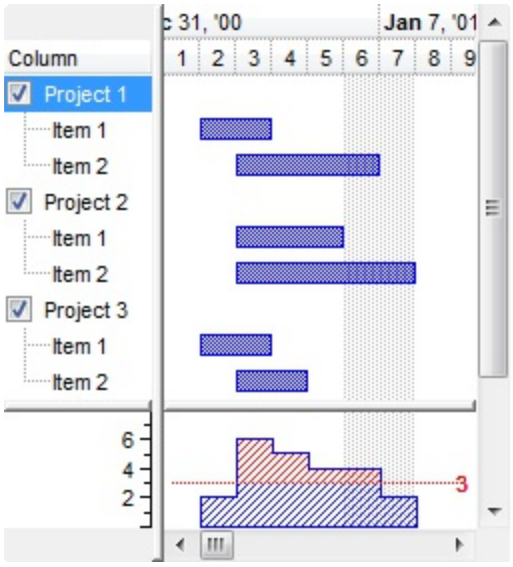
property Bar.HistogramItems as Long

Specifies the number of items being represented in the histogram when overload histogram is shown.

Type	Description
Long	A long expression that specifies the minimum number of items being show in the histogram, if positive, or a fixed number of items, if negative. If 0, the histogram is re-scaled to fit all elements.

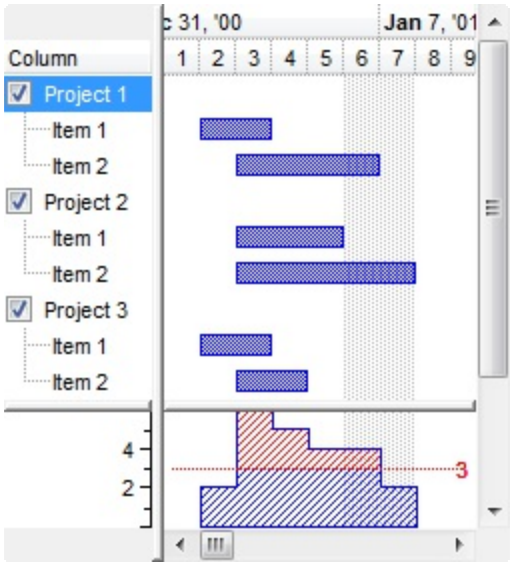
By default, the HistogramItems property is 0. The property has effect while the HistogramType property is exOverload. By default, when the HistogramItems property is 0, the height of the units being displayed in the histogram is computed, so all units fit the histogram area. If the HistogramItems property is greater than 0 this value indicates the number of minimum units being displayed on the vertical axis. If the HistogramItems property less than 0, the absolute value represents the fixed number of units being displayed on the vertical axis.

For instance, the following sample shows the histogram while HistogramItems property is 0:



If the HistogramItems property is 0, and the user resizes the histogram the height of the units being displayed is automatically updated so all units fit the histogram area. Also, if the user includes or excludes items to be shown in the histogram the height of the unit is recomputed. For instance, if a single item is being included in the histogram the entire height of the histogram specifies the height for the unit for the bars. Instead, if the HistogramItems property is greater than 0, the height of the units being displayed in the histogram is changed only if requires multiple units being displayed on the vertical axis.

The following sample shows the histogram while HistogramItems property is -6:



In this case, HistogramItems property is -6, so the number of units being displayed on the vertical axis is always 6 no matter how many units are required.

The following VB sample specifies a fixed number of units being shown in the histogram on the vertical axis:

```
With G2antt1
   With .Chart
      .FirstVisibleDate = #1/1/2001#
      .HistogramVisible = True
      .HistogramHeight = 64
      .PaneWidth(0) = 78
      .HistogramView = 1300
      With .Bars.Item("Task")
         .HistogramPattern = exPatternBDiagonal
         .HistogramCriticalValue = 3
         .HistogramItems = -6
         .HistogramRulerLinesColor = RGB(1,0,0)
      End With
   End With
   .Columns.Add "Column"
   With .Items
      h = .AddItem("Project 1")
      .CellHasCheckBox(h,0) = True
      .CellState(h,0) = 1
      .AddBar .InsertItem(h,0,"Item 1"),"Task",#1/2/2001#,#1/4/2001#
      .AddBar .InsertItem(h,0,"Item 2"),"Task",#1/3/2001#,#1/5/2001#
```

```
      .ExpandItem(h) = True
      h = .AddItem("Project 2")
      .CellHasCheckBox(h,0) = True
      .CellState(h,0) = 1
      .AddBar .InsertItem(h,0,"Item 1"),"Task",#1/4/2001#,#1/7/2001#
      .AddBar .InsertItem(h,0,"Item 2"),"Task",#1/6/2001#,#1/8/2001#
      .ExpandItem(h) = True
      h = .AddItem("Project 3")
      .CellHasCheckBox(h,0) = True
      .CellState(h,0) = 1
      .AddBar .InsertItem(h,0,"Item 1"),"Task",#1/2/2001#,#1/4/2001#
      .AddBar .InsertItem(h,0,"Item 2"),"Task",#1/3/2001#,#1/5/2001#
      .ExpandItem(h) = True
      h = .AddItem("Project 4")
      .CellHasCheckBox(h,0) = True
      .CellState(h,0) = 1
      .AddBar .InsertItem(h,0,"Item 1"),"Task",#1/4/2001#,#1/7/2001#
      .AddBar .InsertItem(h,0,"Item 2"),"Task",#1/6/2001#,#1/8/2001#
      .ExpandItem(h) = True
   End With
End With
```

The following VB.NET sample specifies a fixed number of units being shown in the histogram on the vertical axis:

```
Dim h
With AxG2antt1
   With .Chart
      .FirstVisibleDate = #1/1/2001#
      .HistogramVisible = True
      .HistogramHeight = 64
      .PaneWidth(0) = 78
      .HistogramView = 1300
      With .Bars.Item("Task")
         .HistogramPattern = EXG2ANTTLib.PatternEnum.exPatternBDiagonal
         .HistogramCriticalValue = 3
         .HistogramItems = -6
         .HistogramRulerLinesColor = 1
```

```
    End With
  End With
  .Columns.Add "Column"
  With .Items
    h = .AddItem("Project 1")
    .CellHasCheckBox(h,0) = True
    .CellState(h,0) = 1
    .AddBar .InsertItem(h,0,"Item 1"),"Task",#1/2/2001#,#1/4/2001#
    .AddBar .InsertItem(h,0,"Item 2"),"Task",#1/3/2001#,#1/5/2001#
    .ExpandItem(h) = True
    h = .AddItem("Project 2")
    .CellHasCheckBox(h,0) = True
    .CellState(h,0) = 1
    .AddBar .InsertItem(h,0,"Item 1"),"Task",#1/4/2001#,#1/7/2001#
    .AddBar .InsertItem(h,0,"Item 2"),"Task",#1/6/2001#,#1/8/2001#
    .ExpandItem(h) = True
    h = .AddItem("Project 3")
    .CellHasCheckBox(h,0) = True
    .CellState(h,0) = 1
    .AddBar .InsertItem(h,0,"Item 1"),"Task",#1/2/2001#,#1/4/2001#
    .AddBar .InsertItem(h,0,"Item 2"),"Task",#1/3/2001#,#1/5/2001#
    .ExpandItem(h) = True
    h = .AddItem("Project 4")
    .CellHasCheckBox(h,0) = True
    .CellState(h,0) = 1
    .AddBar .InsertItem(h,0,"Item 1"),"Task",#1/4/2001#,#1/7/2001#
    .AddBar .InsertItem(h,0,"Item 2"),"Task",#1/6/2001#,#1/8/2001#
    .ExpandItem(h) = True
  End With
End With
```

The following C++ sample specifies a fixed number of units being shown in the histogram on the vertical axis:

```
/*
  Copy and paste the following directives to your header file as
  it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'
```

```
   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutFirstVisibleDate("1/1/2001");
   var_Chart->PutHistogramVisible(VARIANT_TRUE);
   var_Chart->PutHistogramHeight(64);
   var_Chart->PutPaneWidth(0,78);
   var_Chart->PutHistogramView((EXG2ANTTLib::HistogramViewEnum)1300);
   EXG2ANTTLib::IBarPtr var_Bar = var_Chart->GetBars()->GetItem("Task");
      var_Bar->PutHistogramPattern(EXG2ANTTLib::exPatternBDiagonal);
      var_Bar->PutHistogramCriticalValue(3);
      var_Bar->PutHistogramItems(-6);
      var_Bar->PutHistogramRulerLinesColor(RGB(1,0,0));
spG2antt1->GetColumns()->Add(L"Column");
EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems();
   long h = var_Items->AddItem("Project 1");
   var_Items->PutCellHasCheckBox(h,long(0),VARIANT_TRUE);
   var_Items->PutCellState(h,long(0),1);
   var_Items->AddBar(var_Items->InsertItem(h,long(0),"Item
1"),"Task","1/2/2001","1/4/2001",vtMissing,vtMissing);
   var_Items->AddBar(var_Items->InsertItem(h,long(0),"Item
2"),"Task","1/3/2001","1/5/2001",vtMissing,vtMissing);
   var_Items->PutExpandItem(h,VARIANT_TRUE);
   h = var_Items->AddItem("Project 2");
   var_Items->PutCellHasCheckBox(h,long(0),VARIANT_TRUE);
   var_Items->PutCellState(h,long(0),1);
   var_Items->AddBar(var_Items->InsertItem(h,long(0),"Item
1"),"Task","1/4/2001","1/7/2001",vtMissing,vtMissing);
   var_Items->AddBar(var_Items->InsertItem(h,long(0),"Item
2"),"Task","1/6/2001","1/8/2001",vtMissing,vtMissing);
   var_Items->PutExpandItem(h,VARIANT_TRUE);
   h = var_Items->AddItem("Project 3");
   var_Items->PutCellHasCheckBox(h,long(0),VARIANT_TRUE);
   var_Items->PutCellState(h,long(0),1);
```

```
   var_Items->AddBar(var_Items->InsertItem(h,long(0),"Item
1"),"Task","1/2/2001","1/4/2001",vtMissing,vtMissing);
   var_Items->AddBar(var_Items->InsertItem(h,long(0),"Item
2"),"Task","1/3/2001","1/5/2001",vtMissing,vtMissing);
   var_Items->PutExpandItem(h,VARIANT_TRUE);
   h = var_Items->AddItem("Project 4");
   var_Items->PutCellHasCheckBox(h,long(0),VARIANT_TRUE);
   var_Items->PutCellState(h,long(0),1);
   var_Items->AddBar(var_Items->InsertItem(h,long(0),"Item
1"),"Task","1/4/2001","1/7/2001",vtMissing,vtMissing);
   var_Items->AddBar(var_Items->InsertItem(h,long(0),"Item
2"),"Task","1/6/2001","1/8/2001",vtMissing,vtMissing);
   var_Items->PutExpandItem(h,VARIANT_TRUE);
```

The following C# sample specifies a fixed number of units being shown in the histogram on the vertical axis:

```
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.FirstVisibleDate = "1/1/2001";
   var_Chart.HistogramVisible = true;
   var_Chart.HistogramHeight = 64;
   var_Chart.set_PaneWidth(0 != 0,78);
   var_Chart.HistogramView = (EXG2ANTTLib.HistogramViewEnum)1300;
   EXG2ANTTLib.Bar var_Bar = var_Chart.Bars["Task"];
      var_Bar.HistogramPattern = EXG2ANTTLib.PatternEnum.exPatternBDiagonal;
      var_Bar.HistogramCriticalValue = 3;
      var_Bar.HistogramItems = -6;
      var_Bar.HistogramRulerLinesColor = 1;
axG2antt1.Columns.Add("Column");
EXG2ANTTLib.Items var_Items = axG2antt1.Items;
   int h = var_Items.AddItem("Project 1");
   var_Items.set_CellHasCheckBox(h,0,true);
   var_Items.set_CellState(h,0,1);
   var_Items.AddBar(var_Items.InsertItem(h,0,"Item
1"),"Task","1/2/2001","1/4/2001",null,null);
   var_Items.AddBar(var_Items.InsertItem(h,0,"Item
2"),"Task","1/3/2001","1/5/2001",null,null);
   var_Items.set_ExpandItem(h,true);
```

```
   h = var_Items.AddItem("Project 2");
   var_Items.set_CellHasCheckBox(h,0,true);
   var_Items.set_CellState(h,0,1);
   var_Items.AddBar(var_Items.InsertItem(h,0,"Item
1"),"Task","1/4/2001","1/7/2001",null,null);
   var_Items.AddBar(var_Items.InsertItem(h,0,"Item
2"),"Task","1/6/2001","1/8/2001",null,null);
   var_Items.set_ExpandItem(h,true);
   h = var_Items.AddItem("Project 3");
   var_Items.set_CellHasCheckBox(h,0,true);
   var_Items.set_CellState(h,0,1);
   var_Items.AddBar(var_Items.InsertItem(h,0,"Item
1"),"Task","1/2/2001","1/4/2001",null,null);
   var_Items.AddBar(var_Items.InsertItem(h,0,"Item
2"),"Task","1/3/2001","1/5/2001",null,null);
   var_Items.set_ExpandItem(h,true);
   h = var_Items.AddItem("Project 4");
   var_Items.set_CellHasCheckBox(h,0,true);
   var_Items.set_CellState(h,0,1);
   var_Items.AddBar(var_Items.InsertItem(h,0,"Item
1"),"Task","1/4/2001","1/7/2001",null,null);
   var_Items.AddBar(var_Items.InsertItem(h,0,"Item
2"),"Task","1/6/2001","1/8/2001",null,null);
   var_Items.set_ExpandItem(h,true);
```

The following VFP sample specifies a fixed number of units being shown in the histogram on the vertical axis:

```
with thisform.G2antt1
   with .Chart
      .FirstVisibleDate = {^2001-1-1}
      .HistogramVisible = .T.
      .HistogramHeight = 64
      .PaneWidth(0) = 78
      .HistogramView = 1300
      with .Bars.Item("Task")
         .HistogramPattern = 6
         .HistogramCriticalValue = 3
```

```
        .HistogramItems = -6
        .HistogramRulerLinesColor = RGB(1,0,0)
    endwith
endwith
.Columns.Add("Column")
with .Items
    h = .AddItem("Project 1")
    .DefaultItem = h
    .CellHasCheckBox(0,0) = .T.
    .DefaultItem = h
    .CellState(0,0) = 1
    .AddBar(.InsertItem(h,0,"Item 1"),"Task",{^2001-1-2},{^2001-1-4})
    .AddBar(.InsertItem(h,0,"Item 2"),"Task",{^2001-1-3},{^2001-1-5})
    .DefaultItem = h
    .ExpandItem(0) = .T.
    h = .AddItem("Project 2")
    .DefaultItem = h
    .CellHasCheckBox(0,0) = .T.
    .DefaultItem = h
    .CellState(0,0) = 1
    .AddBar(.InsertItem(h,0,"Item 1"),"Task",{^2001-1-4},{^2001-1-7})
    .AddBar(.InsertItem(h,0,"Item 2"),"Task",{^2001-1-6},{^2001-1-8})
    .DefaultItem = h
    .ExpandItem(0) = .T.
    h = .AddItem("Project 3")
    .DefaultItem = h
    .CellHasCheckBox(0,0) = .T.
    .DefaultItem = h
    .CellState(0,0) = 1
    .AddBar(.InsertItem(h,0,"Item 1"),"Task",{^2001-1-2},{^2001-1-4})
    .AddBar(.InsertItem(h,0,"Item 2"),"Task",{^2001-1-3},{^2001-1-5})
    .DefaultItem = h
    .ExpandItem(0) = .T.
    h = .AddItem("Project 4")
    .DefaultItem = h
    .CellHasCheckBox(0,0) = .T.
    .DefaultItem = h
```

```
    .CellState(0,0) = 1
    .AddBar(.InsertItem(h,0,"Item 1"),"Task",{^2001-1-4},{^2001-1-7})
    .AddBar(.InsertItem(h,0,"Item 2"),"Task",{^2001-1-6},{^2001-1-8})
    .DefaultItem = h
    .ExpandItem(0) = .T.
  endwith
endwith
```

The following Delphi sample specifies a fixed number of units being shown in the histogram on the vertical axis:

```
with AxG2antt1 do
begin
  with Chart do
  begin
    FirstVisibleDate := '1/1/2001';
    HistogramVisible := True;
    HistogramHeight := 64;
    PaneWidth[0 <> 0] := 78;
    HistogramView := EXG2ANTTLib.HistogramViewEnum(1300);
    with Bars.Item['Task'] do
    begin
      HistogramPattern := EXG2ANTTLib.PatternEnum.exPatternBDiagonal;
      HistogramCriticalValue := 3;
      HistogramItems := -6;
      HistogramRulerLinesColor := 1;
    end;
  end;
  Columns.Add('Column');
  with Items do
  begin
    h := AddItem('Project 1');
    CellHasCheckBox[TObject(h),TObject(0)] := True;
    CellState[TObject(h),TObject(0)] := 1;
    AddBar(InsertItem(h,TObject(0),'Item 1'),'Task','1/2/2001','1/4/2001',Nil,Nil);
    AddBar(InsertItem(h,TObject(0),'Item 2'),'Task','1/3/2001','1/5/2001',Nil,Nil);
    ExpandItem[h] := True;
    h := AddItem('Project 2');
```

```
      CellHasCheckBox[TObject(h),TObject(0)] := True;
      CellState[TObject(h),TObject(0)] := 1;
      AddBar(InsertItem(h,TObject(0),'Item 1'),'Task','1/4/2001','1/7/2001',Nil,Nil);
      AddBar(InsertItem(h,TObject(0),'Item 2'),'Task','1/6/2001','1/8/2001',Nil,Nil);
      ExpandItem[h] := True;
      h := AddItem('Project 3');
      CellHasCheckBox[TObject(h),TObject(0)] := True;
      CellState[TObject(h),TObject(0)] := 1;
      AddBar(InsertItem(h,TObject(0),'Item 1'),'Task','1/2/2001','1/4/2001',Nil,Nil);
      AddBar(InsertItem(h,TObject(0),'Item 2'),'Task','1/3/2001','1/5/2001',Nil,Nil);
      ExpandItem[h] := True;
      h := AddItem('Project 4');
      CellHasCheckBox[TObject(h),TObject(0)] := True;
      CellState[TObject(h),TObject(0)] := 1;
      AddBar(InsertItem(h,TObject(0),'Item 1'),'Task','1/4/2001','1/7/2001',Nil,Nil);
      AddBar(InsertItem(h,TObject(0),'Item 2'),'Task','1/6/2001','1/8/2001',Nil,Nil);
      ExpandItem[h] := True;
   end;
end
```

property Bar.HistogramPattern as PatternEnum

Retrieves or sets a value that indicates the pattern to be used in the histogram.

Type	Description
PatternEnum	A PatternEnum expression that specifies the pattern to be used to display the bar in the histogram

By default, the HistogramPattern property is exPatternEmpty. By default, no bar is represented in the histogram. A bar is represented in the histogram only if HistogramPattern or HistogramColor property is set. Use the HistogramColor property to define the *color or the EBN/skin file* of the pattern to be displayed in the histogram. Use the HistogramBackColor property to specify the histogram's background color. Use the HistogramType property to specify the type of the graph to be displayed in the histogram for specified bar. The AntiAliasing property specifies whether lines, curves or edges are shown smoothly using the antialiasing rendering. The HistogramBorderColor property specifies the color to show the border for values in the histogram. The ResizeUnitScale property determines the refinement for the histogram part. For instance, if the chart displays days, while the bars are represented up to hours, the ResizeUnitScale property on exHour, will determine the histogram to show up to hours.

Please follow the steps in order to view your bars in the histogram.

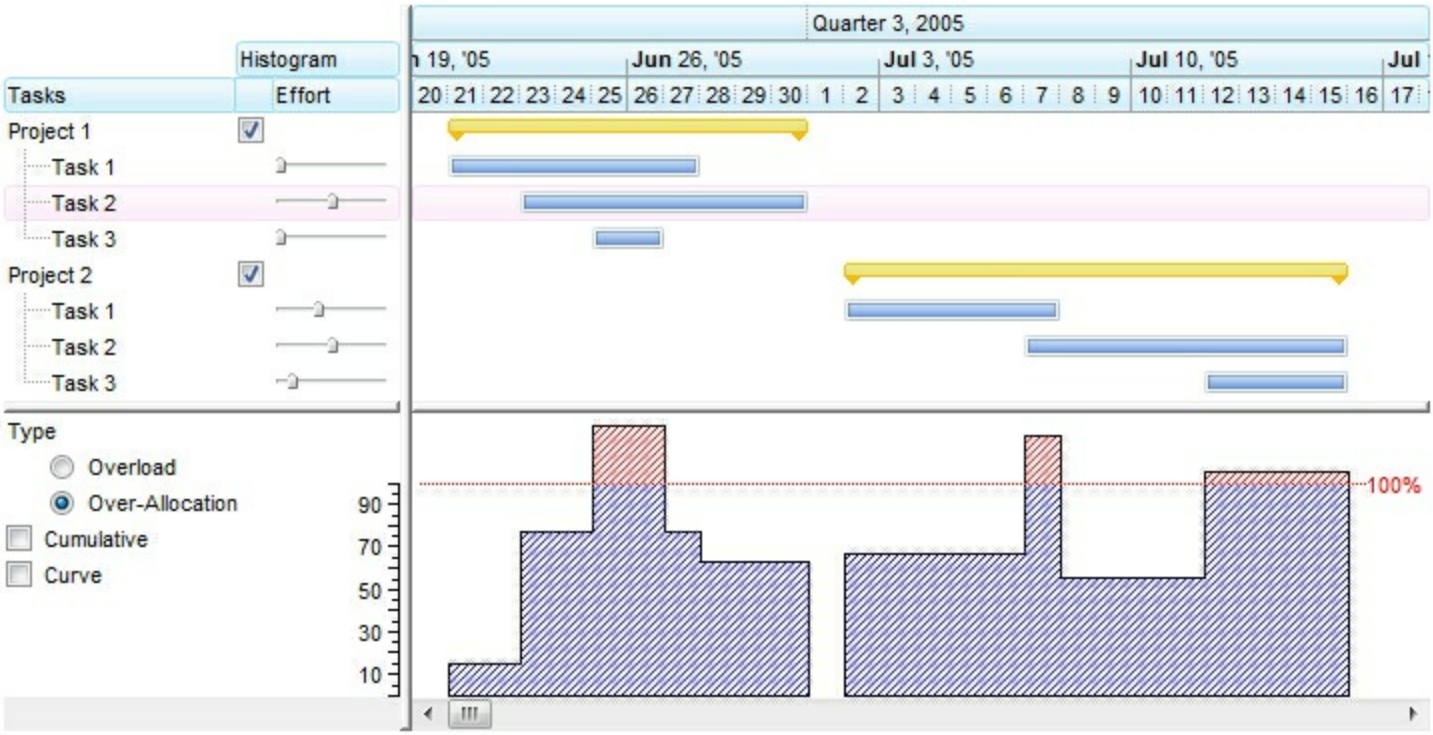
1. Changes the **HistogramVisible** property on True (by default, it is False). After setting the HistogramVisible property on True, the control shows a horizontal splitter in the bottom side of the control.
2. Adjusts the height of the histogram view using the **HistogramHeight** property (by default it is 0). After setting the HistogramHeight property on a value greater than 0, the control shows a the histogram view in the bottom side of the control.
3. Changes the **HistogramPattern or/and HistogramColor** property, else no bars will be shown in the histogram. The HistogramPattern/HistogramColor properties belong to a Bar object. For instance the Chart.Bars("Task").HistogramPattern = exPatternDot, specifies that the Task bars will be represented in the histogram using the exPatternDot pattern ()

The followings are optional properties that you can set in order to customize your histogram:

- The **HistogramType** property indicates the type of the histogram being displayed for a specified bar.
- Use the **HistogramView** property to specify the items being represented in the histogram view. By default, only visible items are displayed in the histogram. For instance, using the HistogramView property you can select the items being represented in the histogram

- Use the HistogramBackColor property to specify the histogram's background color.

The following screen shot shows the bar's diagram if the HistogramPattern property is exPatternBDiagonal. This histogram shows the diagram using a predefined patterns (*value should be greater than 1 and less then 255*).



(HistogramPattern value from 1 to 255)

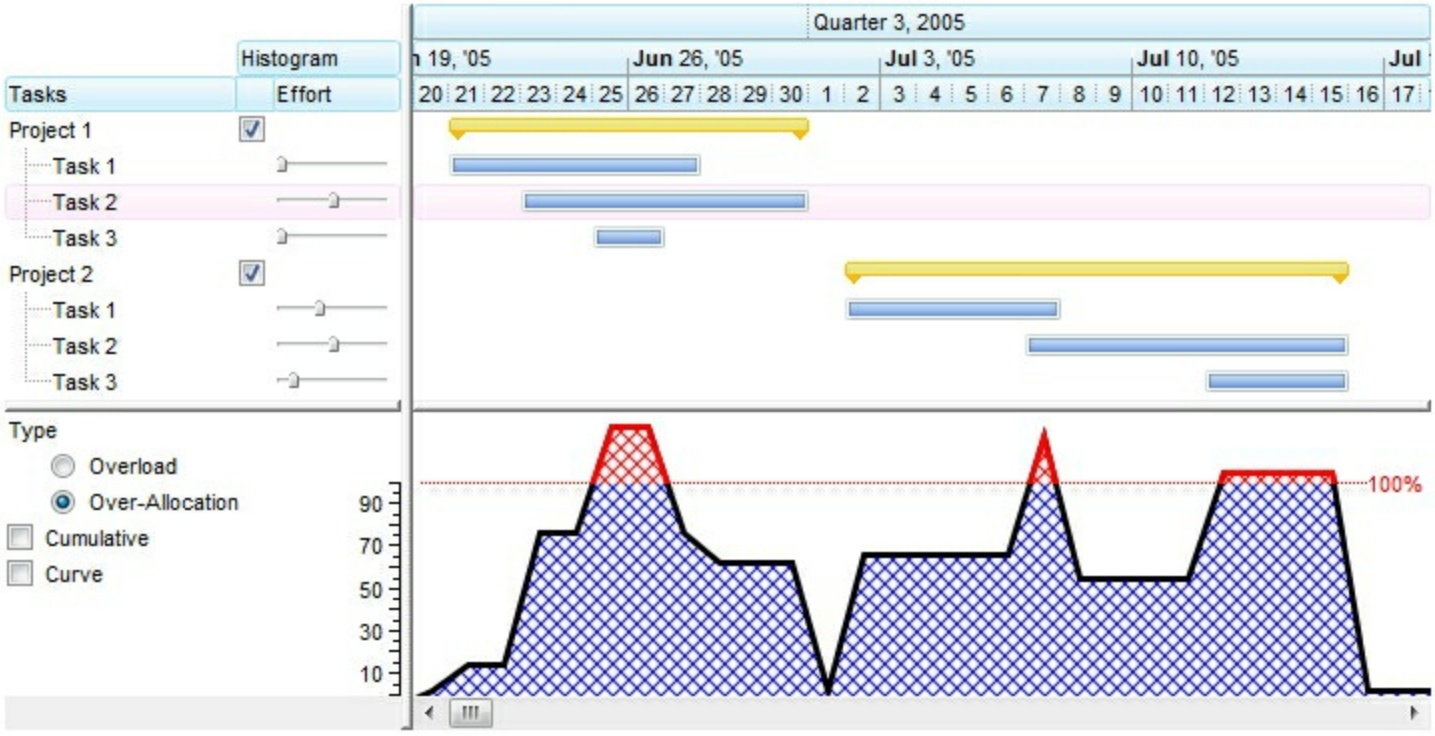
The following screen shot shows the bar's diagram if the HistogramColor property points an EBN object (0x01000000, defines the EBN object being used to show the task bars). In case the HistogramColor property points to an EBN/Skin object (the first 7 bites of the color are used) the HistogramPattern property has no effect, so the EBN object is used to show the diagram.

(HistogramColor defines the EBN/Skin object, HistogramPattern does not count)

The following screen shot shows the bar's diagram if the HistogramColor property points an EBN object (0x01000000, defines the EBN object being used to show headers)

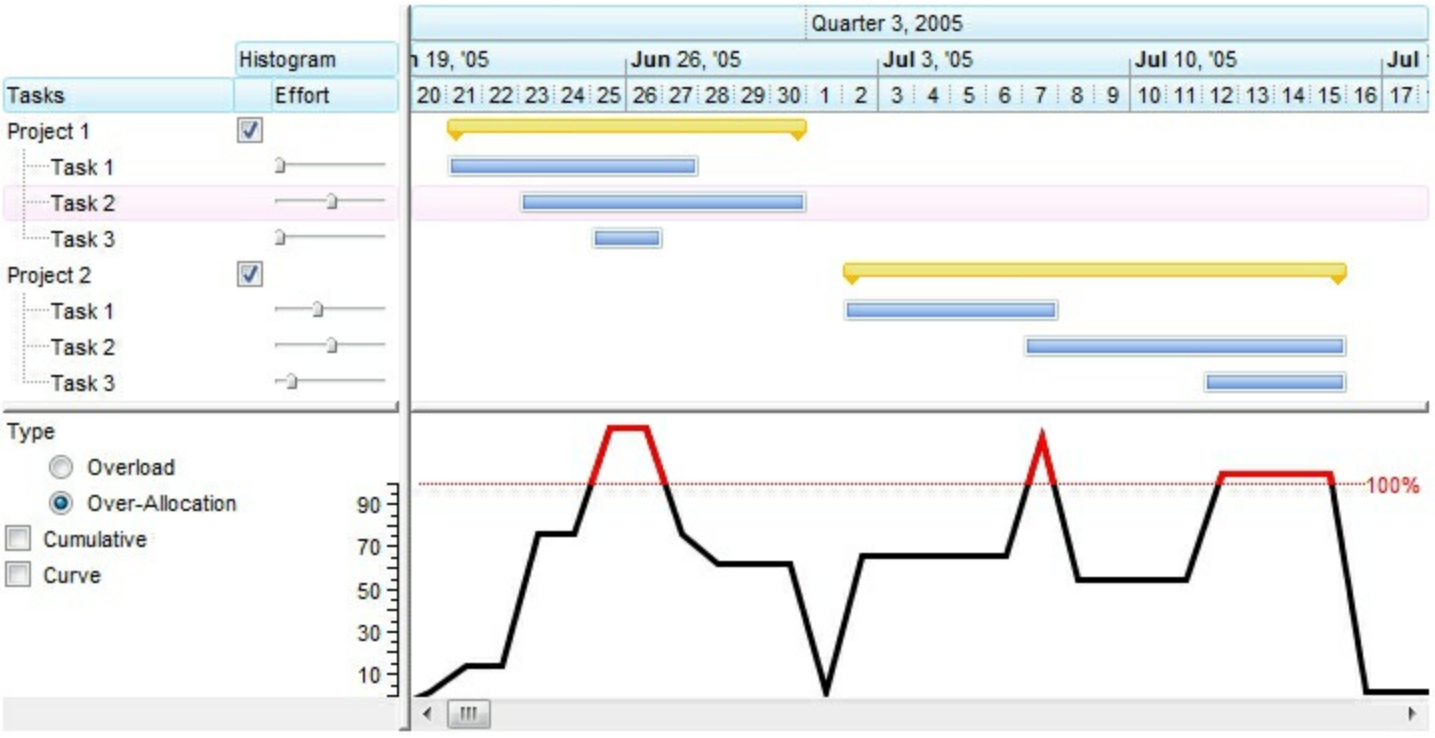
(HistogramColor defines the EBN/Skin file used, HistogramPattern does not count)

The following screen shot shows the bar's diagram if the HistogramPattern property is **exPolygonCurve** + exPatternYard



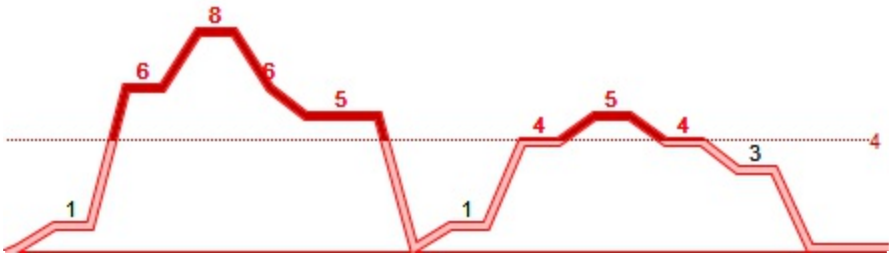
(HistogramPattern value is 256 + value from (1 to 255))

The following screen shot shows the bar's diagram if the HistogramPattern property is **exPolygonCurve** + **exPatternEmpty**, or simple **exPolygonCurve**



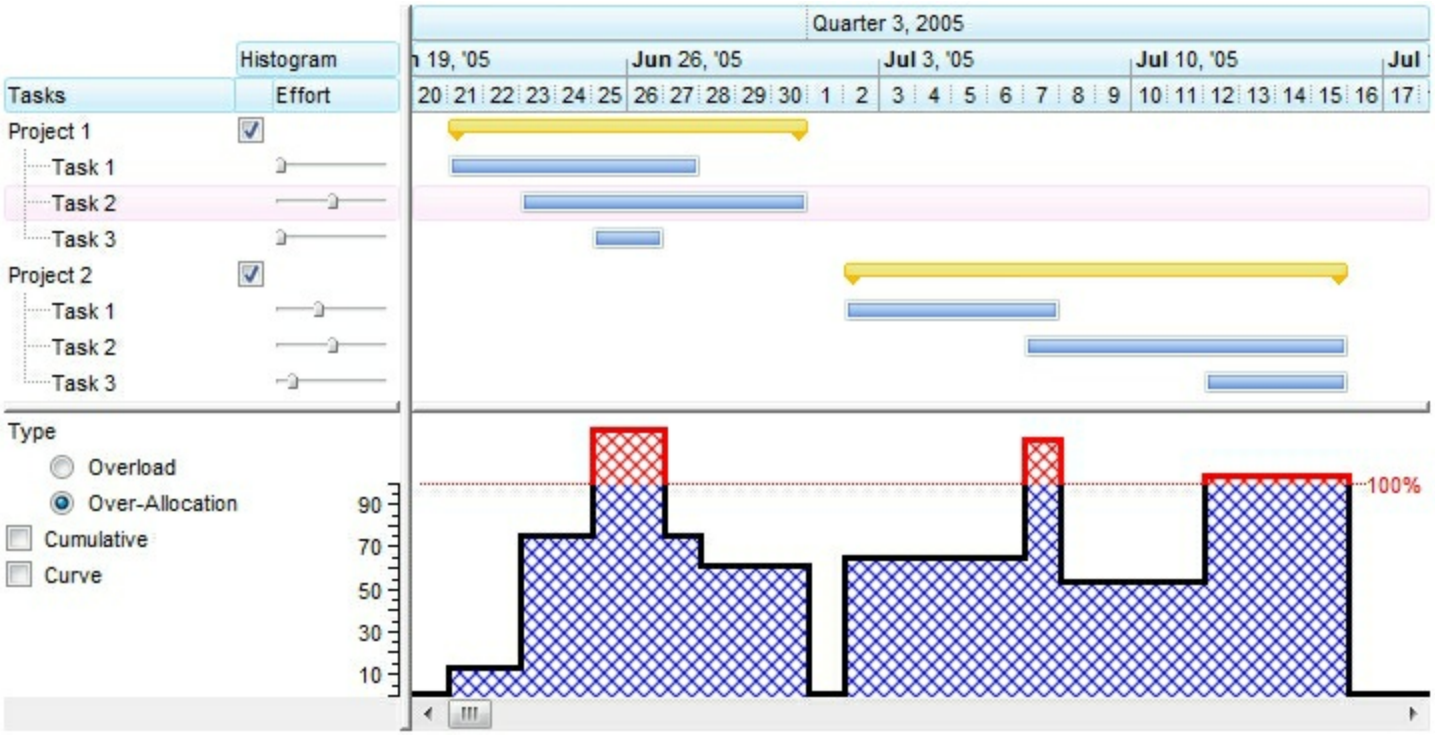
(HistogramPattern value is 256)

The following screen shot shows the bar's diagram if the HistogramPattern property is **exPolygonCurve** + **exPatternEmpty**, or simple **exPolygonCurve**, the AntiAliasing property is True, and the HistogramColor and HistogramBorderColor propertis have different values.



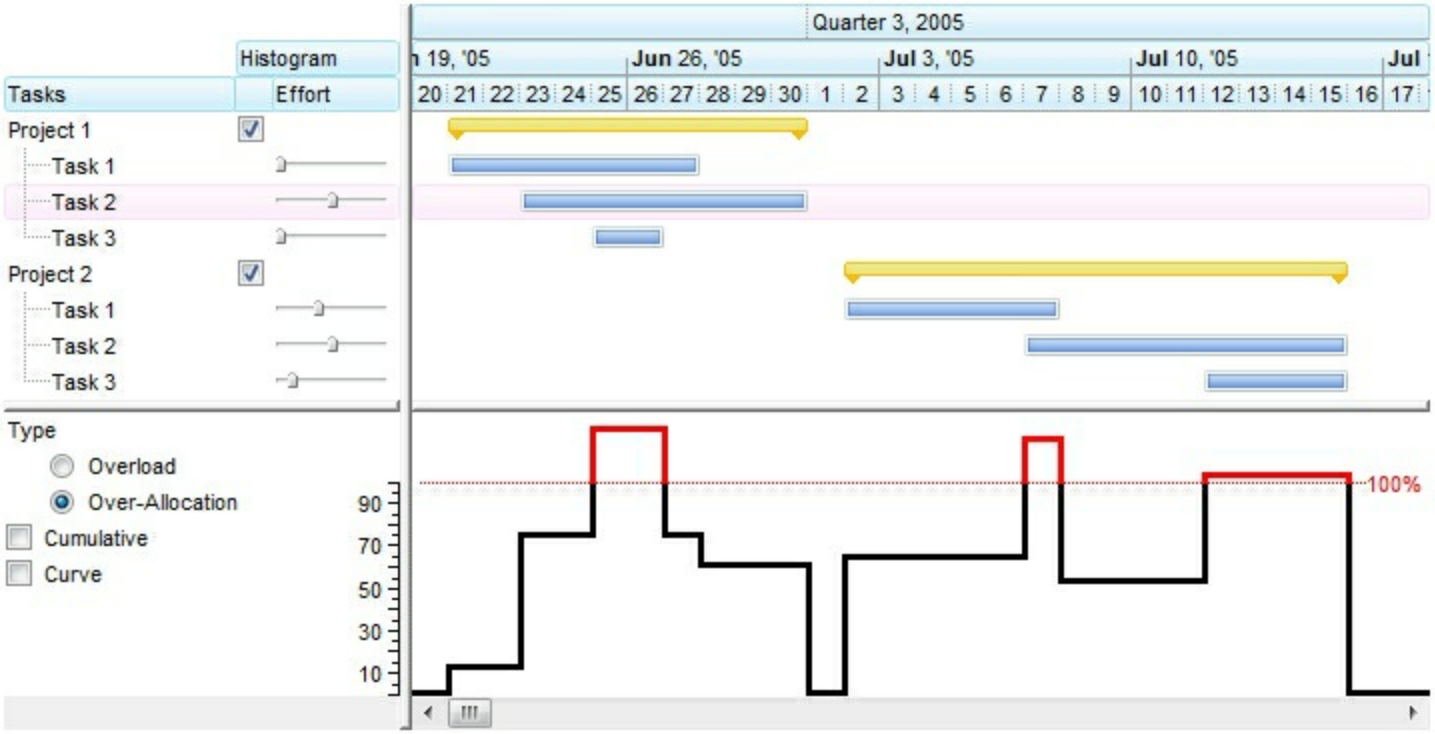
(HistogramPattern value is 256, AntiAliasing = True, HistogramColor != HistogramBorderColor)

The following screen shot shows the bar's diagram if the HistogramPattern property is **exRectangularCurve** + exPatternYard



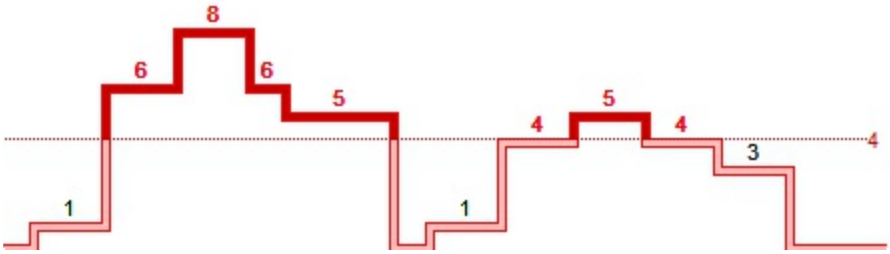
(HistogramPattern value is 2048 *+ value from (1 to 255))*

The following screen shot shows the bar's diagram if the HistogramPattern property is **exRectangularCurve** + **exPatternEmpty,** or simple **exRectangularCurve**



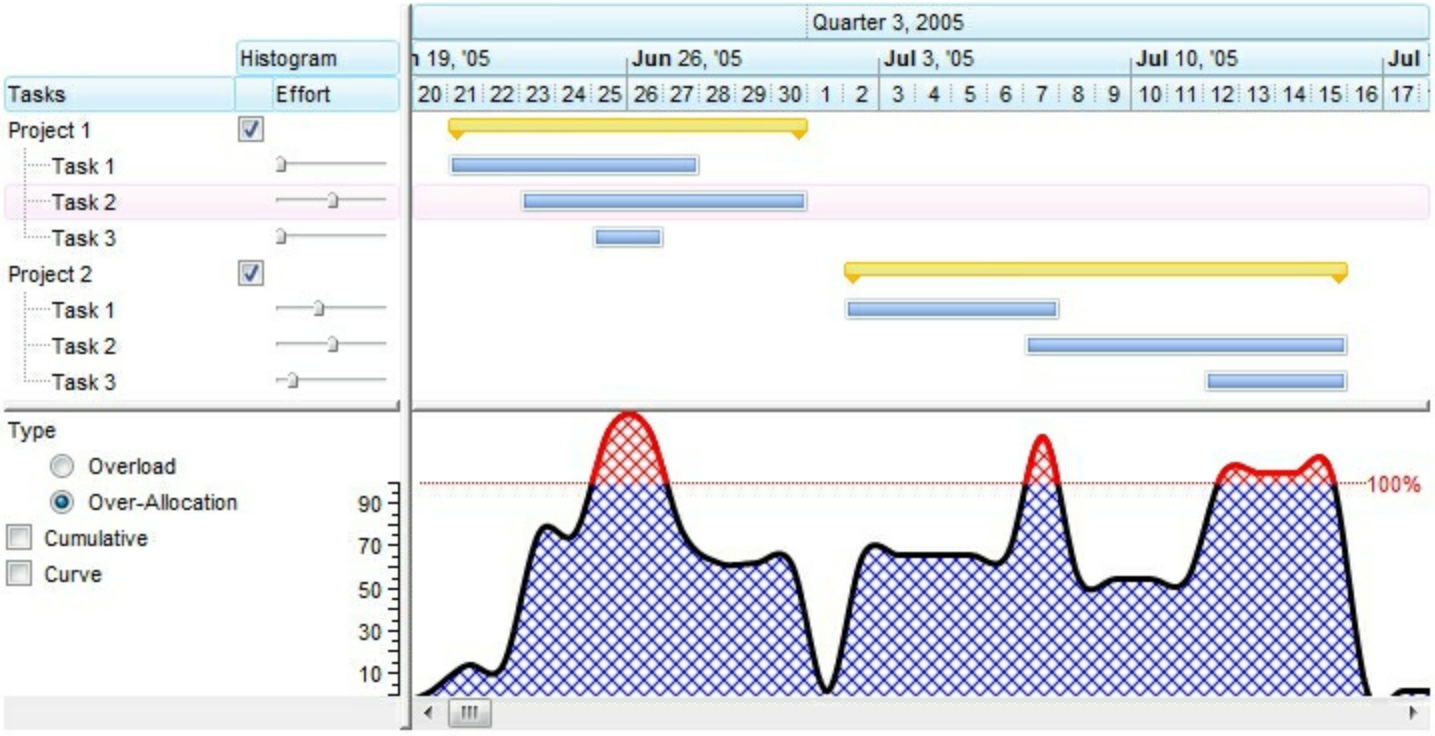
(HistogramPattern value is 2048 *)*

The following screen shot shows the bar's diagram if the HistogramPattern property is **exRectangularCurve** + **exPatternEmpty**, or simple **exRectangularCurve**, the AntiAliasing property is True, and the HistogramColor and HistogramBorderColor propertis have different values.



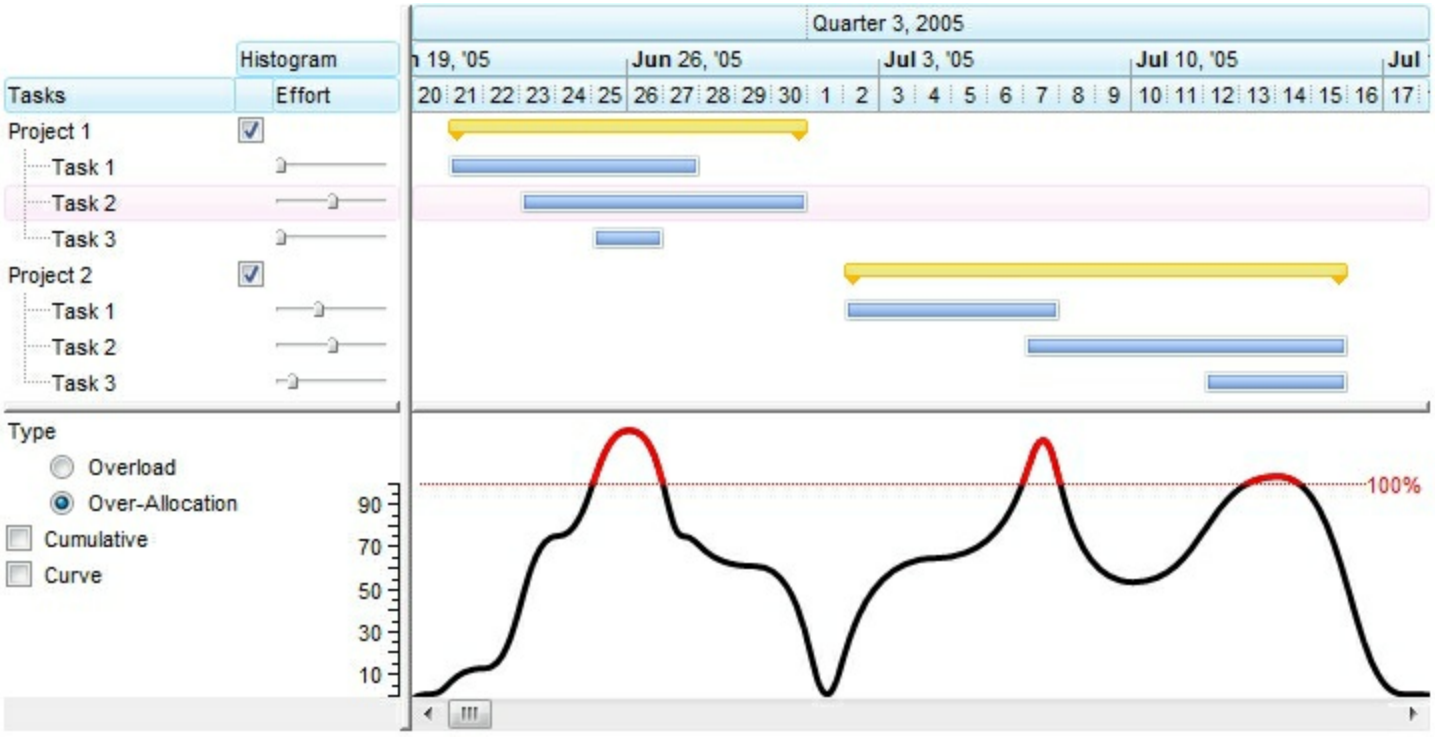
(HistogramPattern value is 2048, AntiAliasing = True, HistogramColor != HistogramBorderColor)

The following screen shot shows the bar's diagram if the HistogramPattern property is **exBezierCurve** + exPatternYard



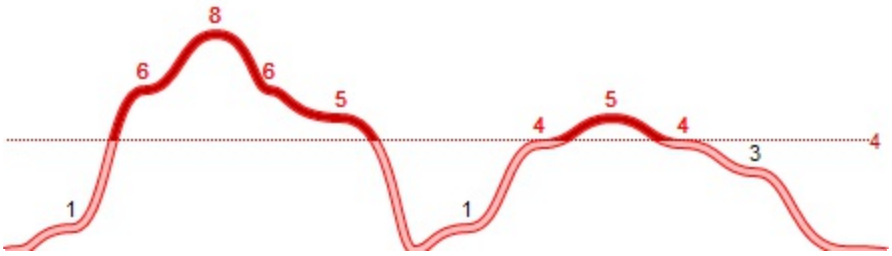
(HistogramPattern value is 512 + *value from (1 to 255))*

The following screen shot shows the bar's diagram if the HistogramPattern property is **exBezierCurve** + **exPatternEmpty**, or simple **exBezierCurve**



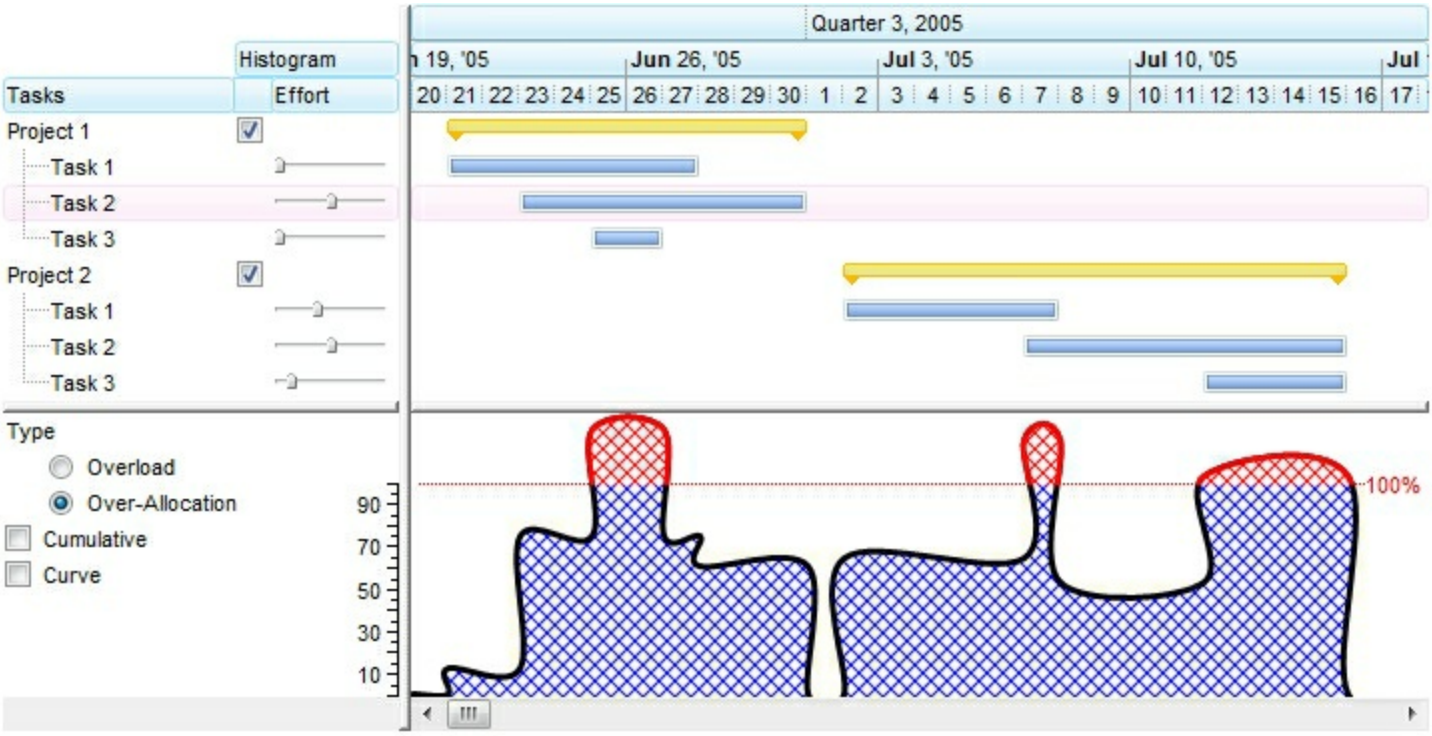
(HistogramPattern value is 512 *)*

The following screen shot shows the bar's diagram if the HistogramPattern property is **exBezierCurve** + **exPatternEmpty**, or simple **exBezierCurve**, the AntiAliasing property is True, and the HistogramColor and HistogramBorderColor propertis have different values.



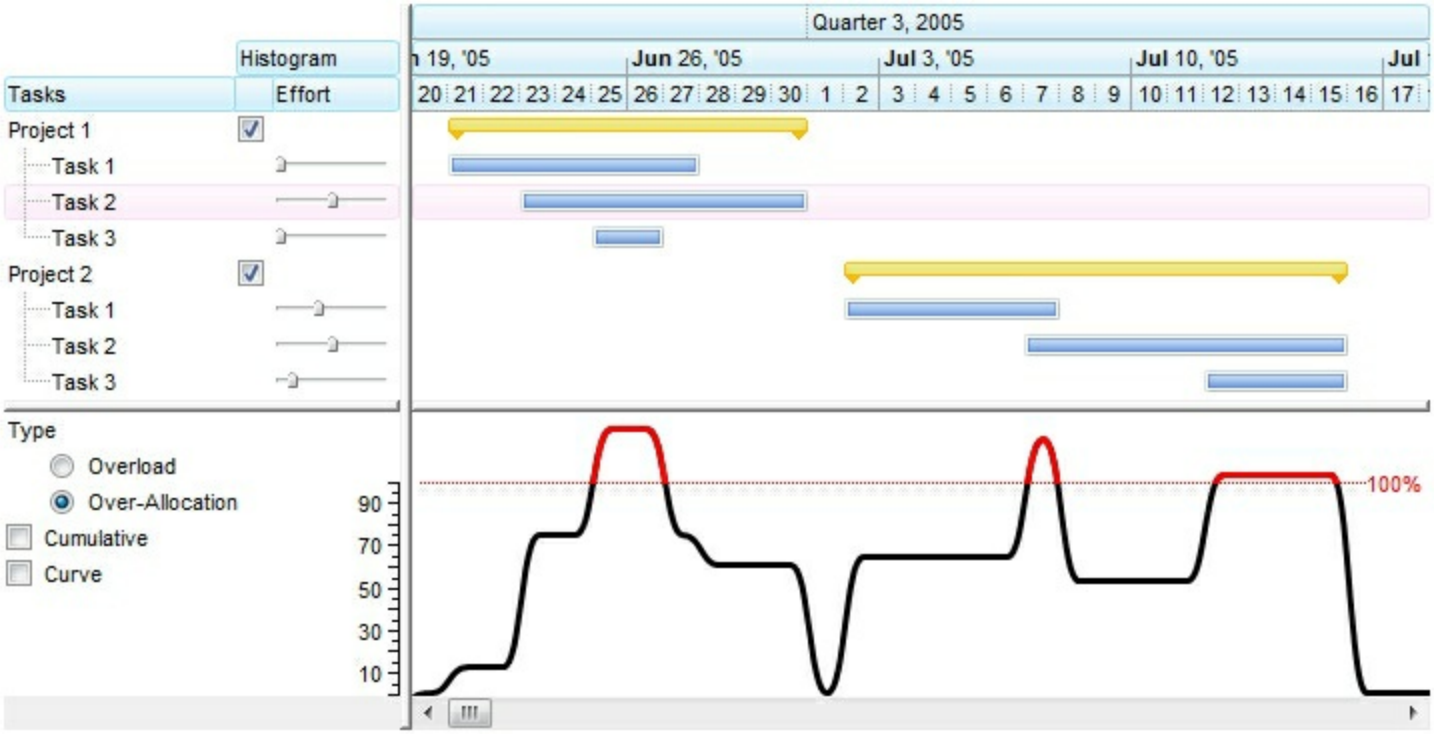
(HistogramPattern value is 512, AntiAliasing = True, HistogramColor != HistogramBorderColor)

The following screen shot shows the bar's diagram if the HistogramPattern property is **exRoundCurve** + exPatternYard



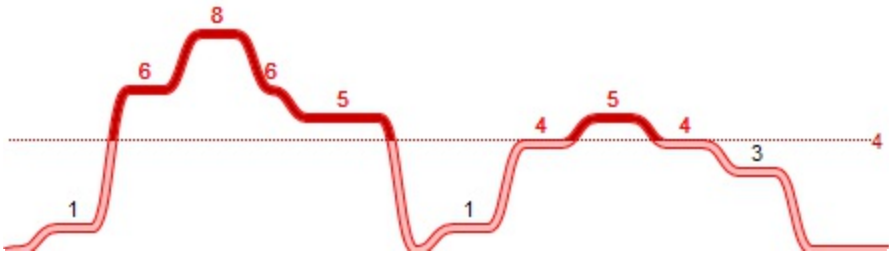
(HistogramPattern value is 1024 *+ value from (1 to 255))*

The following screen shot shows the bar's diagram if the HistogramPattern property is **exRoundCurve** + **exPatternEmpty**, or simple **exRoundCurve**



(HistogramPattern value is 1024 *)*

The following screen shot shows the bar's diagram if the HistogramPattern property is **exRoundCurve** + **exPatternEmpty**, or simple **exRoundCurve**, the AntiAliasing property is True, and the HistogramColor and HistogramBorderColor propertis have different values.

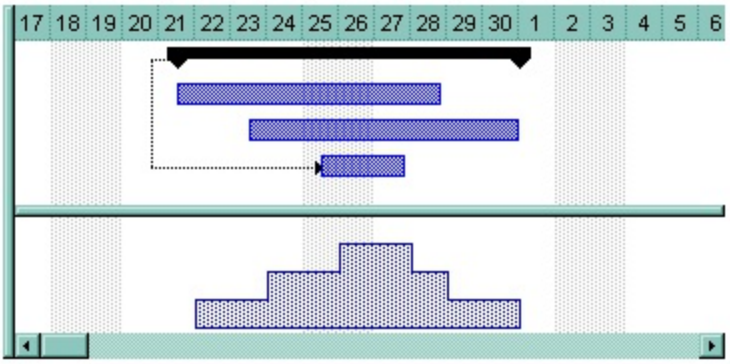


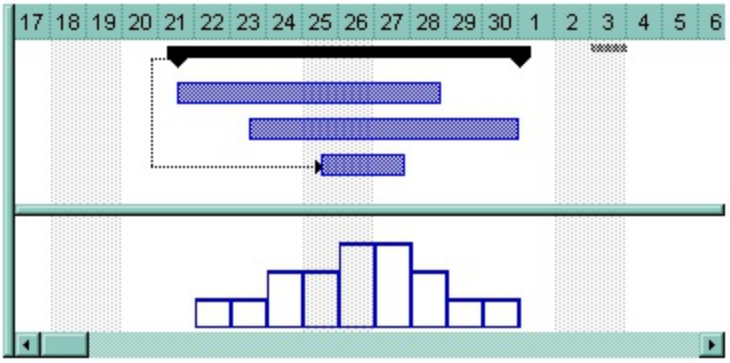
(HistogramPattern value is 1024, AntiAliasing = True, HistogramColor != HistogramBorderColor)

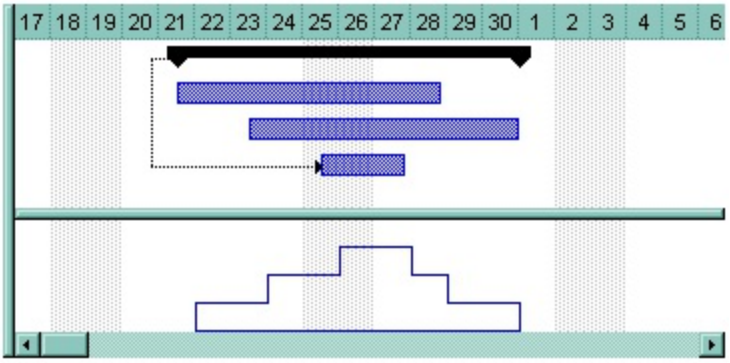
The following screen shot shows the bar's diagram if the HistogramPattern property is exPatternShadow, the HistogramType property includes the exHistCumulative flag, and the HistogramCumulativeOriginalColorBars property is False.

(HistogramPattern value is between 1 and 255 and HistogramType property includes the exHistCumulative flag *)*

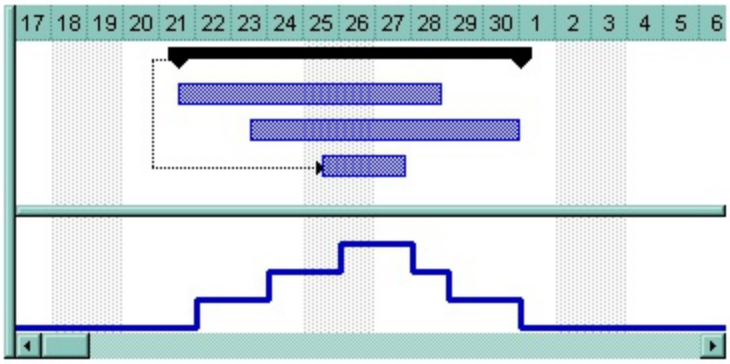
If the HistogramPattern property is a predefined value, the PatternEnum type specifies the shape of the histogram as shown bellow (exPatternDot, exPatternBox and exPatternEmpty):



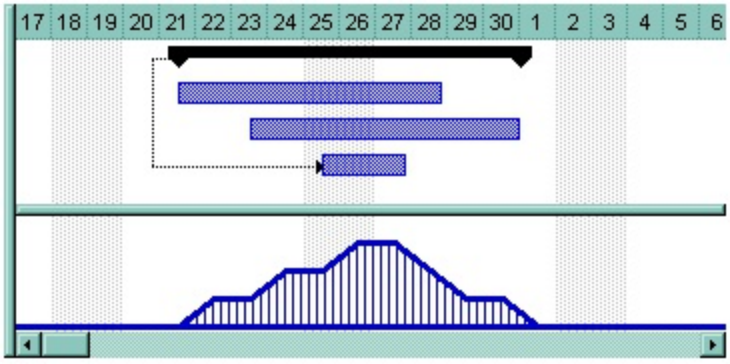




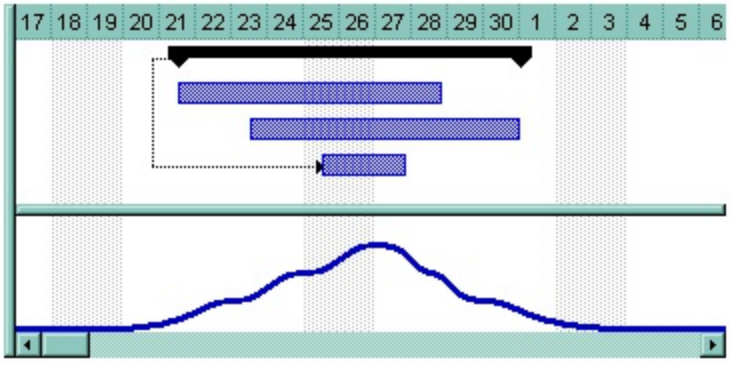
- If the HistogramPattern property is 256, the rectangular curves shows the overloads and subloads of the bar in the specified range of data as follows:



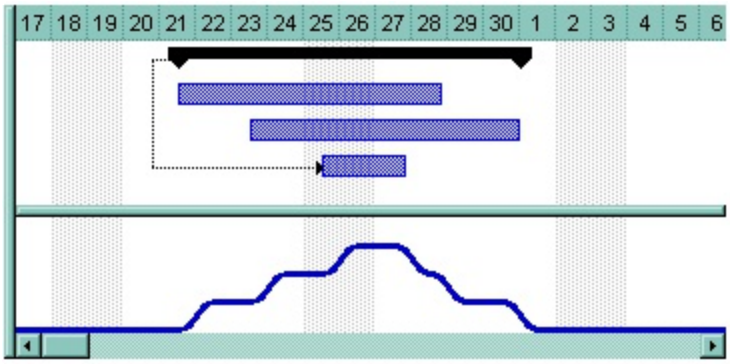
- If the HistogramPattern is 256 plus a predefined PatternEnum value, the histogram shows like follows (in this case the HistogramPattern property is 256 + exPatternVertical = 264):



- If the HistogramPattern property is 512, the bezier curves shows the overloads and subloads of the bar in the specified range of data as follows:

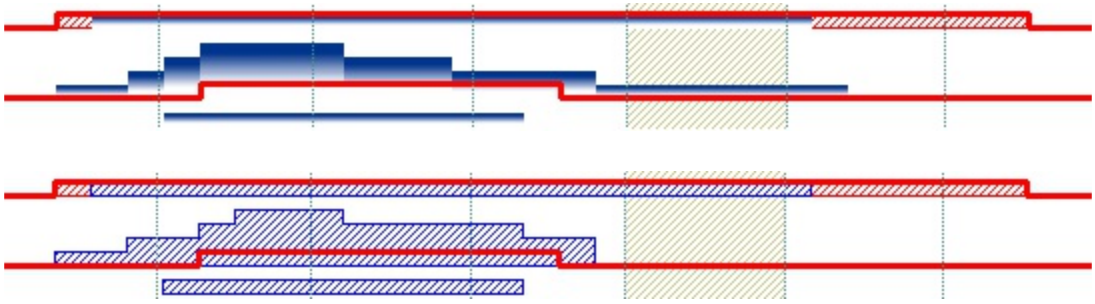


- If the HistogramPattern property is 1024, the round curves shows the overloads and subloads of the bar in the specified range of data as follows:



Bellow you can find how histogram is updated automatically as soon as the user moves or resizes a bar:

Use the HistogramVisible property to show or hide the histogram. Use the HistogramHeight property to specify at runtime the height of the histogram.



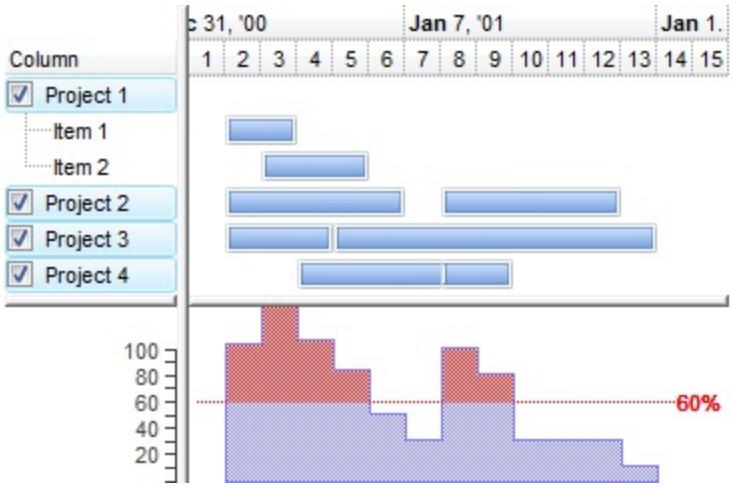
property Bar.HistogramRulerLinesColor as Color

Retrieves or sets a value that indicates the color to show the histogram's ruler lines.

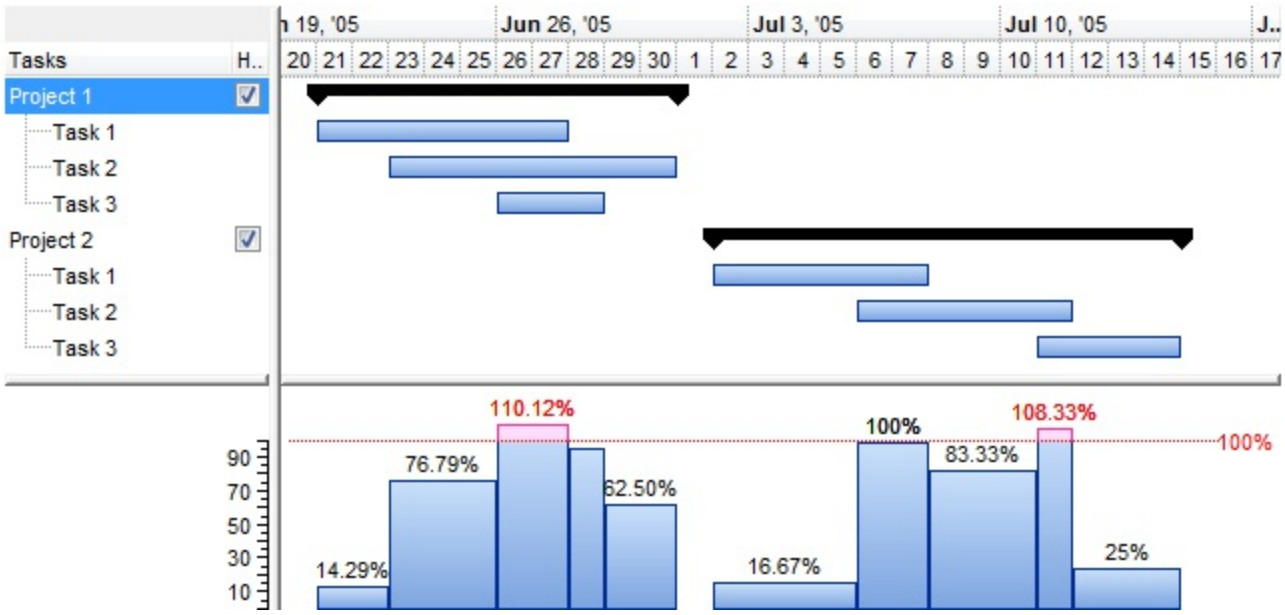
Type	Description
Color	A Color expression that specifies the color to show the ruler. Use the value on 1, incase you actually need a black ruler lines color.

By default, the HistogramRulerLinesColor property is 0, which means that it has no effect. The left side of the histogram displays rulers based on the HistogramType property. If the HistogramType property is exOverload, the ruler displays numbers, while if the HistogramType property is exOverAllocation displays percents. The HistogramGridLinesColor property to specify the color to show the grid lines in the right side of the histogram, when the HistogramType property is exOverload. Use the HistogramBoundsChanged event to notify your application when the left part of the histogram is resized, so inside controls must be re-positioned. You can always use the BeforeDrawPart/AfterDrawPart events to provide your custom drawing in the histogram. The ShowHistogramValues property specifies the formula that returns the color to display the selected values in the histogram for specified type of bar.

The left part of the histogram shows the rulers as in the following screen shot:



The left part of the histogram shows the rulers, while the right side of the histogram shows the values as in the following screen shot:


h 19, '05
Jun 26, '05
Jul 3, '05
Jul 10, '05
J..
Tasks
H..
20 21 22 23 24 25 26 27 28 29 30 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
Project 1
Task 1
Task 2
Task 3
Project 2
Task 1
Task 2
Task 3
110.12%
108.33%
100%
100%
90
70
50
30
10
76.79%
62.50%
83.33%
14.29%
16.67%
25%

property Bar.HistogramType as HistogramTypeEnum

Retrieves or sets a value that indicates the type of the histogram.

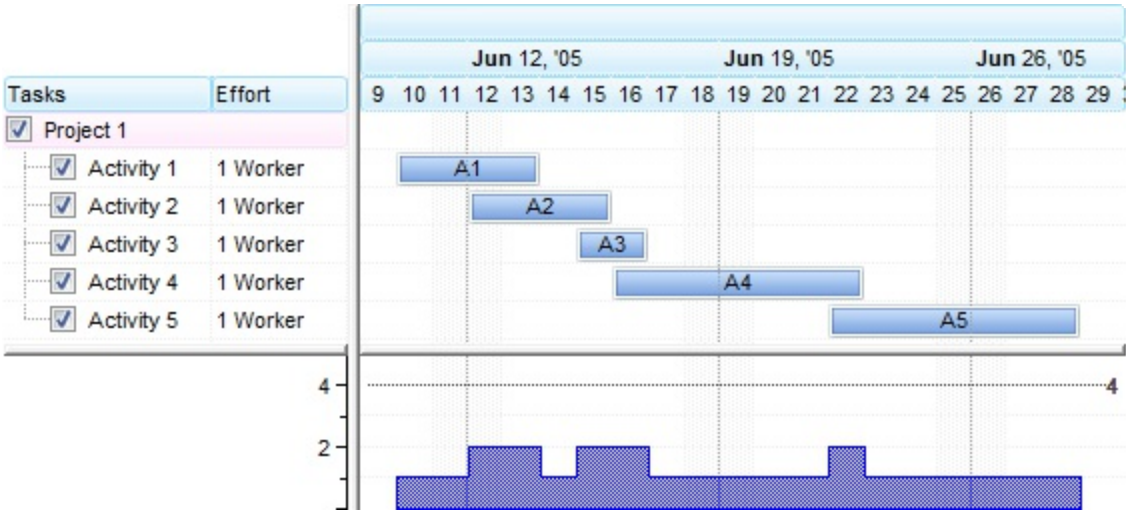
Type	Description
HistogramTypeEnum	A HistogramTypeEnum expression that specifies the type of the histogram-graph to be shown for this bar.

By default, the HistogramType property is exHistOverload. Use the HistogramType property to specify the histogram-graph to be displayed for a specified bar. **A bar is represented in the histogram only if HistogramPattern or HistogramColor property is set.** Use the HistogramColor property to define the *color or the EBN/skin file* of the pattern to be displayed in the histogram. Use the HistogramCriticalValue property to specify a critical value. The critical value is interpreted differently based on the HistogramType property. For instance, if the HistogramType property is exHistOverload, the critical value represents the count of cumulative bars since if the HistogramType property is exHistOverAllocation the critical value represents a percent value. Use the HistogramCriticalColor property to specify the color to display the critical values. Use HistogramRulerLinesColor property to specify the color to show the ruler in the left part of the histogram. The ShowHistogramValues property specifies the formula that returns the color to display the selected values in the histogram for specified type of bar.

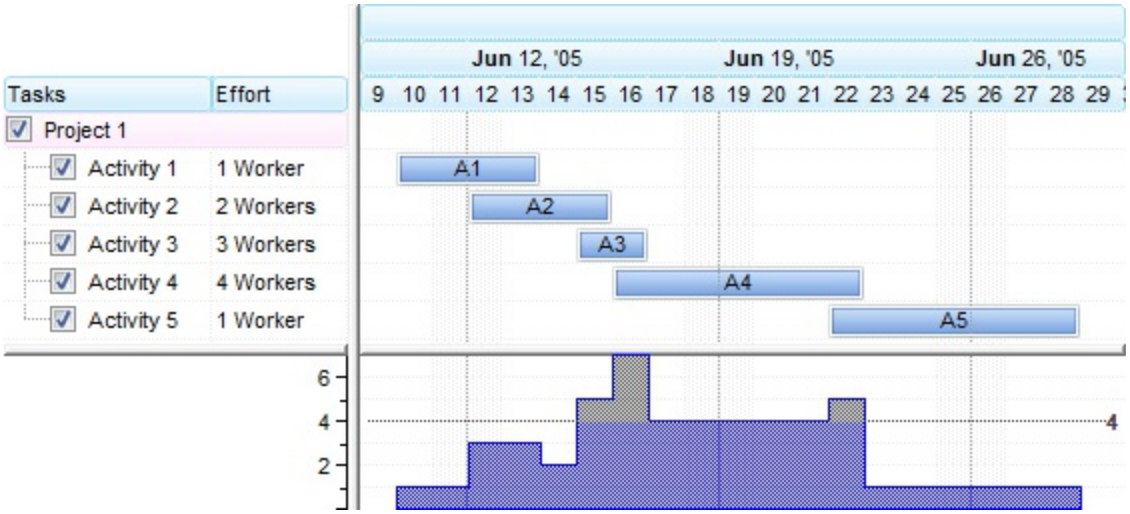
Currently the HistogramType property supports the following values:

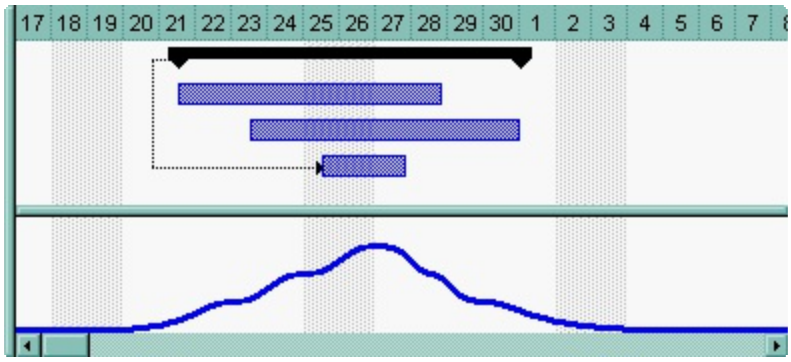
1. **exHistOverload** the histogram-graph shows the count of task day by day or unit by unit. The ItemBar(exBarEffort) specifies the number of units to count for the bar. For instance, if the bars display activities the exBarEffort value can represent the number of workers for each activity so the overload histogram displays the total number of workers on the activity. Use the HistogramGridLinesColor specifies the color to show the grid lines when the HistogramType property is exHistOverload. Use the HistogramItems property to specify the number of items allocated for the current bar to be shown in the histogram. The exHistOverload type can be combined with exHistCumulative.

   The following screen shot shows the exHistOverload histogram when exBarEffort property is 1 (by default):

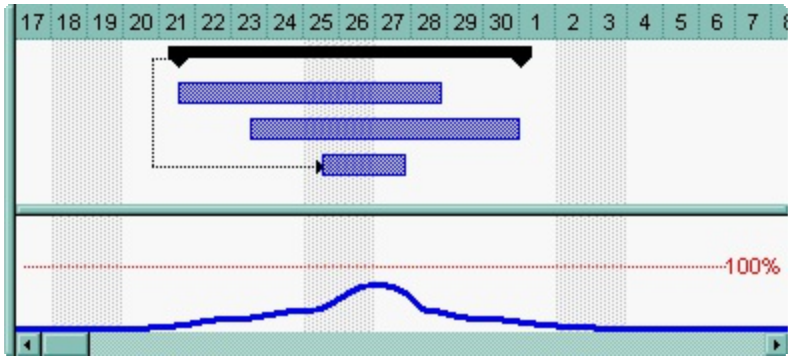


The following screen shot shows the exHistOverload histogram when exBarEffort property is different for bars as seen in the columns section:

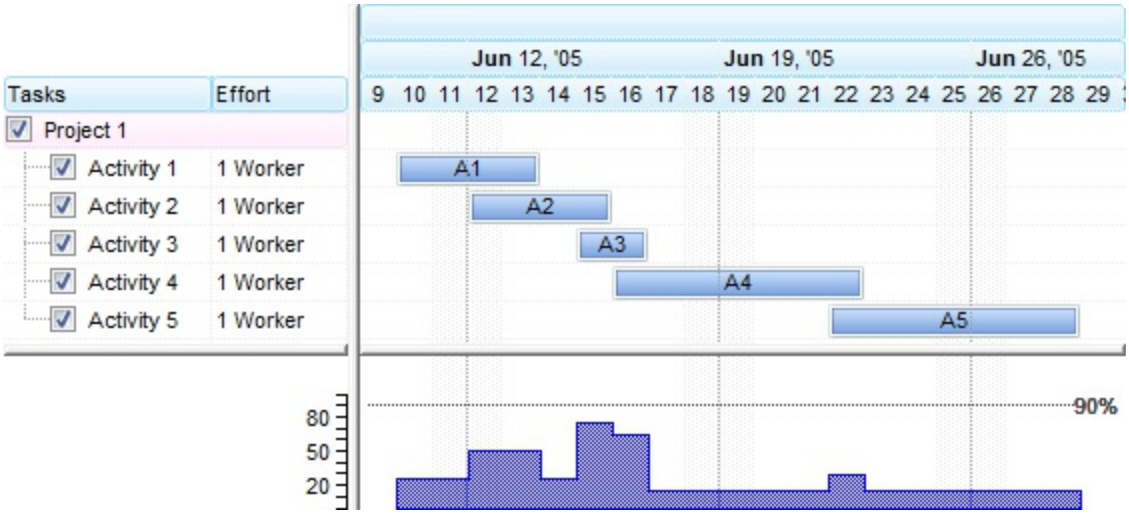




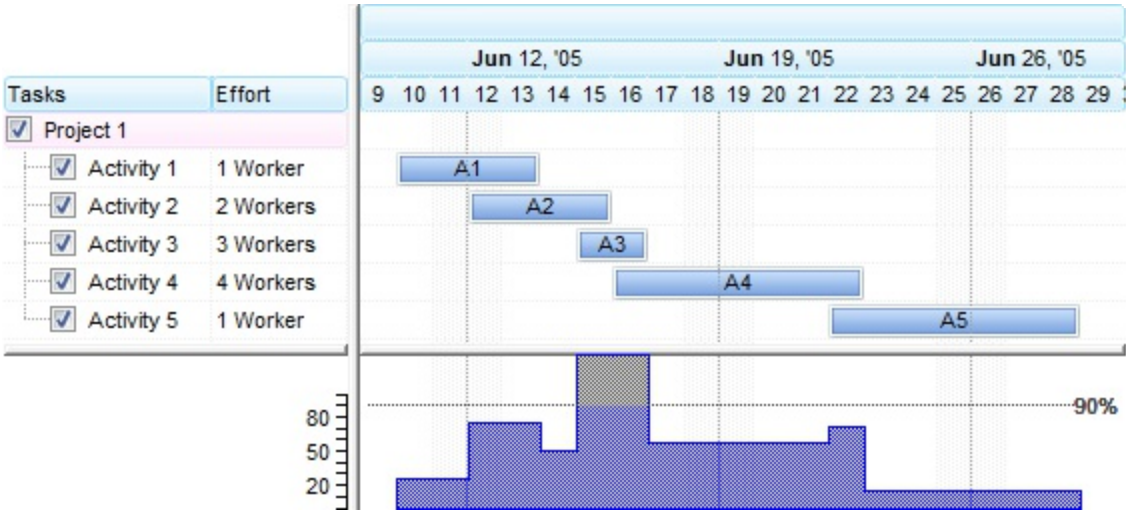
2. **exHistOverAllocation** the histogram shows the work-loads. The work-load for a task is computed as ItemBar(exBarEffort) / length of the bar. The work-load for the task is the work effort / task duration. (i.e. If item.exBarEffort = 1 and gantt bar length is 10 days, then the work-load = 0.1 or 10%). The histogram- graph shows the sum of the work-loads (the work-load of each task item is added, unit by unit). The exHistOverload type can be combined with exHistCumulative.



The following screen shot shows the exHistOverallocation histogram when exBarEffort property is 1 (by default):



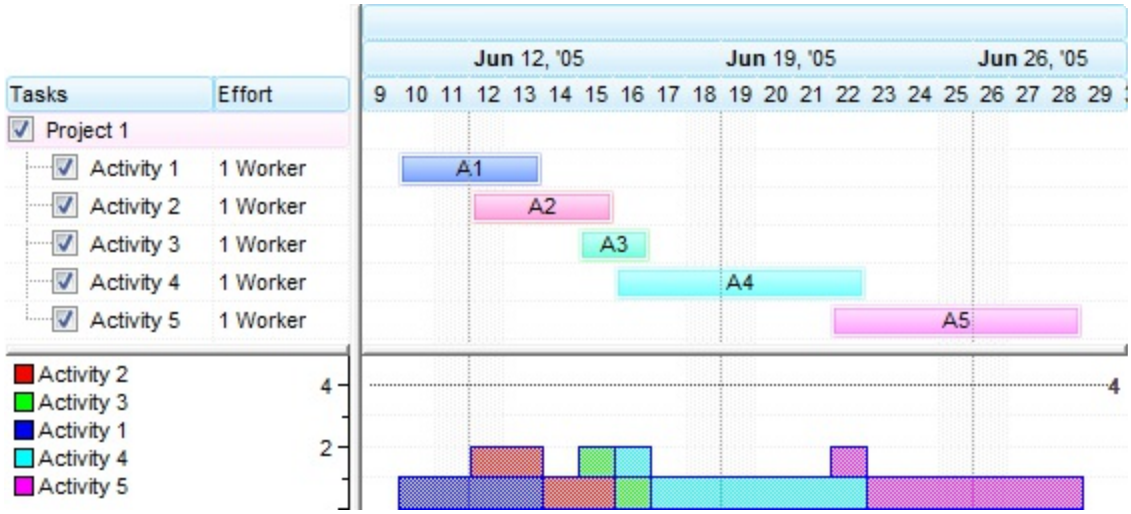
The following screen shot shows the exHistOverallocation histogram when exBarEffort property is different for bars as seen in the columns section:



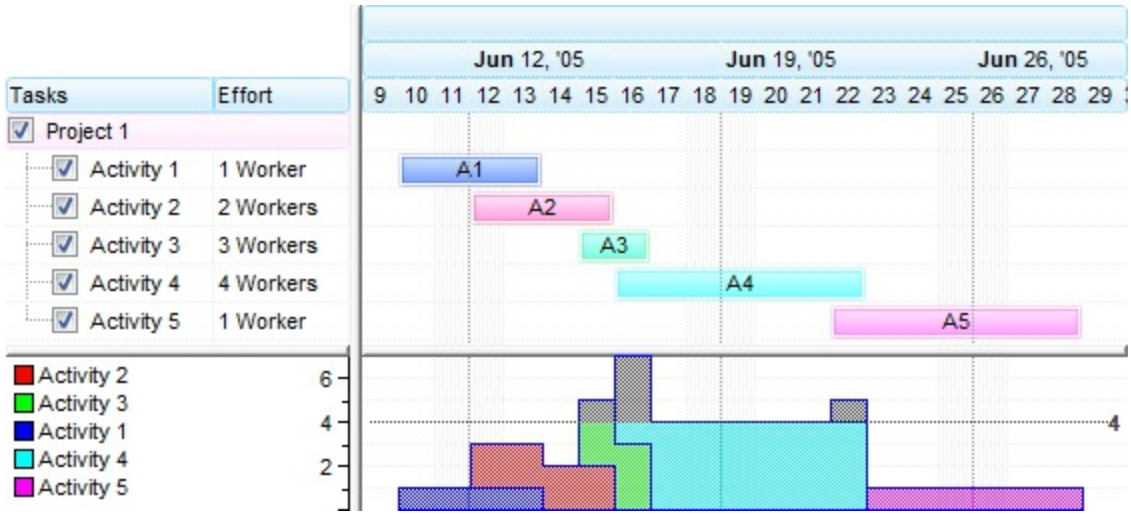
3. **exHistCumulative** the histogram shows the overloads or work-loads using cumulative values and different colors . This option should be combined with exHistOverload to show a cumulative overload histogram, or with the exHistOverAllocation to show a cumulative work loads histogram. The HistogramCumulativeColors property defines the number of colors being displayed in the cumulative histogram. The HistogramCumulativeColor property specifies a cumulative color based on its index.

Use the HistogramCumulativeShowLegend property to specify the index of the column being shown in the left side of the histogram to show the legend of the colors being used for cumulative bars. The HistogramPattern property should not be a curve, in order to show a cumulative histogram, in other words should be a predefined pattern. You can change the original color of the bars that generates the cumulative histogram using the HistogramCumulativeOriginalColorBars property. The following screen shot shows the bars using the original color in the items that generates the histogram (when HistogramCumulativeOriginalColorBars property is True, by default).

The following screen shot shows the exHistOverload + exHistCumulative histogram when exBarEffort property is 1 (by default), and the HistogramCumulativeOriginalColorBars property is False:



The following screen shot shows the exHistOverload + exHistCumulative histogram when exBarEffort property is different, , and the HistogramCumulativeOriginalColorBars property is False:



property Bar.Name as String

Retrieves the name of the bar.

Type	Description
String	A String expression that indicates the name of the Bar.

The Name property indicates the name of the bar. The Name property is read-only. Use the Add or Copy method to add a new bar to the Bars collection, using a different name. Use the AddBar method to add new bars to an item. Use the Shape, Pattern and Color properties to define the appearance for the middle part of the bar. Use the StartShape and StartColor properties to define the appearance for the starting part of the bar. Use the EndShape and EndColor properties to define the appearance for the ending part of the bar.

property Bar.Overlaid(Type as OverlaidBarsTypeEnum) as Variant

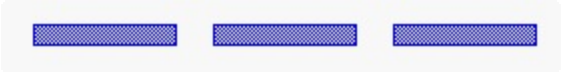
Retrieves or sets a value that indicates options for the specified overlaid type.

Type	Description
Type as OverlaidBarsTypeEnum	Specifies the Type of the overlaid being changed
Variant	A Variant expression that specifies the option for the current overlaid type.

Use the OverlaidType property to specify how the overlaid bars are displayed. Use the Overlaid property to indicate the parameter for specified overlaid type as follows:

- Overlaid(**exOverlaidBarsOffset**) (long expression) specifies the vertical offset, in pixels, to display the overlaid bars. By default, the Overlaid(exOverlaidBarsOffset) property is 3 pixels.

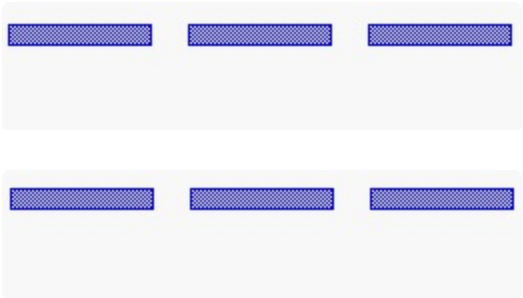
- Overlaid(**exOverlaidBarsIntersect**) (string expression) specifies the name of the bar to be displayed on the portion that laid over bars. By default, the Overlaid(exOverlaidBarsIntersect) property is empty, so nothing is displayed if the Overlaid is exOverlaidBarsIntersect. You MUST specify the name of the task to display the portion that covers the bars if the Overlaid is exOverlaidBarsIntersect. For instance, the following sample creates a copy of the task bar with a different color (red) and display it when two or more tasks covers.



```
With .Chart.Bars.Copy("Task", "RTask")
   .Color = RGB(255, 0, 0)
End With

With .Chart.Bars("Task")
   .OverlaidType = exOverlaidBarsIntersect
   .Overlaid(exOverlaidBarsIntersect) = "RTask"
End With
```

- Overlaid(**exOverlaidBarsStack**) (long expression) specifies the distance between

two bars that covers each other, in pixels. By default, the Overlaid(exOverlaidBarsStack) property is 3 pixels. Use the exOverlaidBarsStack | exOverlaidBarsStackAutoArrange flag to auto arrange the bars inside the item as they best fit. ***If the exOverlaidBarsStack flag is set, the control resizes the item so all bars fits entirely in the item, so we would recommend to set the ScrollBySingleLine property on True, as items with different heights may be displayed . The ItemHeight property specifies the height of the item. Use the ItemMaxHeight property to limit the height of the item when multiple bars covers each other.***



- Overlaid(**exOverlaidBarsTransparent**) (long expression) specifies the percent of transparency to be applied to bars that covers other bars. By default, the Overlaid(exOverlaidBarsTransparent) property is 50% (semi-transparent).

property Bar.OverlaidGroup as String

Specifies the list of bars beside the current bar that may cover each other.

Type	Description
String	A String expression that specifies the list of bars separated by , character that may cover each other.

By default, the OverlaidGroup property is empty, so only bars of the same type, in the same item are displayed using a different offset, transparency, when they cross each other. Use the OverlaidGroup property to specify the list of bars that may cross each other and display using different vertical offset, transparency and so on. The OverlaidType property to specify how the bars that cover each other are displayed. Use the Overlaid property to specify a different parameter for specified overlaid type. Use the OverlaidType and Overlaid properties to display different the bars that laid over or cover other bars.

The following VB sample, creates two new type of bars RTask and GTask from Task bar, and specify when they cross each other inside the item to be displayed using a different offset:

```
With G2antt1
   .DefaultItemHeight = 22
   .Columns.Add "Task"
   With .Chart
      .ResizeUnitScale = exHour
      .PaneWidth(False) = 48
      .FirstVisibleDate = #1/1/2001#
      With .Bars
         .Copy("Task","RTask").Color = 255
         .Copy("Task","GTask").Color = 65280
         With .Item("Task")
            .OverlaidType = exOverlaidBarsOffset or exOverlaidBarsTransparent
            .Overlaid(exOverlaidBarsTransparent) = 70
            .OverlaidGroup = "RTask,GTask"
         End With
      End With
   End With
   With .Items
      h = .AddItem("Task 1")
```

```
      .AddBar h,"Task",#1/4/2001#,#1/8/2001#,"A1"
      .AddBar h,"GTask",#1/7/2001#,#1/12/2001#,"A2"
      .AddBar h,"RTask",#1/10/2001#,#1/15/2001#,"A3"
   End With
End With
```

The following VB.NET sample, creates two new type of bars RTask and GTask from Task bar, and specify when they cross each other inside the item to be displayed using a different offset:

```
Dim h
With AxG2antt1
   .DefaultItemHeight = 22
   .Columns.Add "Task"
   With .Chart
      .ResizeUnitScale = EXG2ANTTLib.UnitEnum.exHour
      .PaneWidth(False) = 48
      .FirstVisibleDate = #1/1/2001#
      With .Bars
         .Copy("Task","RTask").Color = 255
         .Copy("Task","GTask").Color = 65280
         With .Item("Task")
            .OverlaidType = EXG2ANTTLib.OverlaidBarsTypeEnum.exOverlaidBarsOffset or
EXG2ANTTLib.OverlaidBarsTypeEnum.exOverlaidBarsTransparent
            .Overlaid(EXG2ANTTLib.OverlaidBarsTypeEnum.exOverlaidBarsTransparent) = 70
            .OverlaidGroup = "RTask,GTask"
         End With
      End With
   End With
   With .Items
      h = .AddItem("Task 1")
      .AddBar h,"Task",#1/4/2001#,#1/8/2001#,"A1"
      .AddBar h,"GTask",#1/7/2001#,#1/12/2001#,"A2"
      .AddBar h,"RTask",#1/10/2001#,#1/15/2001#,"A3"
   End With
End With
```

The following C# sample, creates two new type of bars RTask and GTask from Task bar,

and specify when they cross each other inside the item to be displayed using a different offset:

```
axG2antt1.DefaultItemHeight = 22;
axG2antt1.Columns.Add("Task");
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.ResizeUnitScale = EXG2ANTTLib.UnitEnum.exHour;
   var_Chart.set_PaneWidth(false,48);
   var_Chart.FirstVisibleDate = "1/1/2001";
   EXG2ANTTLib.Bars var_Bars = var_Chart.Bars;
      var_Bars.Copy("Task","RTask").Color = 255;
      var_Bars.Copy("Task","GTask").Color = 65280;
      EXG2ANTTLib.Bar var_Bar = var_Bars["Task"];
         var_Bar.OverlaidType = (EXG2ANTTLib.OverlaidBarsTypeEnum)
(EXG2ANTTLib.OverlaidBarsTypeEnum.exOverlaidBarsOffset +
EXG2ANTTLib.OverlaidBarsTypeEnum.exOverlaidBarsTransparent);

var_Bar.set_Overlaid(EXG2ANTTLib.OverlaidBarsTypeEnum.exOverlaidBarsTransparent,70);
         var_Bar.OverlaidGroup = "RTask,GTask";
EXG2ANTTLib.Items var_Items = axG2antt1.Items;
   int h = var_Items.AddItem("Task 1");
   var_Items.AddBar(h,"Task","1/4/2001","1/8/2001","A1",null);
   var_Items.AddBar(h,"GTask","1/7/2001","1/12/2001","A2",null);
   var_Items.AddBar(h,"RTask","1/10/2001","1/15/2001","A3",null);
```

The following C++ sample, creates two new type of bars RTask and GTask from Task bar, and specify when they cross each other inside the item to be displayed using a different offset:

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import "D:\\Exontrol\\ExG2antt\\project\\Debug\\ExG2antt.dll"
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
```

```
spG2antt1->PutDefaultItemHeight(22);
spG2antt1->GetColumns()->Add(L"Task");
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
    var_Chart->PutResizeUnitScale(EXG2ANTTLib::exHour);
    var_Chart->PutPaneWidth(VARIANT_FALSE,48);
    var_Chart->PutFirstVisibleDate("1/1/2001");
    EXG2ANTTLib::IBarsPtr var_Bars = var_Chart->GetBars();
        var_Bars->Copy(L"Task",L"RTask")->PutColor(255);
        var_Bars->Copy(L"Task",L"GTask")->PutColor(65280);
        EXG2ANTTLib::IBarPtr var_Bar = var_Bars->GetItem("Task");
            var_Bar->PutOverlaidType((EXG2ANTTLib::OverlaidBarsTypeEnum)
(EXG2ANTTLib::exOverlaidBarsOffset + EXG2ANTTLib::exOverlaidBarsTransparent));
            var_Bar->PutOverlaid(EXG2ANTTLib::exOverlaidBarsTransparent,long(70));
            var_Bar->PutOverlaidGroup(L"RTask,GTask");
EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems();
    long h = var_Items->AddItem("Task 1");
    var_Items->AddBar(h,"Task","1/4/2001","1/8/2001","A1",vtMissing);
    var_Items->AddBar(h,"GTask","1/7/2001","1/12/2001","A2",vtMissing);
    var_Items->AddBar(h,"RTask","1/10/2001","1/15/2001","A3",vtMissing);
```

The following VFP sample, creates two new type of bars RTask and GTask from Task bar, and specify when they cross each other inside the item to be displayed using a different offset:

```
with thisform.G2antt1
    .DefaultItemHeight = 22
    .Columns.Add("Task")
    with .Chart
        .ResizeUnitScale = 65536
        .PaneWidth(.F.) = 48
        .FirstVisibleDate = {^2001-1-1}
        with .Bars
            .Copy("Task","RTask").Color = 255
            .Copy("Task","GTask").Color = 65280
            with .Item("Task")
                .OverlaidType = 257
                .Overlaid(256) = 70
                .OverlaidGroup = "RTask,GTask"
```

```
        endwith
    endwith
  endwith
  with .Items
    h = .AddItem("Task 1")
    .AddBar(h,"Task",{^2001-1-4},{^2001-1-8},"A1")
    .AddBar(h,"GTask",{^2001-1-7},{^2001-1-12},"A2")
    .AddBar(h,"RTask",{^2001-1-10},{^2001-1-15},"A3")
  endwith
endwith
```

property Bar.OverlaidType as OverlaidBarsTypeEnum

Specifies how the overlay bars are shown.

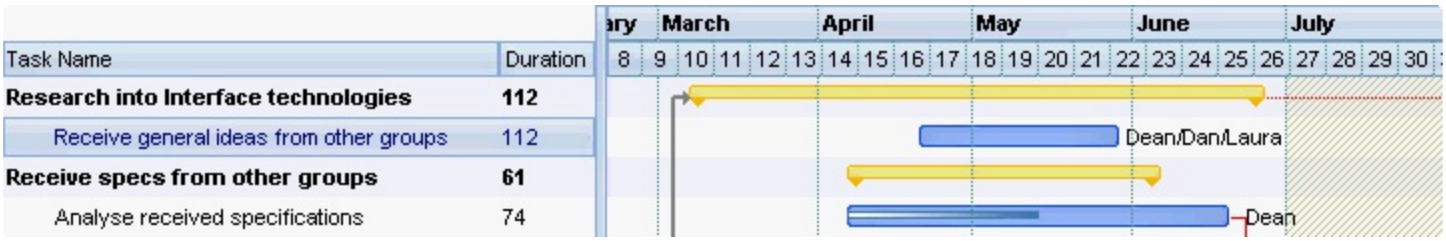
Type	Description
OverlaidBarsTypeEnum	A OverlaidBarsTypeEnum expression that specifies how overlaid bars are shown.

By default, the OverlaidType property is exOverlaidBarsNone. By default (exOverlaidBarsNone) the overlaid bars are not displayed, as shown in the second picture. For instance, if the OverlaidType property is exOverlaidBarsOffset and two task bars are cover each other, they get displayed using a different offset. The IntersectBars property determines if two bars intersects if returns 0. The ItemBar(exBarIntersectWith) property retrieves a collection of bars that interest with the current bar. Use the OverlaidOnMoving property to prevent overlaying the bars while the user moves or resizes the bar at runtime.

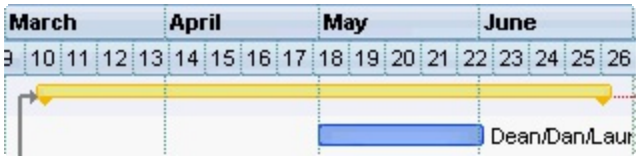
Use the *Overlaid...* properties when multiple bars are displayed in the same item and you want to distingue them when they covers each other. In order to get the overlay feature runs please follow the steps:

- Use the OverlaidType property to specify the type of overlaying when bars covers each other. By default, no overlay support.
- Use the Overlaid property to specify a different parameter for specified overlaid types. For instance, you can specify the vertical offset or transparency being used to display overlaid bars.
- Use the OverlaidGroup property to specify a list of bars (separated by , character) that may cover each other. By default, only bars of the same type in the same item, that covers each other are displayed using different vertical offsets, transparency, and so on. Instead, if the OverlaidGroup property contains another type of bars, they all together may cover each other using different vertical offsets, transparency and so on.

The following picture shows the bars arranged as a stack, when multiple bars cover each others bars: (OverlaidType = exOverlaidBarsStack Or exOverlaidBarsStackAutoArrange)



The following picture shows the bars using a different vertical offset and a different transparency, when multiple bars cover others bars: (OverlaidType = exOverlaidBarsOffset Or exOverlaidBarsTransparent)



By **default**, the following picture shows the bars at the same vertical offset, when multiple bars covers others bars: (OverlaidType = exOverlaidBarsNone)

The OverlaidType displays the bars that covers each other as follow:

- exOverlaidBarsOffset using *different vertical offset.* Use the Overlaid(exOverlaidBarsOffset) property to specify the offset being applied when the bars laid over. The bars gets arranged on maximum 3 layers, and the height of the item is not changed.

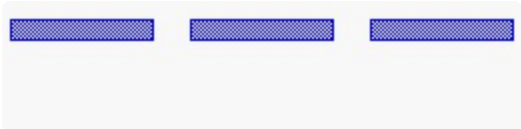
- exOverlaidBarsOffset or exOverlaidBarsTransparent using *different vertical offset and transparency.* The bars gets arranged on maximum 3 layers, and the height of the item is not changed.

- exOverlaidBarsIntersect using a *different bar for the portion that covers*. Use the Overlaid(exOverlaidBarsIntersect) property to specify the name of the bar that's displayed in the portion that's covered by two or multiple bars. For instance, you can add a copy of the current bar with a different color, and when two or multiple bars cover each other, the portion that is laid over can be displayed using your clone bar with a different color. The bars gets arranged on 1 layer only.

- exOverlaidBarsStack *arranges the bars as a stack* as soon as they cover each other. ***If this flag is set, the control resizes the item so all bars fits entirely in the item, so we would recommend to set the ScrollBySingleLine property on True, as items with different heights may be displayed . The ItemHeight property specifies the height of the item. Use the ItemMaxHeight property to limit the height of the item when multiple bars covers each other.*** The Overlaid(exOverlaidBarsStack) specifies the distance in pixels between two bars that

covers each other. The Overlaid(exOverlaidBarsTransparent) specifies the percent of transparency being applied to bars in the same item that are not moved or resized. The bars may be arranged on multiple layers, and the height of the item is being changed.

- exOverlaidBarsStack or exOverlaidBarsStackAutoArrange *arranges the bars as a stack, and auto arrange them to best fit in the item.* The bars may be arranged on multiple layers, and the height of the item is being changed.



property Bar.OverviewColor as Color

Retrieves or sets a value that indicates the color to show the bars of this type in the control's overview panel.

Type	Description
Color	A Color expression that indicates the color to show the bars of this type in the control's overview panel.

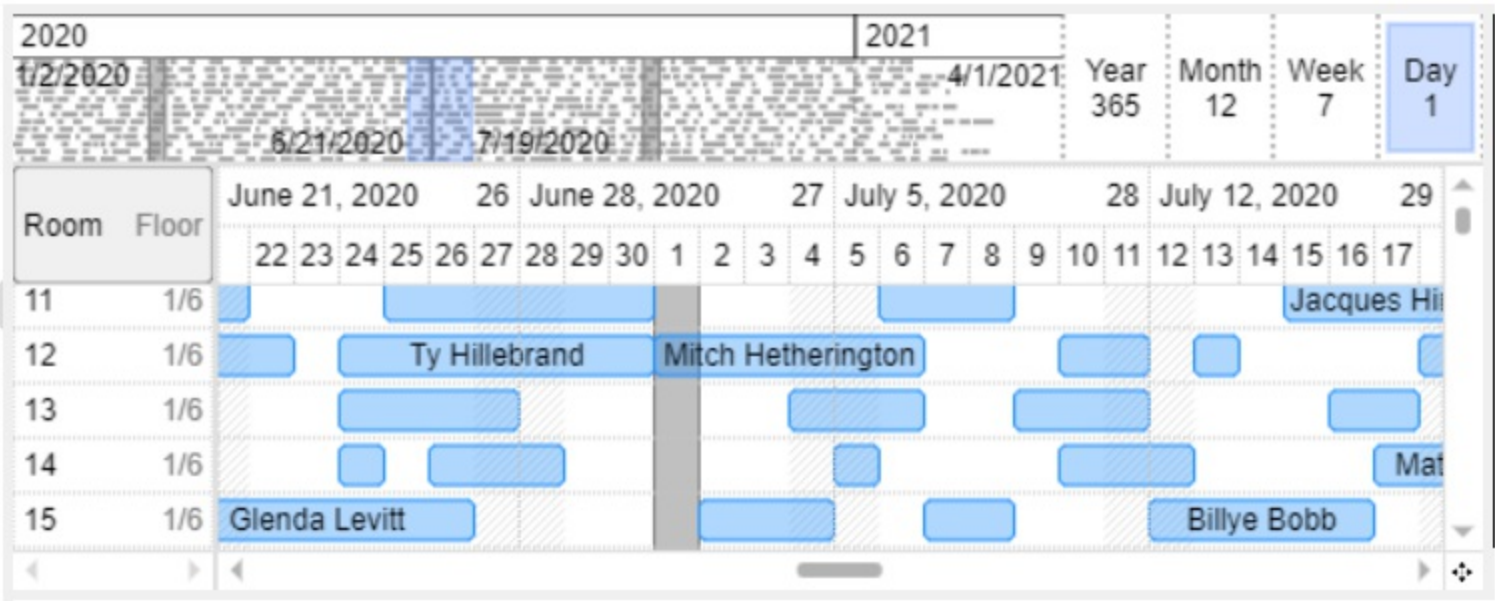
By default, the OverviewColor property is 0, which indicates that it has no effect. The OverviewColor property indicates the color to show the bars of this type in the control's overview panel. The OverviewColor property on -1, hides the overview-representation of bars of this type (hides the bars in the overview).

The color to specify the bar in the overview area is determined as follows:

- If ItemBar(exBarOverviewColor) property is not 0, the exBarOverviewColor indicates the color to show the bar in the overview area, else
- If OverviewColor property is not 0, the OverviewColor property indicates the color to show the bar in the overview area, else
- If the ItemBar(exBarColor) is not 0, the exBarColor indicates the color to show the bar in the overview area, else
- The Color property of the Bar indicates the color to show the bar in the overview part of the control.

(The bar is represented into the control's overview only if its determined color is not -1)

The following screen shot shows the control's overview (while bar are displayed in blue, the overview shows the bar in light-gray):



property Bar.Pattern as PatternEnum

Retrieves or sets a value that indicates the pattern being used to fill the bar.

Type	Description
PatternEnum	A PatternEnum expression that indicates the brush being used to fill the bar.

Use the Pattern property to specify the brush to fill the bar. By default, the Pattern property is exPatternSolid. Use the Color property to specify the color to fill the bar. Use the Shape property to specify the height and the vertical alignment of the middle part of the bar. Use the StartColor property to specify the color for the beginning part of the bar, if the StartShape property is not exShapeIconEmpty. Use the EndColor property to specify the color for the ending part of the bar, if the EndShape property is not exShapeIconEmpty. If the Pattern property is exPatternBox, the StartColor and EndColor properties defines the start and ending color to show a gradient bar. If available, the ItemBar(exBarPattern) property may be used to specify a different pattern for a specified bar only.

The following VB sample creates a new bar called "Task2", that's similar with the "Task" bar excepts that we change the pattern to fill the bar:

```
With G2antt1.Chart.Bars
   With .Copy("Task", "Task2")
      .Pattern = exPatternDot
   End With
End With
```

The following C++ sample creates a new bar called "Task2", that's similar with the "Task" bar excepts that we change the pattern to fill the bar:

```
CBars bars = m_g2antt.GetChart().GetBars();
CBar bar = bars.Copy( "Task", "Task2" );
bar.SetPattern( 2 /*exPatternDot*/ );
```

The following VB.NET sample creates a new bar called "Task2", that's similar with the "Task" bar excepts that we change the pattern to fill the bar:

```
With AxG2antt1.Chart.Bars
   With .Copy("Task", "Task2")
      .Pattern = EXG2ANTTLib.PatternEnum.exPatternDot
   End With
End With
```

The following C# sample creates a new bar called "Task2", that's similar with the "Task" bar excepts that we change the pattern to fill the bar:

```
EXG2ANTTLib.Bar bar = axG2antt1.Chart.Bars.Copy("Task", "Task2");
bar.Pattern = EXG2ANTTLib.PatternEnum.exPatternDot;
```

The following VFP sample creates a new bar called "Task2", that's similar with the "Task" bar excepts that we change the pattern to fill the bar:

```
with thisform.G2antt1.Chart.Bars
   with .Copy("Task", "Task2" )
      .Pattern = 2
   endwith
endwith
```

property Bar.Shape as ShapeBarEnum

Retrieves or sets a value that indicates the shape of the bar.

Type	Description
ShapeBarEnum	A ShapeBarEnum expression that indicates the height and the vertical alignment of the bar

Use the Shape property to specify the height and the vertical alignment of the middle part of the bar. By default, the Shape property is exShapeSolid. Use the Pattern property to specify the brush to fill the bar. Use the Color property to specify the color to fill the bar. Use the StartColor property to specify the color for the beginning part of the bar, if the StartShape property is not exShapeIconEmpty. Use the EndColor property to specify the color for the ending part of the bar, if the EndShape property is not exShapeIconEmpty.

The following VB sample creates a new bar called "Task2", that's similar with the "Task" bar excepts that we change the shape of the new bar bar:

```
With G2antt1.Chart.Bars
   With .Copy("Task", "Task2")
      .Shape = exShapeSolidCenter
   End With
End With
```

The following C++ sample creates a new bar called "Task2", that's similar with the "Task" bar excepts that we change the shape of the new bar bar:

```
CBars bars = m_g2antt.GetChart().GetBars();
CBar bar = bars.Copy( "Task", "Task2" );
bar.SetShape( 3 /*exShapeSolidCenter*/ );
```

The following VB.NET sample creates a new bar called "Task2", that's similar with the "Task" bar excepts that we change the shape of the new bar bar:

```
With AxG2antt1.Chart.Bars
   With .Copy("Task", "Task2")
      .Shape = EXG2ANTTLib.ShapeBarEnum.exShapeSolidCenter
   End With
End With
```

The following C# sample creates a new bar called "Task2", that's similar with the "Task" bar excepts that we change the shape of the new bar bar:

```
EXG2ANTTLib.Bar bar = axG2antt1.Chart.Bars.Copy("Task", "Task2");
bar.Shape = EXG2ANTTLib.ShapeBarEnum.exShapeSolidCenter;
```

The following VFP sample creates a new bar called "Task2", that's similar with the "Task" bar excepts that we change the shape of the new bar bar:

```
with thisform.G2antt1.Chart.Bars
   with .Copy("Task", "Task2" )
      .Shape = 3
   endwith
endwith
```

property Bar.Shortcut as String

Specifies a value that indicates a shortcut for the current bar.

Type	Description
String	A String expression that indicates the shortcut of the bar

The Shortcut property adds a shortcut to this bar. Use the Add method to add new type of bars to the chart. Use the Shortcut property to redefine a known bar. For instance, you can define the bar "Task%Progress:Split", and rename it to "Task", and so all Task bars will be divided by the nonworking area, and may display percent values, in other words, you redefined the Task bars.

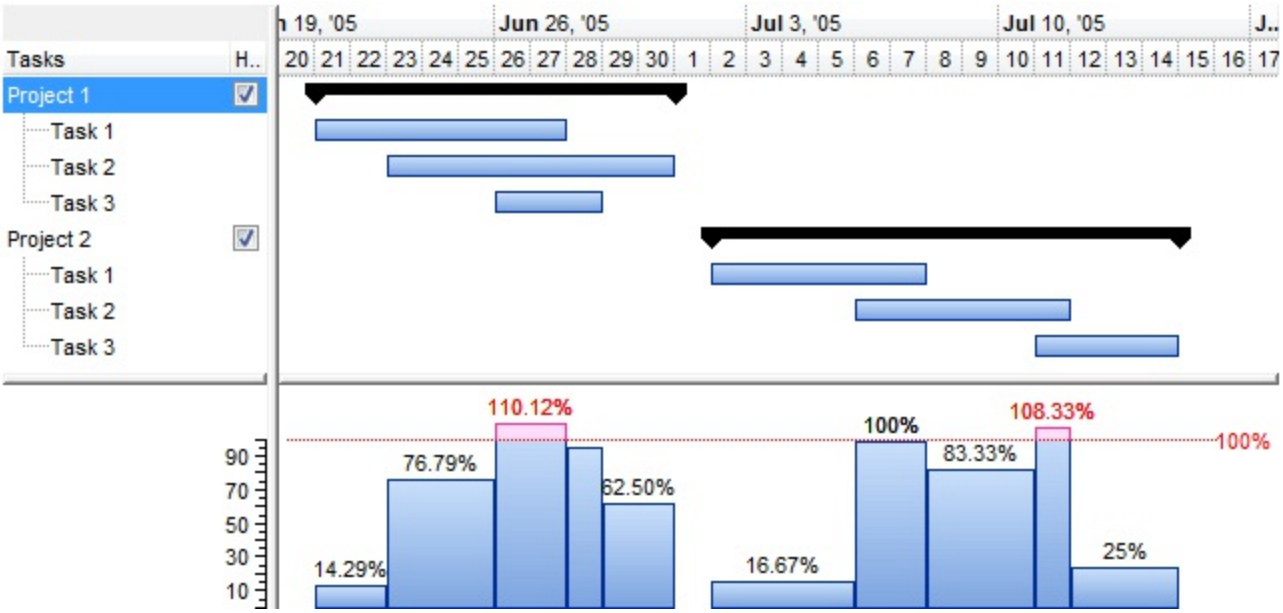
property Bar.ShowHistogramValues as String

Specifies the formula that returns the color to display the selected values in the histogram for specified type of bar.

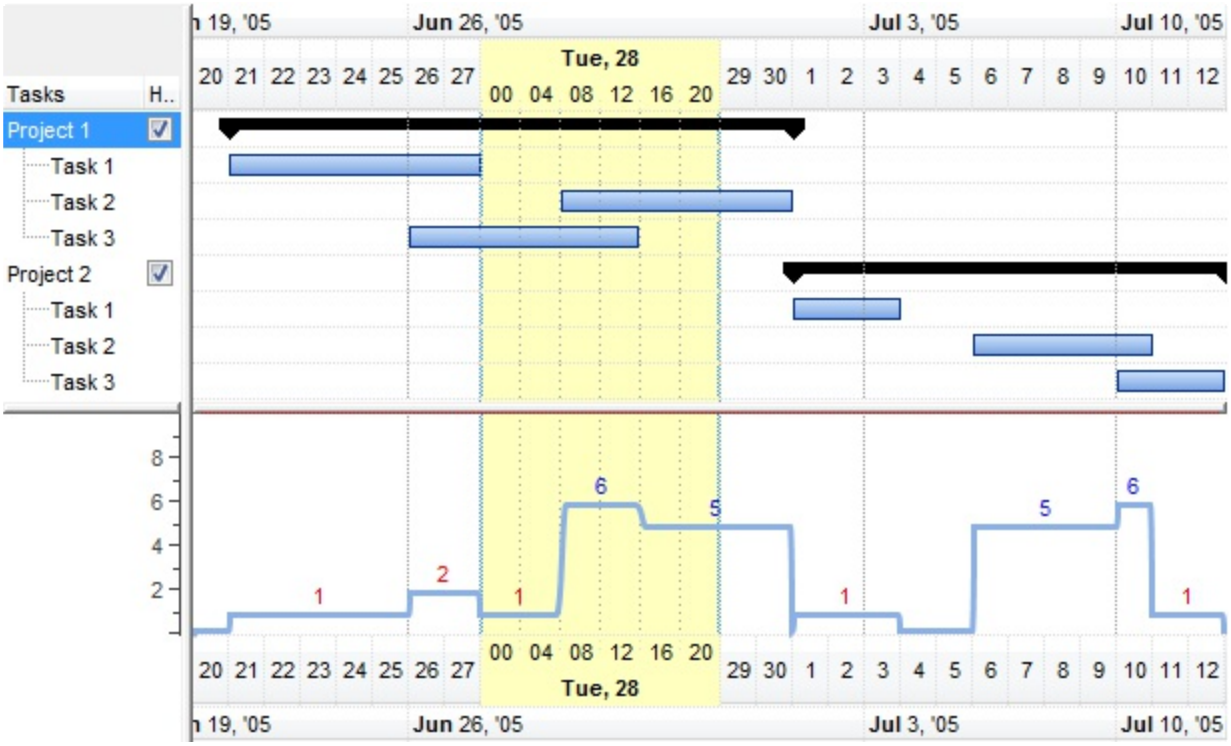
Type	Description
String	A String expression that specifies the color to display the values in the histogram.

By default, the ShowHistogramValues property is empty. If the ShowHistogramValues property is empty or not valid, the values are not shows in the histogram. The easiest way to show the values in the histogram is just using ShowHistogramValues property on "1". The values in the histogram show a % character if the HistogramType property is exHistOverAllocation. The sum of the exBarEffort value of each bar indicates the value being shown in the histogram. The **value** keyword in ShowHistogramValues property indicates the value in the histogram. For instance, "value>5?255:0" displays values greater than 5 with the color 255 (red in RGB format). The "16711680" specifies the color to be shown the values in the histogram (blue in RGB format is RGB(0,0,255) in other words it is 16711680). The HistogramRulerLinesColor property indicates the color to show the rulers in the left side of the histogram. The HistogramValueFromPoint property gets the value in the histogram from the specified location.

The following screen shows values in a exHistOverLoad histogram (ShowHistogramValues property is "value>100?255:1"):



The following screen shows values in a exHistOverAllocation histogram (ShowHistogramValues property is "value>=5?16711680:255"):



The **value** keyword in ShowHistogramValues property indicates the value in the histogram

This property/method supports predefined constants and operators/functions as described here.

property Bar.StartColor as Color

Returns or sets a value that indicates the color for the left side corner.

Type	Description
Color	A Color expression that indicates the color for the starting part of the bar.

Use the StartColor property to specify the color to fill the start part of the bar, if the StartShape property is not exShapeIconEmpty or Pattern is exPatternBox. Use the Color property to specify the color to fill the middle part of the bar. Use the EndColor and EndShape properties to define the appearance of the starting part of the bar. Use the AddShapeCorner property to add custom icons to the bars. In this case, the icon is processed before displaying based on the StartColor/ EndColor property. For instance, if you add an black and white icon, and the StartColor/EndColor is red, the icon will be painted in red. Instead, if the StartColor/EndColor property is -1 (0xFFFFFFFF, not white which is 0x00FFFFFF), the icon is painted as it was added using the AddShapeCorner without any image processing. If the Pattern property is exPatternBox, the StartColor and EndColor properties defines the start and ending color to show a gradient bar.

The following VB sample changes the "Task" bar visual appearance using liner gradient with margins as shown :

```
With G2antt1.Chart.Bars.Item("Task")
   .Color = vbWhite
   .Pattern = exPatternBox
   .StartShape = exShapeIconCircleDot
   .StartColor = vbRed
   .EndShape = exShapeIconCircleDot
   .EndColor = vbBlue
End With
```

The following VB sample changes the "Task" bar visual appearance using liner gradient with solid border as shown :

```
With G2antt1.Chart.Bars.Item("Task")
   .Color = vbRed
   .Pattern = exPatternBox
   .StartColor = vbRed
   .EndColor = vbBlue
End With
```

The following VB sample defines a new bar that looks like this  :

```
With G2antt1.Chart.Bars.Add("Task2")
    .Pattern = exPatternShadow
    .Color = RGB(0, 0, 255)
    .StartShape = exShapeIconCircleDot
    .StartColor = RGB(255, 0, 0)
End With
```

The following C++ sample defines a bar that looks like this above:

```
CBar bar = m_g2antt.GetChart().GetBars().Add("Task2");
bar.SetPattern( 3 /*exPatternShadow*/ );
bar.SetColor( RGB(0, 0, 255) );
bar.SetStartShape( 4 /* exShapeIconCircleDot*/ );
bar.SetStartColor( RGB(255, 0, 0) );
```

The following VB.NET sample defines a bar that looks like this above:

```
With AxG2antt1.Chart.Bars.Add("Task2")
    .Pattern = EXG2ANTTLib.PatternEnum.exPatternShadow
    .Color = RGB(0, 0, 255)
    .StartShape = EXG2ANTTLib.ShapeCornerEnum.exShapeIconCircleDot
    .StartColor = RGB(255, 0, 0)
End With
```

The following C# sample defines a bar that looks like this above:

```
With AxG2antt1.Chart.Bars.Add("Task2")
    .Pattern = EXG2ANTTLib.PatternEnum.exPatternShadow
    .Color = RGB(0, 0, 255)
    .StartShape = EXG2ANTTLib.ShapeCornerEnum.exShapeIconCircleDot
    .StartColor = RGB(255, 0, 0)
End With
```

The following VFP sample defines a bar that looks like this above:

```
with thisform.G2antt1.Chart.Bars.Add("Task2")
    .Pattern = 3 && exPatternShadow
```

```
    .Color = RGB(0, 0, 255)
    .StartShape = 4 && exShapeIconCircleDot
    .StartColor = RGB(255, 0, 0)
EndWith
```

In VB.NET or C# you require the following functions until the .NET framework will provide:

You can use the following VB.NET function:

```
    Shared Function ToUInt32(ByVal c As Color) As UInt32
        Dim i As Long
    i = c.R
    i = i + 256 * c.G
    i = i + 256 * 256 * c.B
    ToUInt32 = Convert.ToUInt32(i)
    End Function
```

You can use the following C# function:

```
private UInt32 ToUInt32(Color c)
{
    long i;
    i = c.R;
    i = i + 256 * c.G;
    i = i + 256 * 256 * c.B;
    return Convert.ToUInt32(i);
}
```

property Bar.StartShape as ShapeCornerEnum

Retrieves or sets a value that indicates the shape of the left side corner.

Type	Description
ShapeCornerEnum	A ShapeCornerEnum expression that defines the shape of the icon being used to draw the corner.

By default, the StartShape property is exShapeIconEmpty. If the StartShape property is exShapeIconEmpty the bas has no starting part. Use the Color property to specify the color to fill the middle part of the bar. Use the Pattern property to specify the brush being used to fill the bar. Use the Shape property to specify the height and the vertical alignment of the middle part of the bar. Use the AddShapeCorner method to add a custom icon to be used as a starting or ending part of the bar. Use the Images or ReplaceIcon method to update the list of control's icons.

The following VB sample adds a custom shape and defines a bar like this :

```
With G2antt1.Chart.Bars
   .AddShapeCorner 12345, 1
   With .Add("Task2")
      .Pattern = exPatternDot
      .Shape = exShapeThinDown
      .StartShape = 12345
      .StartColor = RGB(255, 0, 0)
      .Color = .StartColor
   End With
End With
```

The following C++ sample adds a custom shape and defines a bar like above:

```
CBars bars = m_g2antt.GetChart().GetBars();
bars.AddShapeCorner( COleVariant( (long)12345 ), COleVariant( (long)1 ) );
CBar bar = bars.Add("Task2");
bar.SetPattern( 2 /*exPatternDot*/ );
bar.SetShape( 20 /*exShapeThinDown*/ );
bar.SetStartShape( 12345 );
bar.SetStartColor( RGB(255, 0, 0) );
bar.SetColor( bar.GetStartColor() );
```

The following VB.NET sample adds a custom shape and defines a bar like above:

```
With AxG2antt1.Chart.Bars
   .AddShapeCorner(12345, 1)
   With .Add("Task2")
      .Pattern = EXG2ANTTLib.PatternEnum.exPatternDot
      .Shape = EXG2ANTTLib.ShapeBarEnum.exShapeThinDown
      .StartShape = 12345
      .StartColor = RGB(255, 0, 0)
      .Color = .StartColor
   End With
End With
```

The following VB.NET sample adds a custom icon to the start of all Task bars:

```
With AxG2antt1.Chart.Bars
   .AddShapeCorner(12345, 1)
   .Item("Task").StartShape = 12345
   .Item("Task").StartColor = UInteger.MaxValue
End With
```

The following C# sample adds a custom shape and defines a bar like above:

```
axG2antt1.Chart.Bars.AddShapeCorner(12345, 1);
EXG2ANTTLib.Bar bar = axG2antt1.Chart.Bars.Add("Task2");
bar.Pattern = EXG2ANTTLib.PatternEnum.exPatternDot;
bar.Shape = EXG2ANTTLib.ShapeBarEnum.exShapeThinDown;
bar.StartShape = (EXG2ANTTLib.ShapeCornerEnum)12345;
bar.StartColor = ToUInt32(Color.FromArgb(255, 0, 0));
bar.Color = bar.StartColor;
```

The following C# sample adds a custom icon to the start of all Task bars:

```
EXG2ANTTLib.Bars bars = axG2antt1.Chart.Bars;
bars.AddShapeCorner(12345, 1);
bars["Task"].StartShape = EXG2ANTTLib.ShapeCornerEnum.exShapeIconEmpty + 12345;
bars["Task"].StartColor = 0xFFFFFFFF;
```

The following VFP sample adds a custom shape and defines a bar like above:

```
With thisform.G2antt1.Chart.Bars
```

```
    .AddShapeCorner(12345, 1)
    With .Add("Task2")
        .Pattern = 2 && exPatternDot
        .Shape = 20 && exShapeThinDown
        .StartShape = 12345
        .StartColor = RGB(255, 0, 0)
        .Color = .StartColor
    EndWith
EndWith
```

The following VB sample defines a new bar that looks like this  :

```
With G2antt1.Chart.Bars.Add("Task2")
    .Pattern = exPatternShadow
    .Color = RGB(0, 0, 255)
    .StartShape = exShapeIconCircleDot
    .StartColor = RGB(255, 0, 0)
End With
```

The following C++ sample defines a bar that looks like this above:

```
CBar bar = m_g2antt.GetChart().GetBars().Add("Task2");
bar.SetPattern( 3 /*exPatternShadow*/ );
bar.SetColor( RGB(0, 0, 255) );
bar.SetStartShape( 4 /* exShapeIconCircleDot*/ );
bar.SetStartColor( RGB(255, 0, 0) );
```

The following VB.NET sample defines a bar that looks like this above:

```
With AxG2antt1.Chart.Bars.Add("Task2")
    .Pattern = EXG2ANTTLib.PatternEnum.exPatternShadow
    .Color = RGB(0, 0, 255)
    .StartShape = EXG2ANTTLib.ShapeCornerEnum.exShapeIconCircleDot
    .StartColor = RGB(255, 0, 0)
End With
```

The following C# sample defines a bar that looks like this above:

```
With AxG2antt1.Chart.Bars.Add("Task2")
```

```
    .Pattern = EXG2ANTTLib.PatternEnum.exPatternShadow
    .Color = RGB(0, 0, 255)
    .StartShape = EXG2ANTTLib.ShapeCornerEnum.exShapeIconCircleDot
    .StartColor = RGB(255, 0, 0)
End With
```

The following VFP sample defines a bar that looks like this above:

```
with thisform.G2antt1.Chart.Bars.Add("Task2")
    .Pattern = 3 && exPatternShadow
    .Color = RGB(0, 0, 255)
    .StartShape = 4 && exShapeIconCircleDot
    .StartColor = RGB(255, 0, 0)
EndWith
```

Bars object

The Bars holds a collection of Bar objects. A Bar object defines the look and feel for bars in the chart's area. Use the Bars property to access the Bars collection. Use the Chart object property to access the control's chart. Use the AddBar method to add a bar to an item. The Bars collection holds a collection of predefined and custom bars. The Bars object supports the following methods and properties:

Name	Description
Add	Adds a Bar object to the collection and returns a reference to the newly created object.
AddShapeCorner	Adds a custom shape corner.
Clear	Removes all objects in a collection.
Copy	Copies a Bar object and returns a reference to the newly created object.
Count	Returns the number of objects in a collection.
Item	Returns a specific Column of the Columns collection.
Remove	Removes a specific member from the Bars collection.
RemoveShapeCorner	Removes a custom shape corner.

method Bars.Add (Name as String)

Adds a Bar object to the collection and returns a reference to the newly created object.

Type	Description
Name as String	A String expression that indicates the name of the bar being created. *If the Name parameter includes the ":" or "%"character, it has a special meaning described bellow.*

Return	Description
Bar	A Bar object being inserted.

The Add method adds a new bar to the Bars collection. The look and feel of the newly created bar could depend on the Name parameter like follows:

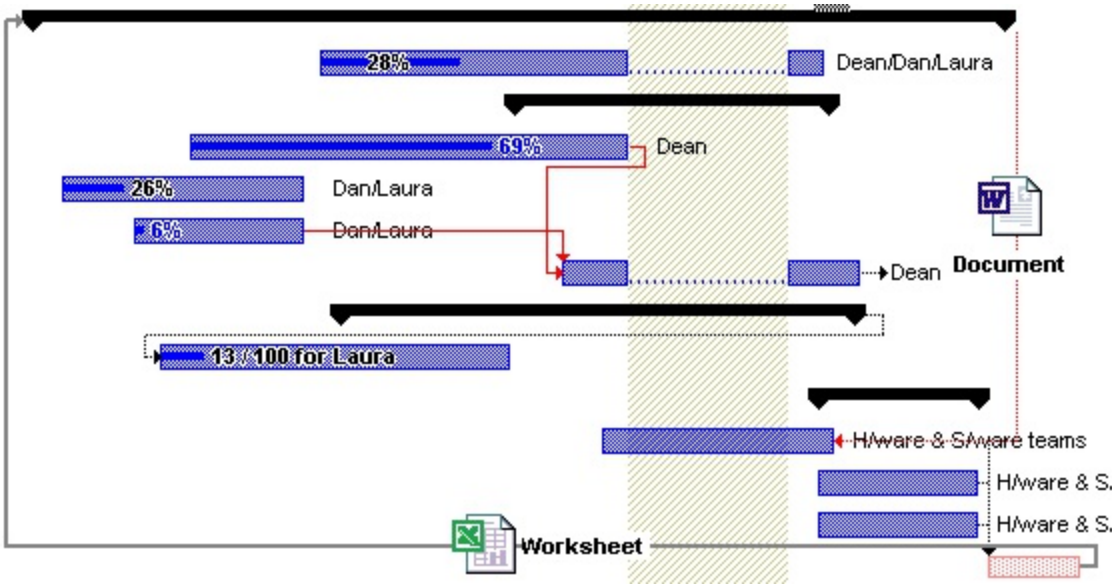
1.  *If the Name parameter doesn't include a **:** or **%** character the Add method adds a regular bar.*
2. 

*If the Name parameter includes a **%** character, so the Name parameter is like **A%B**, the Add method adds a new bar that's a combination of two existing bars A and B so the first bar A is displayed on the full area of the bar, since the second bar B uses the ItemBar(,,exBarPercent) value to determine the percent of the area from the full bar to be painted. Use the ItemBar(,,exBarShowPercentCaption)/ItemBar(,,exBarPercentCaptionFormat) to show and format the percent value as text. Use the ItemBar(,,exBarCanResizePercent) to disable resizing the percent at runtime. For instance, the Add("Task%Progress") adds a combination of Task*  *and Progress* *bars, so the Task shape is displayed on the full bar, and the Progress shape is displayed only on the portion determined by the Items.ItemBar(,,exBarPercent) value. The A and B could be any known bar at the adding time. For instance, if you have added bars like "MyTask" and "MySplit" you can define the bar "MyTask%MySplit", and so on. This option helps you to display proportionally the second shape when the user resizes or moves the bar.*
3.  *If the Name parameter includes a **:** character, so the Name parameter is like **A:B**, the newly created bar indicates a combination of A and B bars, where A is displayed in the working areas, since the B bar is displayed in non-working areas. Use the NonworkingDays, NonworkingHours, ItemNonworkingUnits property to define non-working days or hours. Use the AddNonworkingDate method to add custom dates as being nonworking date. For instance, the Add("Task:Split") property adds a combination of Task*  *and Split*  *bars, so the Task bar is displayed in working area, and the Split bar is displayed in the non-working area. In other words you have a Task bar that 's interrupted for each non-working unit. For instance, "Task:Progress" adds a new bar that displays the Task shape in working areas, and the Progress shape in non-working area. The A and B could be any known*

bar at the adding time. For instance, if you have added bars like "MyTask" and "MySplit" you can define the bar "MyTask:MySplit", and so on.

4. 

If the Name parameter includes % and : characters, so it's like ***A%B:C*** *it combines the cases 2 and 3.*
5. ***OwnerDraw****, defines an owner-draw bar. The BeforeDrawPart(exOwnerDrawBar) event occurs just before drawing the owner-draw bar, while the AfterDrawPart(exOwnerDrawBar) event notifies your application once the owner-draw bar is drawn.*



The Shortcut property adds a shortcut for the bar, so you can use short names when using the AddBar method . Use the AddBar property to add a new bar to an item. Use the Shape, Pattern and Color properties to define the appearance for the middle part of the bar. Use the StartShape and StartColor properties to define the appearance for the starting part of the bar. Use the EndShape and EndColor properties to define the appearance for the ending part of the bar. The Name property indicates the name of the bar. Use the Copy property to create a clone bar. Use the AllowCreateBar property to specify whether the user can create new bars using the mouse. Use the Height property to specify the height for the bar, if case.

By default, the Bars collection includes the following predefined bars:

- "Deadline": ⇩(this bar can be moved, can't be resized)
- "Project Summary":  (use the DefineSummaryBars method to define bars that belongs to a summary bar)
- "Summary":  (use the DefineSummaryBars method to define bars that belongs to a summary bar)
- "Milestone": ◆(this bar can be moved, can't be resized)
- "Progress":

- "Split":
- "Task":

The Color property of the Bar object specifies the color being used to paint the bar. This property changes the colors for all bars with the same name. For instance, if you have 3 "Task" bars, and you are changing the color for the "Task" bar, the color is applied to all "Task" bars in the chart. For instance, in order to provide "Task" bars with different colors, you can use the Copy method to copy the Task bar to a new bar, and use the Color to change the color of the bar. The following function generates a Task bar with specified color:

```
Private Function AddTask(ByVal gantt As EXG2ANTTLibCtl.G2antt, ByVal clr As Long) As
String
   Dim sT As String
   sT = "Task:" & clr
   With gantt.Chart.Bars.Copy("Task", sT)
      .color = clr
   End With
   AddTask = sT
End Function
```

The following VB sample adds a custom shape and defines a bar like this :

```
With G2antt1.Chart.Bars
   .AddShapeCorner 12345, 1
   With .Add("Task2")
      .Pattern = exPatternDot
      .Shape = exShapeThinDown
      .StartShape = 12345
      .StartColor = RGB(255, 0, 0)
      .Color = .StartColor
   End With
End With
```

The following C++ sample adds a custom shape and defines a bar like above:

```
CBars bars = m_g2antt.GetChart().GetBars();
bars.AddShapeCorner( COleVariant( (long)12345 ), COleVariant( (long)1 ) );
CBar bar = bars.Add("Task2");
bar.SetPattern( 2 /*exPatternDot*/ );
```

```
bar.SetShape( 20 /*exShapeThinDown*/ );
bar.SetStartShape( 12345 );
bar.SetStartColor( RGB(255, 0, 0) );
bar.SetColor( bar.GetStartColor() );
```

The following VB.NET sample adds a custom shape and defines a bar like above:

```
With AxG2antt1.Chart.Bars
   .AddShapeCorner(12345, 1)
   With .Add("Task2")
      .Pattern = EXG2ANTTLib.PatternEnum.exPatternDot
      .Shape = EXG2ANTTLib.ShapeBarEnum.exShapeThinDown
      .StartShape = 12345
      .StartColor = RGB(255, 0, 0)
      .Color = .StartColor
   End With
End With
```

The following C# sample adds a custom shape and defines a bar like above:

```
axG2antt1.Chart.Bars.AddShapeCorner(12345, 1);
EXG2ANTTLib.Bar bar = axG2antt1.Chart.Bars.Add("Task2");
bar.Pattern = EXG2ANTTLib.PatternEnum.exPatternDot;
bar.Shape = EXG2ANTTLib.ShapeBarEnum.exShapeThinDown;
bar.StartShape = (EXG2ANTTLib.ShapeCornerEnum)12345;
bar.StartColor = ToUInt32(Color.FromArgb(255, 0, 0));
bar.Color = bar.StartColor;
```

The following VFP sample adds a custom shape and defines a bar like above:

```
With thisform.G2antt1.Chart.Bars
   .AddShapeCorner(12345, 1)
   With .Add("Task2")
      .Pattern = 2 && exPatternDot
      .Shape = 20 && exShapeThinDown
      .StartShape = 12345
      .StartColor = RGB(255, 0, 0)
      .Color = .StartColor
   EndWith
```

EndWith

method Bars.AddShapeCorner (Key as Variant, Icon as Variant)

Adds a custom shape corner.

Type	Description
Key as Variant	A Long expression that indicates the key of the new icon being added
Icon as Variant	A long expression that indicates the handle of the icon being inserted, or the index of the icon being added.

Use the AddShapeCorner method to define a corner from an icon. Use the StartShape and EndShape properties to define the start and end parts of the bar using custom shapes. Use the Images or ReplaceIcon method to update the list of control's icons. Use the RemoveShapeCorner method to remove a custom shape. The control includes a list of predefined shapes like shown in the ShapeCornerEnum type. **The icon is processed before displaying based on the StartColor/ EndColor property. For instance, if you add an black and white icon, and the StartColor/EndColor is red, the icon will be painted in red. Instead, if the StartColor/EndColor property is -1 (0xFFFFFFFF, not white which is 0x00FFFFFF), the icon is painted as it was added using the AddShapeCorner without any image processing.** If the StartColor/EndColor property is not -1, it indicates the color being applied to the icon.

The following VB sample adds a custom shape and defines a bar like this  :

```
With .Chart.Bars
        .AddShapeCorner 12345, 1
        With .Add("Task2")
           .Pattern = exPatternDot
           .Shape = exShapeThinDown
           .EndShape = 12345
           .EndColor = RGB(255, 0, 0)
           .Color = .EndColor
        End With
    End With
```

The following C++ sample adds a custom shape and defines a bar like above:

```
CBars bars = m_g2antt.GetChart().GetBars();
bars.AddShapeCorner( COleVariant( (long)12345 ), COleVariant( (long)1 ) );
CBar bar = bars.Add("Task2");
bar.SetPattern( 2 /*exPatternDot*/ );
```

```
bar.SetShape( 20 /*exShapeThinDown*/ );
bar.SetEndShape( 12345 );
bar.SetEndColor( RGB(255, 0, 0) );
bar.SetColor( bar.GetEndColor() );
```

The following VB.NET sample adds a custom shape and defines a bar like above:

```
With AxG2antt1.Chart.Bars
   .AddShapeCorner(12345, 1)
   With .Add("Task2")
      .Pattern = EXG2ANTTLib.PatternEnum.exPatternDot
      .Shape = EXG2ANTTLib.ShapeBarEnum.exShapeThinDown
      .EndShape = 12345
      .EndColor = RGB(255, 0, 0)
      .Color = .EndColor
   End With
End With
```

The following VB.NET sample adds a custom icon to the start of all Task bars:

```
With AxG2antt1.Chart.Bars
   .AddShapeCorner(12345, 1)
   .Item("Task").StartShape = 12345
   .Item("Task").StartColor = UInteger.MaxValue
End With
```

The following C# sample adds a custom shape and defines a bar like above:

```
axG2antt1.Chart.Bars.AddShapeCorner(12345, 1);
EXG2ANTTLib.Bar bar = axG2antt1.Chart.Bars.Add("Task2");
bar.Pattern = EXG2ANTTLib.PatternEnum.exPatternDot;
bar.Shape = EXG2ANTTLib.ShapeBarEnum.exShapeThinDown;
bar.EndShape = (EXG2ANTTLib.ShapeCornerEnum)12345;
bar.EndColor = ToUInt32(Color.FromArgb(255, 0, 0));
bar.Color = bar.EndColor;
```

The following C# sample adds a custom icon to the start of all Task bars:

```
EXG2ANTTLib.Bars bars = axG2antt1.Chart.Bars;
```

```
bars.AddShapeCorner(12345, 1);
bars["Task"].StartShape = EXG2ANTTLib.ShapeCornerEnum.exShapeIconEmpty + 12345;
bars["Task"].StartColor = 0xFFFFFFFF;
```

The following VFP sample adds a custom shape and defines a bar like above:

```
With thisform.G2antt1.Chart.Bars
   .AddShapeCorner(12345, 1)
   With .Add("Task2")
      .Pattern = 2 && exPatternDot
      .Shape = 20 && exShapeThinDown
      .EndShape = 12345
      .EndColor = RGB(255, 0, 0)
      .Color = .EndColor
   EndWith
EndWith
```

method Bars.Clear ()

Removes all objects in a collection.

Type	Description

Use the Clear method to clear the Bars collection. Use the Remove method to remove a bar from the Bars collection. Use the Add method to add new bars to the collection. Use the ClearBars method to clear the bars from an item. Use the RemoveBar method to remove a bar from an item. Use the Refresh method to refresh the control.

method Bars.Copy (Name as String, NewName as String)

Copies a Bar object and returns a reference to the newly created object.

Type	Description
Name as String	A String expression that indicates the name of the bar being copied.
NewName as String	A String expression that indicates the name of the new bar.

Return	Description
Bar	A Bar object being created.

Use the Copy property create a clone for a specified bar. Use the Shape, Pattern and Color properties to define the appearance for the middle part of the bar. Use the StartShape and StartColor properties to define the appearance for the starting part of the bar. Use the EndShape and EndColor properties to define the appearance for the ending part of the bar.

The following VB sample creates a new bar called "Task2", that's similar with the "Task" bar excepts that we change the color to fill the bar:

```
With G2antt1.Chart.Bars
   With .Copy("Task", "Task2")
      .Color = RGB(255, 0, 0)
   End With
End With
```

The following C++ sample creates a new bar called "Task2", that's similar with the "Task" bar excepts that we change the color to fill the bar:

```
CBars bars = m_g2antt.GetChart().GetBars();
CBar bar = bars.Copy( "Task", "Task2" );
bar.SetColor( RGB(255,0,0) );
```

The following VB.NET sample creates a new bar called "Task2", that's similar with the "Task" bar excepts that we change the color to fill the bar:

```
With AxG2antt1.Chart.Bars
   With .Copy("Task", "Task2")
      .Color = ToUInt32(Color.Red)
   End With
```

```
End With
```

The following C# sample creates a new bar called "Task2", that's similar with the "Task" bar excepts that we change the color to fill the bar:

```
EXG2ANTTLib.Bar bar = axG2antt1.Chart.Bars.Copy("Task", "Task2");
bar.Color = ToUInt32(Color.Red);
```

The following VFP sample creates a new bar called "Task2", that's similar with the "Task" bar excepts that we change the color to fill the bar:

```
with thisform.G2antt1.Chart.Bars
   with .Copy("Task", "Task2" )
      .Color = RGB(255,0,0)
   endwith
endwith
```

In VB.NET or C# you require the following functions until the .NET framework will provide:

You can use the following VB.NET function:

```
    Shared Function ToUInt32(ByVal c As Color) As UInt32
       Dim i As Long
    i = c.R
    i = i + 256 * c.G
    i = i + 256 * 256 * c.B
    ToUInt32 = Convert.ToUInt32(i)
    End Function
```

You can use the following C# function:

```
private UInt32 ToUInt32(Color c)
{
   long i;
   i = c.R;
   i = i + 256 * c.G;
   i = i + 256 * 256 * c.B;
   return Convert.ToUInt32(i);
}
```

property Bars.Count as Long

Returns the number of objects in a collection.

Type	Description
Long	A long expression that indicates the number of Bar objects in the Bars collection.

The Count property counts the bars in the collection. Use the Item property to access a Bar object in the Bars collection. Use the Remove method to remove a bar from the Bars collection. Use the Clear method to clear the Bars collection. Use the Name property to retrieve the name of the bar. Use the ItemBar(exBarsCount) property to retrieve the number of bars in a specified item.

The following VB sample enumerates the Bar objects in the Bars collection (the order of the elements is arbitrary):

```
With G2antt1.Chart
   Dim b As EXG2ANTTLibCtl.Bar
   For Each b In .Bars
      Debug.Print b.Name
   Next
End With
```

The following VB sample enumerates the Bar objects in the Bars collection (the list is alphabetically sorted):

```
With G2antt1.Chart.Bars
   Dim i As Long
   For i = 0 To .Count - 1
      Debug.Print .Item(i).Name
   Next
End With
```

The following C++ sample enumerates the Bar objects in the Bars collection:

```
CBars bars = m_g2antt.GetChart().GetBars();
for ( long i = 0; i < bars.GetCount(); i++ )
   OutputDebugString( bars.GetItem( COleVariant( i ) ).GetName() );
```

The following VB.NET sample enumerates the Bar objects in the Bars collection:

```
With AxG2antt1.Chart
   Dim b As EXG2ANTTLib.Bar
   For Each b In .Bars
      Debug.Write(b.Name)
   Next
End With
```

The following VB.NET sample enumerates the Bar objects in the Bars collection:

```
With AxG2antt1.Chart.Bars
   Dim i As Integer
   For i = 0 To .Count - 1
      Debug.Write(.Item(i).Name)
   Next
End With
```

The following C# sample enumerates the Bar objects in the Bars collection:

```
EXG2ANTTLib.Bars bars = axG2antt1.Chart.Bars;
for (int i = 0; i < bars.Count; i++)
   System.Diagnostics.Debug.Write(bars[i].Name);
```

The following VFP sample enumerates the Bar objects in the Bars collection:

```
local i
With thisform.G2antt1.Chart.Bars
   for i = 0 to .Count - 1
      wait window nowait .Item(i).Name
   next
EndWith
```

property Bars.Item (Name as Variant) as Bar

Returns a specific Column of the Columns collection.

Type	Description
Name as Variant	A string expression that indicates the name of the bar being accessed, a long expression that indicates the index of the Bar being accessed
Bar	A Bar object being accessed.

Use the Item property to access a Bar object in the Bars collection. The Count property counts the bars in the collection. Use the Remove method to remove a bar from the Bars collection. Use the Clear method to clear the Bars collection. Use the Name property to retrieve the name of the bar. The Bars collection contains several predefined bars like follows:

By default, the Bars collection includes the following predefined bars:

- "Deadline":
- "Project Summary":
- "Summary":
- "Milestone":
- "Progress":
- "Split":
- "Task":

The following VB sample enumerates the Bar objects in the Bars collection (the order of the elements is arbitrary):

```
With G2antt1.Chart
   Dim b As EXG2ANTTLibCtl.Bar
   For Each b In .Bars
      Debug.Print b.Name
   Next
End With
```

The following VB sample enumerates the Bar objects in the Bars collection (the list is alphabetically sorted):

```
With G2antt1.Chart.Bars
   Dim i As Long
   For i = 0 To .Count - 1
```

```
    Debug.Print .Item(i).Name
  Next
End With
```

The following C++ sample enumerates the Bar objects in the Bars collection:

```
CBars bars = m_g2antt.GetChart().GetBars();
for ( long i = 0; i < bars.GetCount(); i++ )
  OutputDebugString( bars.GetItem( COleVariant( i ) ).GetName() );
```

The following VB.NET sample enumerates the Bar objects in the Bars collection:

```
With AxG2antt1.Chart
  Dim b As EXG2ANTTLib.Bar
  For Each b In .Bars
    Debug.Write(b.Name)
  Next
End With
```

The following VB.NET sample enumerates the Bar objects in the Bars collection:

```
With AxG2antt1.Chart.Bars
  Dim i As Integer
  For i = 0 To .Count - 1
    Debug.Write(.Item(i).Name)
  Next
End With
```

The following C# sample enumerates the Bar objects in the Bars collection:

```
EXG2ANTTLib.Bars bars = axG2antt1.Chart.Bars;
for (int i = 0; i < bars.Count; i++)
  System.Diagnostics.Debug.Write(bars[i].Name);
```

The following VFP sample enumerates the Bar objects in the Bars collection:

```
local i
With thisform.G2antt1.Chart.Bars
  for i = 0 to .Count - 1
    wait window nowait .Item(i).Name
```

```
    next
EndWith
```

method Bars.Remove (Name as Variant)

Removes a specific member from the Bars collection.

Type	Description
Name as Variant	A string expression that indicates the name of the bar being removes, a long expression that indicates the index of the Bar being removed

Use the Remove method to remove a bar from the Bars collection. Use the Add method to add new bars to the collection. Use the Clear method to clear the bars collection. Use the ClearBars method to clear the bars from an item. Use the RemoveBar method to remove a bar from an item. Use the Refresh method to refresh the control.

method Bars.RemoveShapeCorner (Key as Variant)

Removes a custom shape corner.

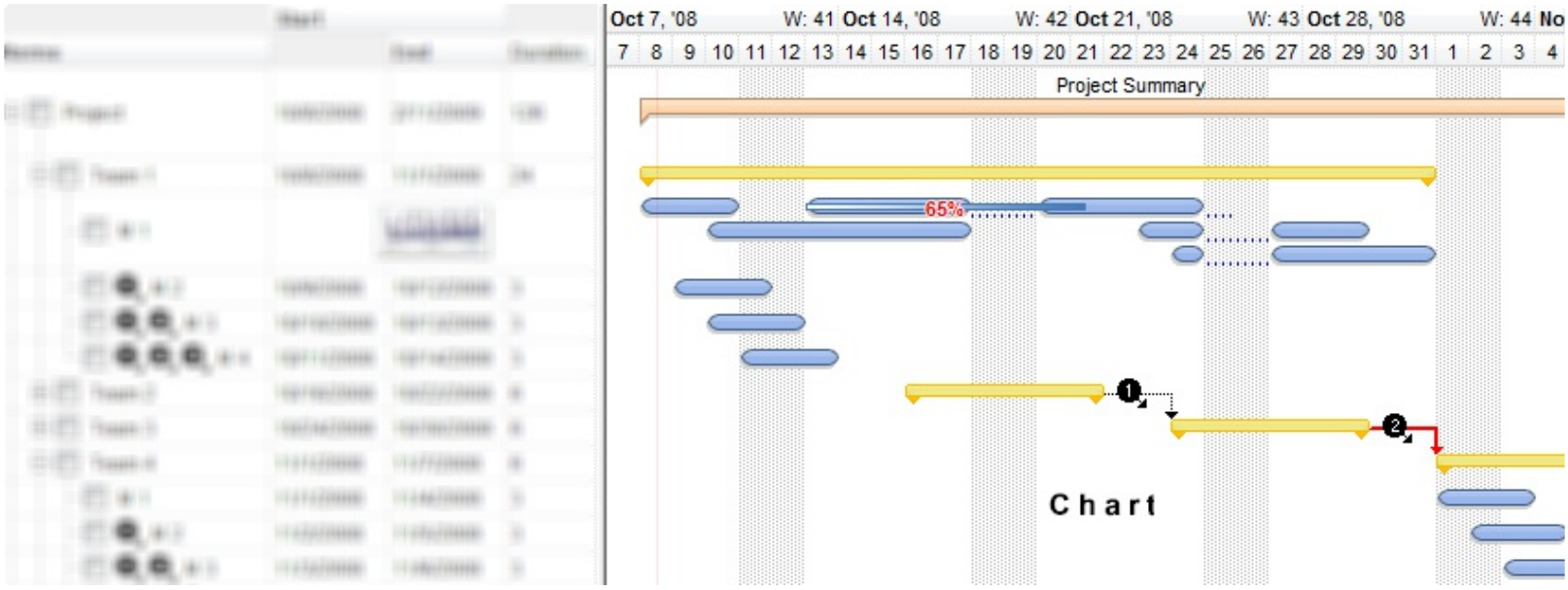
Type	Description
Key as Variant	A long expression that indicates the key of the shape being removed.

Use the RemoveShapeCorner property to remove a shape corner being added using the AddShapeCorner method. Use the StartShape and EndShape properties to define the start and end parts of the bar using custom shapes. Use the Images or ReplaceIcon method to update the list of control's icons. The control includes a list of predefined shapes like shown in the ShapeCornerEnum type.

Chart object

The Chart object contains all properties and methods related to the G2antt chart area. The chart area displays the visible bars only. Use the Bars property to access the control's Bars collection. Use the PaneWidth property to specify the width of the chart area. Use the AddBar property to add new bars to an item. Use the LevelCount property to specify the number of levels in the control's header.

The following screen shots show only the chart part of the control:



The Chart object supports the following properties and methods:

Name	Description
AddNonworkingDate	Adds a nonworking date.
AdjustLevelsToBase	Specifies whether the levels are adjusted on the base level.
AllowCreateBar	Allows creating new bars using the mouse.
AllowInsideZoom	Specifies whether the chart can magnify only parts of the chart.
AllowLinkBars	Specifies whether the user can link the bars using the mouse.
AllowNonworkingBars	Specifies whether the chart treats bars with exBarTreatAsNonworking as non-working parts of the item.
AllowOverviewZoom	Gets or sets a value that indicates whether the user can zoom the chart at runtime.
	Specifies whether the user can enlarge (zoom-in,zoom-

AllowResizeChart	out) or resize the chart using the control's header, middle mouse button.
AllowResizeInsideZoom	Specifies whether the user can resize the inside zoom unit.
AllowSelectDate	Specifies whether the user selects dates at runtime.
AllowSelectObjects	Sets or gets a value that indicates whether the user can select objects in the chart.
AllowSplitPane	Specifies whether the chart panel supports splitting.
AllowUndoRedo	Enables or disables the Undo/Redo feature.
AllowZoomOnFly	Magnifies the bar from the cursor, when the user presses the CTRL / SHIFT key combination.
AMPM	Specifies the AM and PM indicators.
BackColor	Retrieves or sets a value that indicates the chart's background color.
BackColorLevelHeader	Specifies the background color for the chart's levels.
BackColorZoomOnFly	Specifies the background color for the zoom-on-fly panel.
BarFromPoint	Retrieves the bar from point.
Bars	Retrieves the Bars collection.
BarsAllowSizing	Specifies whether bars can be resized at run-time.
CanRedo	Retrieves a value that indicates whether the chart can perform a Redo operation.
CanUndo	Retrieves a value that indicates whether the chart can perform an Undo operation.
ClearItemBackColor	Clears the item's background color in the chart area.
ClearNonworkingDates	Clears nonworking dates.
ColumnsFont	Retrieves or sets the font to display the columns in the chart section.
ColumnsFormatLevel	Specifies the CRD format layout to display the columns in the chart section.
ColumnsTransparent	Specifies the percent of the transparency to display the columns in the chart.
CondInsideZoom	Specifies the formula that indicates the dates that can be zoomed at runtime.
CountVisibleUnits	Counts the number of units within the specified range.
DateFromPoint	Retrieves the date from the cursor.

DateTickerLabel	Retrieves or sets a value that indicates the format to display the bar's start and end date while creating, moving or resizing it.
DefaultInsideZoomFormat	Retrieves the format of the inside zoom units.
DrawDateTicker	Retrieves or sets a value that indicates whether the control draws a ticker around the current date while cursor hovers the chart's client area.
DrawGridLines	Retrieves or sets a value that indicates whether the grid lines are visible or hidden.
DrawLevelSeparator	Retrieves or sets a value that indicates whether lines between levels are shown or hidden.
EndBlockUndoRedo	Ends recording the UI operations and adds the undo/redo operations as a block, so they all can be restored at once, if Undo method is performed.
EndPrintDate	Retrieves or sets a value that indicates the printing end date.
FirstVisibleDate	Retrieves or sets a value that indicates the first visible date.
FirstWeekDay	Specifies the first day of the week.
ForeColor	Retrieves or sets a value that indicates the chart's foreground color.
ForeColorLevelHeader	Specifies the foreground color for the chart's levels.
FormatDate	Formats the date.
GridLineStyle	Retrieves or sets a value that indicates style for the gridlines being shown in the chart area.
GroupUndoRedoActions	Groups the next to current Undo/Redo Actions in a single block.
HistogramBackColor	Specifies the background color of the chart's histogram.
HistogramHeaderVisible	Specifies whether a copy of chart's header is displayed in the bottom side of the histogram.
HistogramHeight	Specifies whether the height of the chart's histogram.
HistogramUnitCount	Specifies the time-scale count to determine the effort of bars with variable-effort (effort of expression/string type)
HistogramUnitScale	Specifies the time-scale unit to determine the effort of bars with variable-effort (effort of expression/string type)
	Gets the value in the histogram at specified date-time, for

HistogramValue	giving type of bars or/and groups.
HistogramValueFromPoint	Retrieves the value from the histogram at the cursor position.
HistogramView	Specifies the list of items being included in the histogram.
HistogramVisible	Specifies whether the chart's histogram layout is visible or hidden.
HistogramZOrder	Specifies the z-order of the bars to be shown within the chart's histogram.
InsideZoomOnDblClick	Gets or sets a value that indicates whether a portion of the chart is magnified or zoomed when the user double click a date.
InsideZooms	Retrieves the collection of inside zoom dates.
IsDateVisible	Specifies whether the date fits the control's chart area.
IsNonworkingDate	Specifies whether giving date-time is a nonworking unit.
ItemBackColor	Retrieves or sets a background color for a specific item, in the chart area.
Label	Retrieves or sets a value that indicates the predefined format of the level's label for a specified unit.
LabelToolTip	Retrieves or sets a value that indicates the predefined format of the level's tooltip for a specified unit.
Level	Retrieves the level based on its index.
LevelCount	Specifies the number of levels in the control's header.
LevelFromPoint	Retrieves the index of the level from the point.
LinkFromPoint	Retrieves the link from the point.
LinksColor	Specifies the color to draw the links between the bars.
LinksStyle	Specifies the style to draw the links between the bars.
LinksWidth	Specifies the width in pixels of the pen to draw the links between the bars.
LocAMPM	Retrieves the time marker such as AM or PM using the current user regional and language settings.
LocFirstWeekDay	Indicates the first day of the week, as specified in the regional settings.
LocMonthNames	Retrieves the list of month names, as indicated in the regional settings, separated by space.

LocWeekDays	Retrieves the list of names for each week day, as indicated in the regional settings, separated by space.
MarkNow	Specifies the the current time to show in the chart.
MarkNowColor	Specifies the background color or the visual appearance of the object that indicates the current time in the chart.
MarkNowCount	Specifies the number of time units to count while highlighting the current time.
MarkNowDelay	Specifies the delay to show the current time in the chart.
MarkNowTransparent	Specifies the percent of the transparency to display the object that marks the current time.
MarkNowUnit	Retrieves or sets a value that indicates the base time unit while highlighting the current time.
MarkNowWidth	Specifies the width in pixels of the object that shows the current time.
MarkSelectDateColor	Retrieves or sets a value that indicates the color to mark the selected date in the chart.
MarkTimeZone	Highlights a specified time zone from start to end with a different background color, pattern, transparency, HTML captions and so on.
MarkTodayColor	Retrieves or sets a value that indicates the color to mark today in the chart.
MaxUnitWidth	Specifies the maximum value for Chart.UnitWidth property while enlarge or zoom-in/zoom-out operation is performed.
MinUnitWidth	Specifies the minimum value for Chart.UnitWidth property while enlarge or zoom-in/zoom-out operation is performed.
MonthNames	Retrieves or sets a value that indicates the list of month names, separated by space.
NextDate	Gets the next date based on the unit.
NonworkingDays	Retrieves or sets a value that indicates the non-working days, for each week day a bit.
NonworkingDaysColor	Retrieves or sets a value that indicates the color to fill the non-working days.
NonworkingDaysPattern	Retrieves or sets a value that indicates the pattern being used to fill non-working days.
NonworkingHours	Retrieves or sets a value that indicates the non-working hours, for each hour in a day a bit.

NonworkingHoursColor	Retrieves or sets a value that indicates the color to fill the non-working hours.
NonworkingHoursPattern	Retrieves or sets a value that indicates the pattern being used to fill non-working hours.
NoteFromPoint	Retrieves the note from the point.
Notes	Retrieves the Notes collection.
OverlaidOnMoving	Specifies whether the overlaid bars are re-arranged while the user moves or resizes at runtime a bar.
OverviewBackColor	Specifies the background color of the chart's overview.
OverviewHeight	Indicates the height of the chart's overview.
OverviewLevelLines	Indicates the index of the level that displays the grid line in the chart's overview.
OverviewSelBackColor	Specifies the selection color of the chart's overview.
OverviewSelTransparent	Specifies the percent of the transparency to display the selection in the overview parts of the control.
OverviewShowMarkTimeZones	Specifies whether the chart's overview shows the marked time-zones.
OverviewShowSelectDates	Specifies whether the chart's overview shows the selected dates.
OverviewToolTip	Retrieves or sets a value that indicates the format of the tooltip being shown while the cursor hovers the chart's overview area.
OverviewVisible	Specifies whether the chart's overview layout is visible or hidden.
OverviewZoomCaption	Specifies the captions for each zooming unit.
OverviewZoomUnit	Indicates the width in pixels of the zooming unit in the overview.
PaneWidth	Specifies the width for the left or side pane.
Picture	Retrieves or sets a graphic to be displayed in the chart.
PictureDisplay	Retrieves or sets a value that indicates the way how the graphic is displayed on the chart's background
Redo	Redoes the next action in the chart's Redo queue.
RedoListAction	Lists the Redo actions that can be performed in the chart.
RedoRemoveAction	Removes the first redo actions that can be performed in the chart.

RemoveNonworkingDate	Removes a nonworking date.
RemoveSelection	Removes the selected objects within the chart.
RemoveTimeZone	Removes a time-zone being highlighted using the MarkTimeZone method.
ResizeUnitCount	Specifies the number of time units while resizing, moving or creating bars by dragging.
ResizeUnitScale	Retrieves or sets a value that indicates the base time unit while resizing, moving or creating the bars by dragging.
ScrollBar	Shows or hides the chart's horizontal scroll bar.
ScrollRange	Specifies the range of dates to scroll within.
ScrollTo	Scrolls the chart so the specified date is visible.
SelBackColor	Retrieves or sets a value that indicates the selection background color.
SelBarColor	Retrieves or sets a value that indicates the color or EBN object to display the selected bars.
SelectDate	Selects or unselects a specific date in the chart.
SelectDates	Indicates a collection of date-time units being selected.
SelectLevel	Indicates the index of the level that highlights the selected dates.
SelectOnClick	Specifies whether an item gets selected once the user clicks the chart area.
SelForeColor	Retrieves or sets a value that indicates the selection foreground color.
SelLinkColor	Specifies the color to show the selected link (or for rectangular links an EBN object to highlights the link, without changing the link's color).
ShowCollapsedBars	Gets or sets a value that indicates whether the collapsed items displays their child bars.
ShowEmptyBars	Specifies whether empty bars are shown or hidden. An empty bar has the start and end dates identical.
ShowEmptyBarsUnit	Specifies the unit to be added to the end date, so empty bars are shown.
ShowLinks	Retrieves or sets a value that indicates whether the links between bars are visible or hidden.
ShowLinksColor	Retrieves or sets a value that indicates the color to display

	the links based on the user selection.
ShowLinksStyle	Retrieves or sets a value that indicates the style to display the links based on the user selection.
ShowLinksWidth	Retrieves or sets a value that indicates the width to display the links based on the user selection.
ShowNonworkingDates	Shows or hides nonworking dates.
ShowNonworkingHours	Shows or hides nonworking hours.
ShowNonworkingUnits	Retrieves or sets a value that indicates whether the non-working units are visible or hidden.
ShowNotes	Specifies whether all notes or boxes are shown or hidden.
ShowTransparentBars	Gets or sets a value that indicates percent of the transparency to display the bars.
SplitPaneWidth	Specifies the width of split panels, separated by comma.
StartBlockUndoRedo	Starts recording the UI operations as a block of undo/redo operations.
StartPrintDate	Retrieves or sets a value that indicates the printing start date.
TimeZoneFromPoint	Retrieves the time-zone from the cursor.
TimeZoneInfo	Retrieves information about the time-zone giving its key.
ToolTip	Retrieves or sets a value that indicates the format of the tooltip being shown while the user scrolls the chart.
Undo	Performs the last Undo operation.
UndoListAction	Lists the Undo actions that can be performed in the chart.
UndoRedoQueueLength	Gets or sets the maximum number of Undo/Redo actions that may be stored to the chart's queue.
UndoRemoveAction	Removes the last the undo actions that can be performed in the chart.
UnitScale	Retrieves or sets a value that indicates the base unit being displayed.
UnitWidth	Specifies the width in pixels for the minimal unit.
UnitWidthNonworking	Specifies the width in pixels for the minimal unit.
UnselectDates	Unselects all dates in the chart.
UpdateOnMoving	Specifies whether the control moves or resizes all related bars or just the bar being moved or resized.

WeekDays	Retrieves or sets a value that indicates the list of names for each week day, separated by space.
WeekNumberAs	Specifies the way the control displays the week number.
Zoom	Sets or retrieves the magnification scale of the chart.
ZoomOnFlyCaption	Specifies the caption to be shown in the zoom-on-fly panel, when the cursor hovers a bar.

method Chart.AddNonworkingDate (Date as Variant)

Adds a nonworking date.

Type	Description
Date as Variant	A Date expression that indicates the date being marked as nonworking day or a string expression that specifies the repetitive expression that defines the non-working days as Easter, Christmas or Holydays. For instance the "month(value)=7 or (month(value) = 12 and day(value)=25)" indicates July and December 25th is a non-working dates. The string version of the AddNonworkingDate supports value formatting. The **value** keyword indicates the date being queried. *If the expression is not syntactically correct the non-working date expression is not added and so represented.*

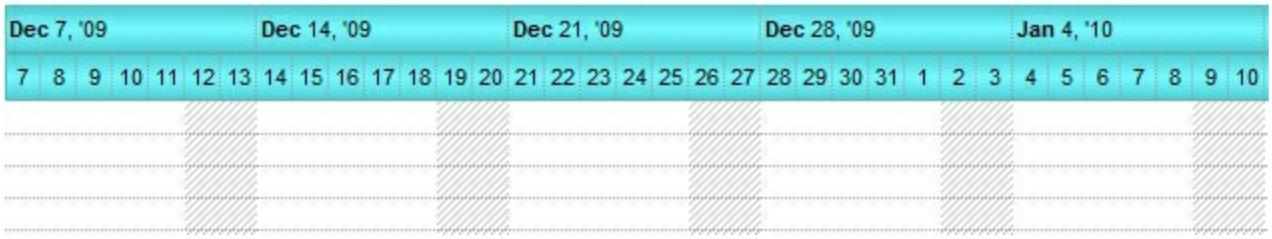
Use the AddNonworkingDate method to add custom dates as nonworking days. Use the NonworkingDays property to mark days in a week as being as nonworking. Use the ShowNonworkingDates property to show or hide the nonworking dates in the control's chart area. Use the RemoveNonworkingDate method to remove a specified date from the nonworking dates collection. The RemoveNonworkingDate method removes only a date previously added using the AddNonworkingDate method. Use the ClearNonworkingDates method to remove all nonworking dates. Use the NonworkingDaysPattern property to specify the pattern being used to fill non-working days. The NonworkingDaysColor property specifies the color being used to fill the non-working days. Use the DateChange event to notify whether the user browses a new date in the chart area. Use the IsNonworkingDate property to retrieve a value that indicates whether a date is marked as nonworking day. Use the Add("A:B") to add a bar that displays the bar A in the working area, and B in non-working areas. Use the ItemNonworkingUnits property to specify different non-working zones for different items.

The control supports the following ways of specify the non-working parts for items:

- NonworkingDays and NonworkingHours properties indicate the nonworking parts of the chart being applied to all items with the exception of those that use the ItemNonworkingUnits property.
- AddNonworkingDate method adds custom dates as being nonworking date which is applied to all items with the exception of those that use the ItemNonworkingUnits property. You can use the AddNonworkingDate to add manually dates or a repetitive expression that defines the non-working days as Easter, Christmas or Holydays.
- ItemNonworkingUnits property defines the repetitive expression to specify the non-working parts in the item.

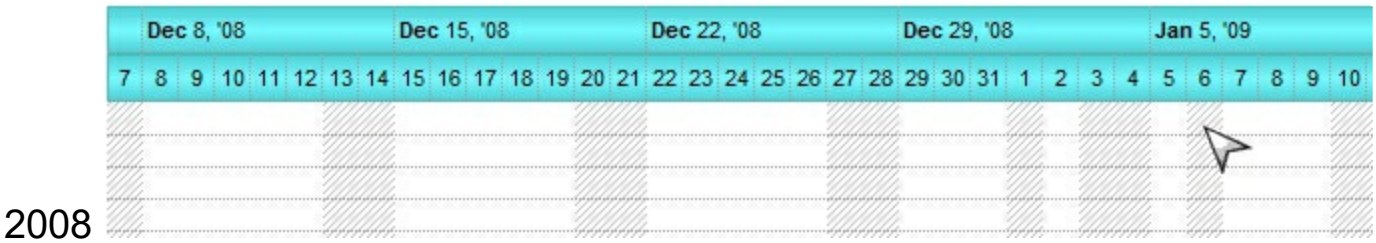
- ItemBar(exBarTreatAsNonworking) indicates whether the bar defines actually the non-working part of the item in addition to ItemNonworkingUnits property (which is required also)

The following screen shot shows the chart with no custom non-working dates (just defined by the NonworkingDays property as exSaturday and exSunday)

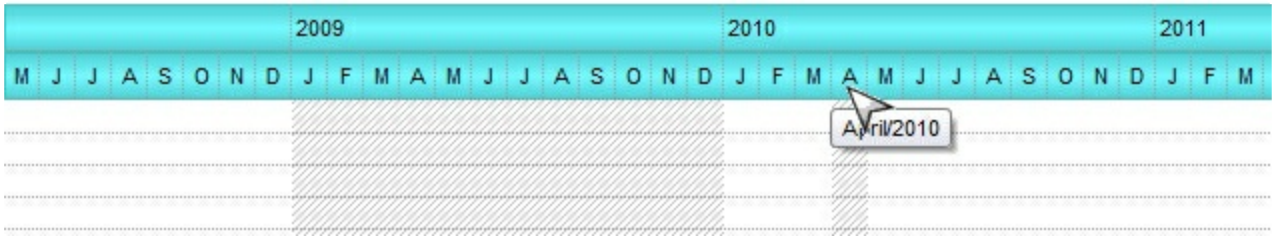


The following screen shot shows the chart with 3 custom non-working dates (**#12/22/2009#, #12/23/2009#, #12/24/2009#**)

The following screen shot shows the chart with a repetitive formula defining the January the 1st and 6th as "**month(value) = 1 and (day(value) in (1,6))**"



The following screen shot shows the chart with a repetitive formula defining the year 2009 as being non-working and the April of 2010 using "year(value) = 2009 or (month(value) = 4 and year(value) = 2010)"



Here's few samples for repetitive expression:

- "month(value) = 7" indicates the entire July month
- "shortdateF(value) left 5 in ('01/01','01/06','04/25','05/01','06/02','08/15','11/01','12/08','12/25','12/26')" indicates Jan 1, Jan 6, Apr 25, May 1, Jun 2, Aug 15, Nov 1, Dec 8, Dec 25 (Christmas), Dec 26.
- "month(value) = 1 and (day(value) in (1,6))" indicates the Jan 1 and Jan 6.
- "hour(value) in (12,13,14)" indicates the time between 12 and 14 as being non-working.
- "not(month(value) in (3,4)) ? 0 : (floor(value)=floor(date(dateS('3/1/' + year(value)) + ((1:=(((255 - 11 * (year(value) mod 19)) - 21) mod 30) + 21) + (=:1 > 48 ? -1 : 0) + 6 - ((year(value) + int(year(value) / 4)) + =:1 + (=:1 > 48 ? -1 : 0) + 1) mod 7))))" indicates the **Easter** day.
- "not(month(value) in (3,4,5)) ? 0 : (floor(value)=(2:=floor(date(dateS('3/1/' + year(value)) + ((1:=(((255 - 11 * (year(value) mod 19)) - 21) mod 30) + 21) + (=:1 > 48 ? -1 : 0) + 6 - ((year(value) + int(year(value) / 4)) + =:1 + (=:1 > 48 ? -1 : 0) + 1) mod 7)))) or (floor(value)= =:2 + 1))" indicates the **Easter** Sunday and a day after

The expression supports predefined functions listed here. The **value** keyword in the expression indicates the date value being queried.

The following samples handles the DateChange event to add a new hard-coded date. The DateChange event notifies the application once the chart displays or changes its first visible date. This version could be time consuming, but it can be improved. For instance, you can add a member or a has table that changes / adds a new working date when the year is changed so actually the action could be added, only when the chart displays a new year.

The following VB sample marks the 11th of each month as nonworking day (the code enumerates the visible dates, and marks one by one, if case):

```
Private Sub G2antt1_DateChange()
  With G2antt1
    .BeginUpdate
```

```
    With .Chart
      Dim d As Date
      d = .FirstVisibleDate
      Do While .IsDateVisible(d)
        If Day(d) = 11 Then
          If Not (.IsNonworkingDate(d)) Then
            .AddNonworkingDate d
          End If
        End If
        d = .NextDate(d, exDay, 1)
      Loop
    End With
    .EndUpdate
  End With
End Sub
```

The following VB.NET sample marks the 11th of each month as nonworking day:

```
Private Sub AxG2antt1_DateChange(ByVal sender As Object, ByVal e As System.EventArgs)
Handles AxG2antt1.DateChange
  With AxG2antt1
    .BeginUpdate()
    With .Chart
      Dim d As Date = .FirstVisibleDate
      Do While .IsDateVisible(d)
        If d.Day = 11 Then
          If Not (.IsNonworkingDate(d)) Then
            .AddNonworkingDate(d)
          End If
        End If
        d = .NextDate(d, EXG2ANTTLib.UnitEnum.exDay, 1)
      Loop
    End With
    .EndUpdate()
  End With
End Sub
```

The following C# sample marks the 11th of each month as nonworking day:

```
private void axG2antt1_DateChange(object sender, EventArgs e)
{
   axG2antt1.BeginUpdate();
   EXG2ANTTLib.Chart chart = axG2antt1.Chart;
   DateTime d = Convert.ToDateTime(chart.FirstVisibleDate);
   while ( chart.get_IsDateVisible(d) )
   {
      if ( d.Day == 11 )
         if ( !chart.get_IsNonworkingDate( d ) )
            chart.AddNonworkingDate(d);
      d = chart.get_NextDate(d, EXG2ANTTLib.UnitEnum.exDay, 1);
   }
   axG2antt1.EndUpdate();
}
}
```

The following VFP sample marks the 11th of each month as nonworking day (DateChange event):

```
*** ActiveX Control Event ***

With thisform.G2antt1
   .BeginUpdate
   With .Chart
      local d
      d = .FirstVisibleDate
      Do While .IsDateVisible(d)
         If Day(d) = 11 Then
            If Not (.IsNonworkingDate(d)) Then
               .AddNonworkingDate(d)
            EndIf
         EndIf
         d = .NextDate(d, 4096, 1)
      enddo
   EndWith
   .EndUpdate
EndWith
```

property Chart.AdjustLevelsToBase as Boolean

Specifies whether the levels are adjusted on the base level.

Type	Description
Boolean	A boolean expression that specifies whether the levels are arranged based on the base level.

By default, the AdjustLevelsToBase property is False. Use the AdjustLevelsToBase property on True, in case you are using a not-contiguous time scale, so you need to align the tick lines from different levels. For instance, if the ShowNonworkingUnits property is False, the AdjustLevelsToBase property is automatically set on True, so the time scale is aligned to the base level. Use the DrawLevelSeperator property to draw horizontally lines between levels inside the chart's header. The DrawTickLines / DrawTickLinesFrom property specify whether the vertically lines between time-units are shown in the level.

property Chart.AllowCreateBar as CreateBarEnum

Allows creating new bars using the mouse.

Type	Description
CreateBarEnum	A CreateBarEnum expression that indicates whether the user can create new bars using the mouse.

By default, the AllowCreateBar property is exCreateBarManual. Use the AllowCreateBar property to disable creating new bars using the mouse. The control fires the CreateBar event when the user releases the mouse in the chart area. The CreateBar event is fired only if the AllowCreateBar property is not zero. The control prevents creating new bars inside disable items, so you can not create new bars in disabled items. The EnableItem property specifies whether an item is enabled or disabled. The ItemBar(,,exBarsCount) property counts the number of bars in giving item

- If the AllowCreateBar property is **exCreateBarAuto** or **exCreateBarAutoEndInclusive**, the control automatically adds a new bar to the item, with the key "newbar", of "Task" type, so it looks like this: . Use the ItemBar property to change the key or the name or any other property of the newly created bar whose exBarKey property is "newbar" and it's exBarName is "Task". In this case, if the CreateBar event is not handled, the user can't add more than a single bar to the selected item, as the "newbar" is not unique, instead, if you handle the CreateBar event, and assign a different key for the newly created bar, several bars can be added to the same item. If the user clicks the empty / non-items zone of the chart, the control may fire the AddItem event for the newly added items so the newly bar will be shown in the clicked area. In this case, the Item parameter indicates the handle of the item that has been added at the last. In other words, the control automatically adds new items and creates the newly bar on the last added item, if the Chart.AllowCreateBar property is exCreateBarAuto.
- If the AllowCreateBar property is **exCreateBarManual** or **exCreateBarManualEndInclusive**, you need to handle the CreateBar event to add new bars using the AddBar method. Samples, are shown bellow. If the Item parameter of the CreateBar event is negative, its absolute value indicates the number of items to be added from the last visible item, so it fits the clicked part of the chart. For instance, CreateBar(-3,Start,End) indicates that a 3 more items should be added so it covers the clicked zone.

If the AllowCreateBar property is exCreateBarAuto, the following samples change the key and the type of the bar being displayed as soon as the CreateBar event is called:

The following VB6 sample changes the key of the newly created bar "newbar", and the name of the bar being displayed as "Task" to "Progress":

```
Private Sub G2antt1_CreateBar(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal DateStart As Date, ByVal DateEnd As Date)
    With G2antt1.Items
        .ItemBar(Item, "newbar", exBarName) = "Progress"
        .ItemBar(Item, "newbar", exBarKey) = DateStart
    End With
End Sub
```

The following VB6 sample prevents creating new tasks/bars on rows/items that already contain bars:

```
Private Sub G2antt1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Dim i, c As Long, hit As HitTestInfoEnum
    With G2antt1
        Dim nAllowCreateBar As CreateBarEnum
        nAllowCreateBar = exCreateBarAuto
            i = .ItemFromPoint(-1, -1, c, hit)
            If (i <> 0) Then
                If Not (0 = .Items.ItemBar(i, "<*>", exBarsCount)) Then
                    nAllowCreateBar = exNoCreateBar
                End If
            End If
        .Chart.AllowCreateBar = nAllowCreateBar
    End With
End Sub
```

The following VB6 sample allows creating bars on leaf items only (items with no children)

```
Private Sub G2antt1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Dim i, c As Long, hit As HitTestInfoEnum
    With G2antt1
```

```
    Dim nAllowCreateBar As CreateBarEnum
    nAllowCreateBar = exCreateBarAuto
      i = .ItemFromPoint(-1, -1, c, hit)
      If (i <> 0) Then
        If Not .Items.ChildCount(i) = 0 Then
          nAllowCreateBar = exNoCreateBar
        End If
      End If
    .Chart.AllowCreateBar = nAllowCreateBar
  End With
End Sub
```

The following VB6 sample disables or prevents creating bars inside specific items (items with no parent):

```
Private Sub G2antt1_CreateBar(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal DateStart As
Date, ByVal DateEnd As Date)
  With G2antt1.Items
    If (.ItemParent(Item) = 0) Then
      .RemoveBar Item, "newbar"
    End If
  End With
End Sub
```

The following C# sample changes the key of the newly created bar "newbar", and the name of the bar being displayed as "Task" to "Progress":

```
private void axG2antt1_CreateBar(object sender,
AxEXG2ANTTLib._IG2anttEvents_CreateBarEvent e)
{
  axG2antt1.Items.set_ItemBar(e.item, "newbar",
EXG2ANTTLib.ItemBarPropertyEnum.exBarName, "Progress");
  axG2antt1.Items.set_ItemBar(e.item, "newbar",
EXG2ANTTLib.ItemBarPropertyEnum.exBarKey, e.dateStart );
}
```

The following VB.NET sample changes the key of the newly created bar "newbar", and the name of the bar being displayed as "Task" to "Progress":

```
Private Sub AxG2antt1_CreateBar(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_CreateBarEvent) Handles AxG2antt1.CreateBar
   With AxG2antt1.Items
      .ItemBar(e.item, "newbar", EXG2ANTTLib.ItemBarPropertyEnum.exBarName) =
"Progress"
      .ItemBar(e.item, "newbar", EXG2ANTTLib.ItemBarPropertyEnum.exBarKey) =
e.dateStart
   End With
End Sub
```

The following C++ sample changes the key of the newly created bar "newbar", and the name of the bar being displayed as "Task" to "Progress":

```
void OnCreateBarG2antt1(long Item, DATE DateStart, DATE DateEnd)
{
   CItems items = m_g2antt.GetItems();
   items.SetItemBar( Item, COleVariant( _T("newbar") ), 0 /*exBarName*/, COleVariant(
_T("Progress") ) );
   items.SetItemBar( Item, COleVariant( _T("newbar") ), 9 /*exBarKey*/, COleVariant(
DateStart ) );
}
```

The following VFP sample changes the key of the newly created bar "newbar", and the name of the bar being displayed as "Task" to "Progress":

```
*** ActiveX Control Event ***
LPARAMETERS item, datestart, dateend

with thisform.G2antt1.Items
   .DefaultItem = item
   thisform.G2antt1.Template = "Items.ItemBar(0,`newbar`,0) = `Progress`"
   thisform.G2antt1.Template = "Items.ItemBar(0,`newbar`,9) = `" + dtos(datestart) + "`"
endwith
```

The Template property helps you to call any of the control's property using x-script.

If the AllowCreateBar property is exCreateBarManual, the following samples adds a new task bar, as soon as the CreateBar is called:

The following C# sample adds a new task, when the user releases the mouse:

```
private void axG2antt1_CreateBar(object sender,
AxEXG2ANTTLib._IG2anttEvents_CreateBarEvent e)
{
   Random randomKey = new Random();
   axG2antt1.BeginUpdate();
   axG2antt1.Items.AddBar(e.item, "Task", e.dateStart, e.dateEnd, randomKey.Next(), "");
   axG2antt1.EndUpdate();
}
```

The following C++ sample adds a new task, when the user releases the mouse:

```
void OnCreateBarG2antt1(long Item, DATE DateStart, DATE DateEnd)
{
   m_g2antt.BeginUpdate();
   CItems items = m_g2antt.GetItems();
   items.AddBar( Item, COleVariant( "Task" ), COleVariant( DateStart ), COleVariant( DateEnd
), COleVariant( (long)rand() ), COleVariant( "" ) );
   m_g2antt.EndUpdate();
}
```

The following VB sample adds a new task, when the user releases the mouse:

```
Private Sub G2antt1_CreateBar(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal DateStart As
Date, ByVal DateEnd As Date)
   With G2antt1
      .BeginUpdate
      With .Items
         .AddBar Item, "Task", DateStart, DateEnd, Rnd
      End With
      .EndUpdate
   End With
End Sub
```

The following VFP sample adds a new task, when the user releases the mouse:

```
*** ActiveX Control Event ***
LPARAMETERS item, datestart, dateend
```

```
with thisform.G2antt1
   .BeginUpdate
   with .Items
      .AddBar( item, "Task", datestart, dateend, RAND() )
   endwith
   .EndUpdate
endwith
```

The following VB.NET sample adds a new task, when the user releases the mouse:

```
Private Sub AxG2antt1_CreateBar(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_CreateBarEvent) Handles AxG2antt1.CreateBar
   With AxG2antt1
      .BeginUpdate()
      With .Items
         .AddBar(e.item, "Task", e.dateStart, e.dateEnd, Rnd())
      End With
      .EndUpdate()
   End With
End Sub
```

property Chart.AllowInsideZoom as Boolean

Specifies whether the chart can magnify only parts of the chart.

Type	Description
Boolean	A Boolean expression that specifies whether the char may display magnified only portions of the chart.

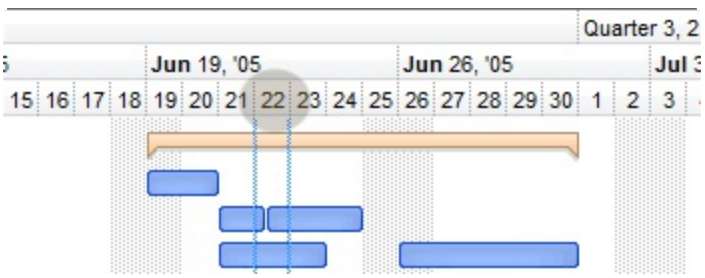
By default, the AllowInsideZoom property is False. The inside zoom feature allows displaying portions of the chart with different time scale units. For instance, you can display the bars on hours, while the chart still displays days. Once the AllowInsideZoom property is True, the user can double clicks the chart's header, so this portion gets magnified. Also, at runtime, the user can resize the time scale units, so the unit gets magnified. Each inside zoom unit is fully customizable, so you can change the background color for the portion being zoomed, draw the grid lines, specify a different format label to be displayed while the unit gets zoomed, and so on. The DateFromPoint property retrieves the date from the cursor based on the inside zoom unit also. Use the DefaultInsideZoomFormat property to specify the format (as background color, grid lines, labels), for new inside zoom units. The InsideZoomOnDblClick property specifies whether the date being double clicked gets magnified. The AllowResizeInsideZoom property specifies whether the user may resize the time units in the levels area, so they get magnified. The CondInsideZoom property specifies a formula that determines the dates that can be magnified by double clicking or resizing.

Use the InsideZooms property to access the inside zoom units.

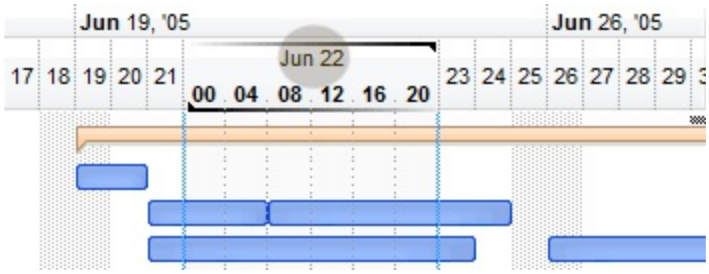
Beside inside zooming, your application can provide the following options to help user while performing moving or resizing the bars at runtime:

- grid lines, that can be shown only when moving or resizing, using the ChartStartChanging and ChartEndChanging events, or all the time
- select date, to specify the margins of the area you want to highlight
- ticker, that shows the cursor's position in the chart, or while resizing, it shows exactly the size and the position of the bar, including starting and ending date.
- ability to specify a resizing/moving unit, different that the displayed one ie while the chart displays days, you can specify the resizing unit on hours.

The following chart displays days:

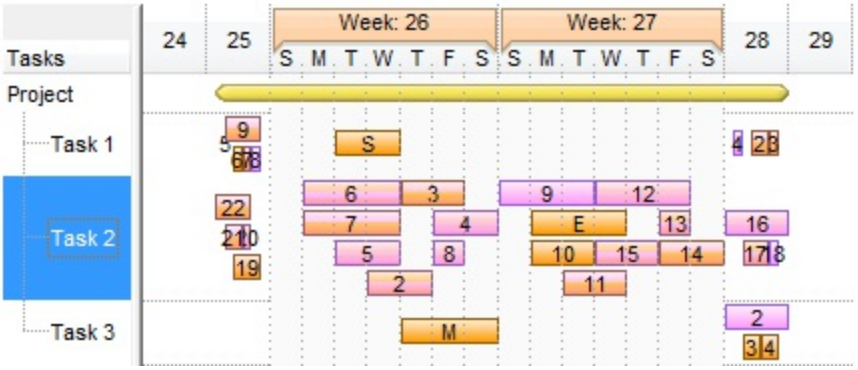


The Jun 22, gets magnified to hours so it looks like follows (the first line displays the day, while the second displays the hours, the rest of the chart displays days):

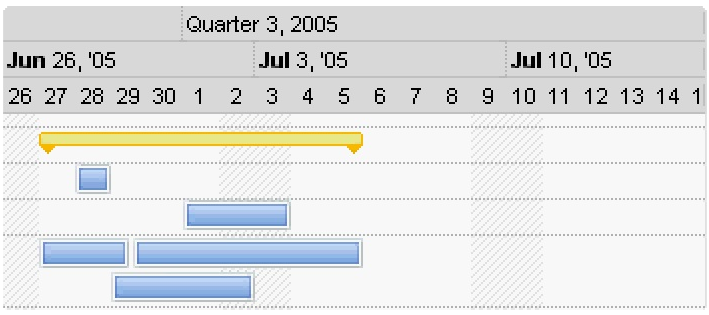


The chart displays weeks:

while the week 26 and 27 gets magnified to days it looks like follows (the first line displays the week number, while the second line displays days, the rest of the chart displays weeks):



The following animation shows resizing the bars, using the inside zoom feature:

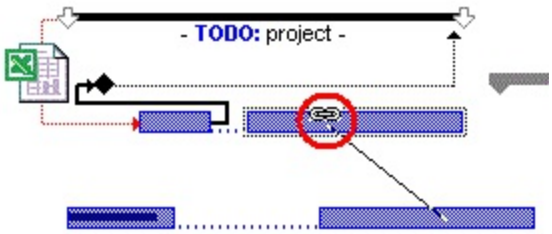


property Chart.AllowLinkBars as Boolean

Specifies whether the user can link the bars using the mouse.

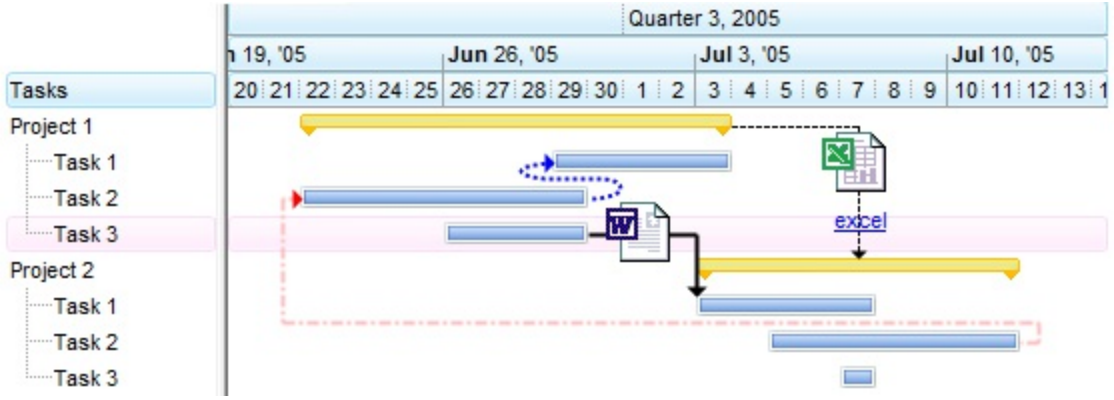
Type	Description
Boolean	A Boolean expression that indicates whether the user can link two bars using the mouse.

By default, the AllowLinkBars property is True. The AllowLinkBars property specifies whether the user can click a bar, and drag the link to a new bar. Use the ShowLink property to show or hide the links in the chart area. If the ShowLink property is False, the AllowLinkBars has no effect. Linking the bars using the mouse works like follows. The user clicks the bar where the link wants to start, and while keeping the left mouse button, he drags up or down the mouse, until the cursor us changed to a link cursor, and from there the user can select a new bar to link to. Once that the user releases the mouse over a bar, the control adds a new link between these two selected bars, if there were no link between bars. Use the AddLink method to add links programmatically. The AddLink event is fired when the control adds a new link between selected bars.



The red circle shows the link cursor and the dot rectangle around the bar that indicates the selected bar to link to. Once that the user releases the mouse on the red circle, the AddLink event is called to specify the key of the link being added. Use the Link property to set or get the properties and options for the specified link.

The following screen shot shows the type of links you can display:



property Chart.AllowNonworkingBars as Boolean

Specifies whether the chart treats bars with exBarTreatAsNonworking as non-working parts of the item.

Type	Description
Boolean	A Boolean expression that specifies whether the bars with exBarTreatAsNonworking set on True indicate non-working parts of the items.

By default, the AllowNonworkingBars property is False. The chart supports bars that may indicates non-working parts of the items where they are hosted. The AllowNonworkingBars have effect only for bar with exBarTreatAsNonworking set on True hosted on an item with the ItemNonworkingUnits property points to a not empty and valid expression. For instance, if the bar with exBarTreatAsNonworking is hosted to an item that has ItemNonworkingUnits property on empty (by default) it is treated as a normal bar. Use the ShowNonworkingUnits property to hide the non-working units. Use the ShowNonworkingDates property to specify whether the the days are shown or hidden while the ShowNonworkingUnits property is False.

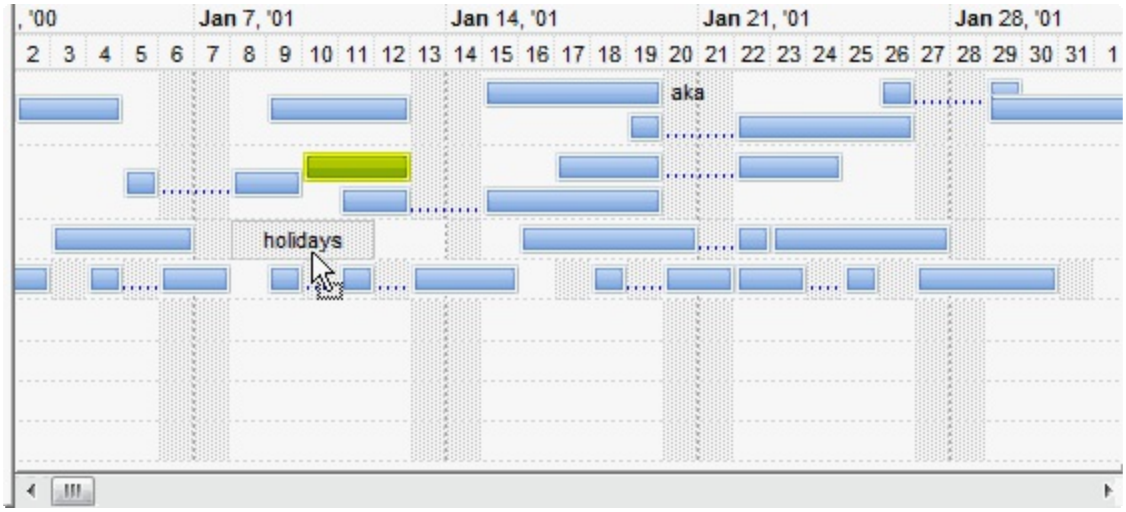
The control supports the following ways of specify the non-working parts for items:

- NonworkingDays and NonworkingHours properties indicate the nonworking parts of the chart being applied to all items with the exception of those that use the ItemNonworkingUnits property.
- AddNonworkingDate method adds custom dates as being nonworking date which is applied to all items with the exception of those that use the ItemNonworkingUnits property.
- ItemNonworkingUnits property defines the repetitive expression to specify the non-working parts in the item.
- ItemBar(exBarTreatAsNonworking) indicates whether the bar defines actually the non-working part of the item in addition to ItemNonworkingUnits property (which is required also)

In conclusion, a bar is treated as a non-working part inside the item if:

- AllowNonworkingBars property is True.
- ItemNonworkingUnits property is not empty, and points to a valid expression. The ItemNonworkingUnits property indicates a repetitive expression to determine the parts of the item being non-working.
- ItemBar(exBarTreatAsNonworking) is True.

The following screen shot shows a "holidays" bar that indicates the non-working parts of the item where it is hosted:



In this sample you can notice that all bars preserves their length (working part), while the "holidays" or any other bar bar is moving.

The following samples adds a bar to be treated as a nonworking part like a "holidays" bar:

VBA (MS Access, Excel...)

```
With G2antt1
   .BeginUpdate
   With .Chart
      .FirstVisibleDate = #1/1/2001#
      .LevelCount = 2
      .PaneWidth(False) = 48
      .AllowNonworkingBars = True
      0.Bars.Add("Task:Split").Shortcut = "Task"
   End With
   .Columns.Add "Tasks"
   With .Items
      h = .AddItem("Task 1")
      .ItemNonworkingUnits(h,False) = "weekday(value) in (0,6)"
      .AddBar h,"",#1/2/2001#,#1/5/2001#,"A","holyday"
      .ItemBar(h,"A",38) = True
      .AddBar h,"Task",#1/5/2001#,#1/12/2001#,"Z"
      .ItemBar(h,"Z",20) = True
   End With
   .EndUpdate
End With
```

VB6

```
With G2antt1
   .BeginUpdate
   With .Chart
      .FirstVisibleDate = #1/1/2001#
      .LevelCount = 2
      .PaneWidth(False) = 48
      .AllowNonworkingBars = True
      0.Bars.Add("Task:Split").Shortcut = "Task"
   End With
   .Columns.Add "Tasks"
   With .Items
      h = .AddItem("Task 1")
      .ItemNonworkingUnits(h,False) = "weekday(value) in (0,6)"
      .AddBar h,"",#1/2/2001#,#1/5/2001#,"A","holyday"
      .ItemBar(h,"A",exBarTreatAsNonworking) = True
      .AddBar h,"Task",#1/5/2001#,#1/12/2001#,"Z"
      .ItemBar(h,"Z",exBarKeepWorkingCount) = True
   End With
   .EndUpdate
End With
```

VB.NET

```
Dim h
With Exg2antt1
   .BeginUpdate()
   With .Chart
      .FirstVisibleDate = #1/1/2001#
      .LevelCount = 2
      .set_PaneWidth(False,48)
      .AllowNonworkingBars = True
      0.Bars.Add("Task:Split").Shortcut = "Task"
   End With
   .Columns.Add("Tasks")
   With .Items
      h = .AddItem("Task 1")
      .set_ItemNonworkingUnits(h,False,"weekday(value) in (0,6)")
```

```
        .AddBar(h,"",#1/2/2001#,#1/5/2001#,"A","holyday")

.set_ItemBar(h,"A",exontrol.EXG2ANTTLib.ItemBarPropertyEnum.exBarTreatAsNonworking,T

        .AddBar(h,"Task",#1/5/2001#,#1/12/2001#,"Z")

.set_ItemBar(h,"Z",exontrol.EXG2ANTTLib.ItemBarPropertyEnum.exBarKeepWorkingCount,Tr

    End With
    .EndUpdate()
End With
```

VB.NET for /COM

```
Dim h
With AxG2antt1
    .BeginUpdate()
    With .Chart
        .FirstVisibleDate = #1/1/2001#
        .LevelCount = 2
        .PaneWidth(False) = 48
        .AllowNonworkingBars = True
        0.Bars.Add("Task:Split").Shortcut = "Task"
    End With
    .Columns.Add("Tasks")
    With .Items
        h = .AddItem("Task 1")
        .ItemNonworkingUnits(h,False) = "weekday(value) in (0,6)"
        .AddBar(h,"",#1/2/2001#,#1/5/2001#,"A","holyday")
        .ItemBar(h,"A",EXG2ANTTLib.ItemBarPropertyEnum.exBarTreatAsNonworking) = True
        .AddBar(h,"Task",#1/5/2001#,#1/12/2001#,"Z")
        .ItemBar(h,"Z",EXG2ANTTLib.ItemBarPropertyEnum.exBarKeepWorkingCount) = True
    End With
    .EndUpdate()
End With
```

C++

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutFirstVisibleDate("1/1/2001");
   var_Chart->PutLevelCount(2);
   var_Chart->PutPaneWidth(VARIANT_FALSE,48);
   var_Chart->PutAllowNonworkingBars(VARIANT_TRUE);
   0var_Chart->GetBars()->Add(L"Task:Split")->PutShortcut(L"Task");
spG2antt1->GetColumns()->Add(L"Tasks");
EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems();
   long h = var_Items->AddItem("Task 1");
   var_Items->PutItemNonworkingUnits(h,VARIANT_FALSE,L"weekday(value) in (0,6)");
   var_Items->AddBar(h,"","1/2/2001","1/5/2001","A","holyday");
   var_Items->PutItemBar(h,"A",EXG2ANTTLib::exBarTreatAsNonworking,VARIANT_TRUE);
   var_Items->AddBar(h,"Task","1/5/2001","1/12/2001","Z",vtMissing);
   var_Items->PutItemBar(h,"Z",EXG2ANTTLib::exBarKeepWorkingCount,VARIANT_TRUE);
spG2antt1->EndUpdate();
```

C#

```
exg2antt1.BeginUpdate();
exontrol.EXG2ANTTLib.Chart var_Chart = exg2antt1.Chart;
   var_Chart.FirstVisibleDate = Convert.ToDateTime("1/1/2001");
   var_Chart.LevelCount = 2;
   var_Chart.set_PaneWidth(false,48);
   var_Chart.AllowNonworkingBars = true;
   0var_Chart.Bars.Add("Task:Split").Shortcut = "Task";
exg2antt1.Columns.Add("Tasks");
exontrol.EXG2ANTTLib.Items var_Items = exg2antt1.Items;
```

```
   int h = var_Items.AddItem("Task 1");
   var_Items.set_ItemNonworkingUnits(h,false,"weekday(value) in (0,6)");

var_Items.AddBar(h,"",Convert.ToDateTime("1/2/2001"),Convert.ToDateTime("1/5/2001"),"A



var_Items.set_ItemBar(h,"A",exontrol.EXG2ANTTLib.ItemBarPropertyEnum.exBarTreatAsNonv



var_Items.AddBar(h,"Task",Convert.ToDateTime("1/5/2001"),Convert.ToDateTime("1/12/200



var_Items.set_ItemBar(h,"Z",exontrol.EXG2ANTTLib.ItemBarPropertyEnum.exBarKeepWorkin

exg2antt1.EndUpdate();
```

C# for /COM

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.FirstVisibleDate = Convert.ToDateTime("1/1/2001");
   var_Chart.LevelCount = 2;
   var_Chart.set_PaneWidth(false,48);
   var_Chart.AllowNonworkingBars = true;
   0var_Chart.Bars.Add("Task:Split").Shortcut = "Task";
axG2antt1.Columns.Add("Tasks");
EXG2ANTTLib.Items var_Items = axG2antt1.Items;
   int h = var_Items.AddItem("Task 1");
   var_Items.set_ItemNonworkingUnits(h,false,"weekday(value) in (0,6)");

var_Items.AddBar(h,"",Convert.ToDateTime("1/2/2001"),Convert.ToDateTime("1/5/2001"),"A



var_Items.set_ItemBar(h,"A",EXG2ANTTLib.ItemBarPropertyEnum.exBarTreatAsNonworking,t



var_Items.AddBar(h,"Task",Convert.ToDateTime("1/5/2001"),Convert.ToDateTime("1/12/200
```

```
var_Items.set_ItemBar(h,"Z",EXG2ANTTLib.ItemBarPropertyEnum.exBarKeepWorkingCount,t

axG2antt1.EndUpdate();
```

Delphi

```
with AxG2antt1 do
begin
   BeginUpdate();
   with Chart do
   begin
      FirstVisibleDate := '1/1/2001';
      LevelCount := 2;
      PaneWidth[False] := 48;
      AllowNonworkingBars := True;
      0Bars.Add('Task:Split').Shortcut := 'Task';
   end;
   Columns.Add('Tasks');
   with Items do
   begin
      h := AddItem('Task 1');
      ItemNonworkingUnits[h,TObject(False)] := 'weekday(value) in (0,6)';
      AddBar(h,'','1/2/2001','1/5/2001','A','holyday');
      ItemBar[h,'A',EXG2ANTTLib.ItemBarPropertyEnum.exBarTreatAsNonworking] :=
TObject(True);
      AddBar(h,'Task','1/5/2001','1/12/2001','Z',Nil);
      ItemBar[h,'Z',EXG2ANTTLib.ItemBarPropertyEnum.exBarKeepWorkingCount] :=
TObject(True);
   end;
   EndUpdate();
end
```

VFP

```
with thisform.G2antt1
   .BeginUpdate
```

```
  with .Chart
    .FirstVisibleDate = {^2001-1-1}
    .LevelCount = 2
    .PaneWidth(.F.) = 48
    .AllowNonworkingBars = .T.
    0.Bars.Add("Task:Split").Shortcut = "Task"
  endwith
  .Columns.Add("Tasks")
  with .Items
    h = .AddItem("Task 1")
    .ItemNonworkingUnits(h,.F.) = "weekday(value) in (0,6)"
    .AddBar(h,"",{^2001-1-2},{^2001-1-5},"A","holyday")
    .ItemBar(h,"A",38) = .T.
    .AddBar(h,"Task",{^2001-1-5},{^2001-1-12},"Z")
    .ItemBar(h,"Z",20) = .T.
  endwith
  .EndUpdate
endwith
```

property Chart.AllowOverviewZoom as OverviewZoomEnum

Gets or sets a value that indicates whether the user can zoom and scale the chart at runtime.

Type	Description
OverviewZoomEnum	An OverviewZoomEnum expression that specifies when the control displays the zooming scale.

By default, the AllowOverviewZoom property is exZoomOnRClick. The zooming scale displays the list of visible units. **A visible unit is an unit whose Label property is not empty. So, the Label property indicates the zooming units in the zoom scale. If you plan to use zooming in your chart please review each Label and LabelToolTip properties.** Once the user selects a new time scale unit in the overview zoom area, the control fires the OverviewZoom event.

- If the AllowOverviewZoom property is exZoomOnRClick the zooming scale is shown only if the user right clicks the overview area. The zooming scale stays visible while the user keeps the right button down. Once the user releases the mouse over a new unit, the chart gets scaled by that unit. During this, ESC key cancels the zooming operation and restores the chart.
- If the AllowOverviewZoom property is exAlwaysZoom the zooming scale is displayed in the right side of the overview area. This way, the available (visible) units are always displays on the right side of the overview area. Clicking any of these units makes the control to scale the chart to specified unit. The OverviewZoomUnit property indicates the width in pixels of the zooming unit.
- If the AllowOverviewZoom property is exDisableZoom the user can't zoom or scale the chart at runtime using the overview area.

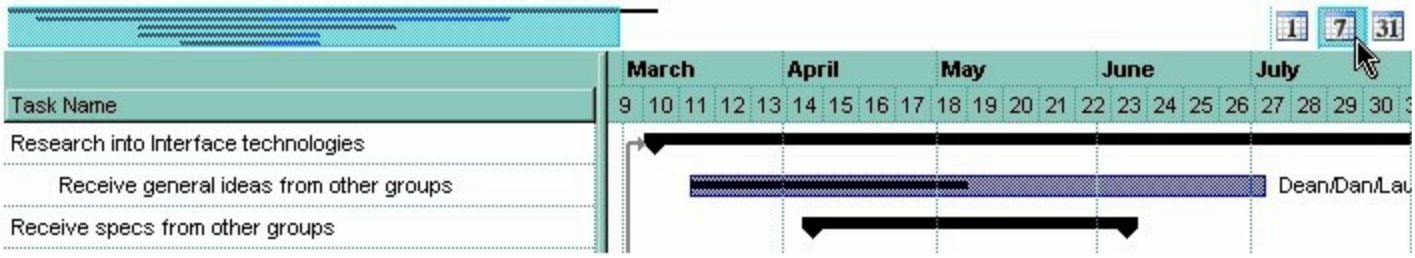
The zooming scale may be displayed on the overview area only if:

- AllowOverviewZoom property is not exDisableZoom
- OverviewVisible property is True
- OverviewHeight property is greater than 0
- there are at least two visible units, that has the Label property not empty.

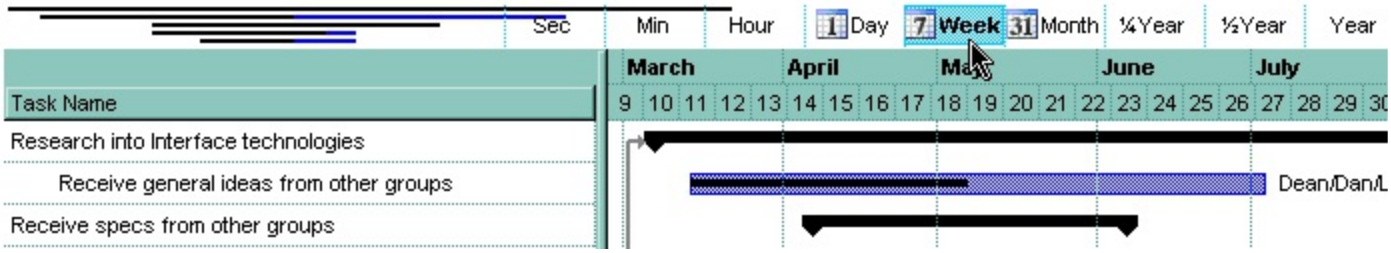
Use the OverviewVisible property to show or hide the control's overview area. The OverviewZoomCaption property indicates the caption being displayed in each zooming unit. The OverviewZoomUnit property indicates the width in pixels of the zooming unit. The LabelToolTip retrieves or sets a value that indicates the predefined format of the level's tooltip for a specified unit. Use the Zoom method to programmatically zoom and scale the chart. Use the UnitScale property to change the unit of the lowest level.

The following picture shows the zooming scale on the overview area [exAlwaysZoom] (

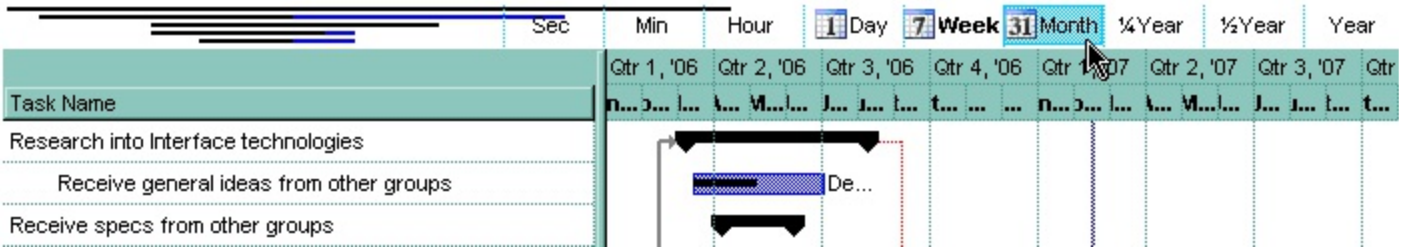
you can click the 1, 7 or 31, and the chart is scaled to days, weeks or moths):



*The following picture shows the control when the user **right** clicks the overview area (as the chart displays weeks) [exZoomOnRClick]:*



*The following picture shows the control while the user drags the cursor to the Month while keeping the **right** button (as the chart displays months):*



property Chart.AllowResizeChart as ResizeChartEnum

Specifies whether the user can enlarge (zoom-in,zoom-out) or resize the chart using the control's header, middle mouse button.

Type	Description
ResizeChartEnum	A ResizeChartEnum expression that indicates the way user can resize or enlarge the control's chart.

By default, the AllowResizeChart property is exDisableResizeChart, so the user is not able to perform any enlargement or zooming using the control's header or middle mouse button. The AllowResizeChart property allows the user to resize or enlarge the chart at runtime. The ChartStartChanging/ChartEndChaning events are fired to notify your application that the user starts or ends resizing/enlarging the chart. Here's a short presentation of how the resizing/enlarging could work. The user can resize the chart by drag and drop the left or right resize-margins of the overview-selection, while the Background(exOverviewSelResize) property is not zero.

When the user resizes or enlarges the chart at runtime, the following properties may be changed:

- UnitWidth, indicates the width of the base time-scale, in pixels. The MinUnitWidth property indicates the minimum value for the UnitWidth property when resizing/enlarging is performed. The MaxUnitWidth property indicates the maximum value for the UnitWidth property when resizing/enlarging is performed.
- UnitScale, indicates the time-scale unit for the chart. This property is changed ONLY if the AllowResizeChart property includes the exAllowChangeUnitScale flag.
- FirstVisibleDate, indicates the first visible date in the chart.

Here's some cases of the AllowResizeChart property values:

- **exAllowResizeChartHeader**, the user can resize the chart, by increasing or decreasing the chart's unit width by dragging the chart's header.
- **exAllowResizeChartHeader + exAllowChangeUnitScale**, the user can zoom-in or zoom-out the chart, and the UnitScale property is changed. For instance, if the control's UnitWidth reaches the MinUnitWidth, the UnitScale property is changed to the next time-unit available. If the UnitWidth reaches the MaxUnitWidth, the UnitScale property is changed to the prev time-unit available.
- **exAllowResizeChartHeader + exAllowResizeChartMiddle + exAllowChangeUnitScale**, the user can zoom-in/zoom-out the chart using the control's header as well as clicking the middle mouse button of the control.

In conclusion, if the AllowResizeChart property includes no exAllowChangeUnitScale flag, the UnitScale property of the chart is not changed while resizing is performed, so actually

only the UnitWidth and FirstVisibleDate may be changed. If the AllowResizeChart property includes exAllowChangeUnitScale flag, all of the mentioned properties may be changed. The Label property of the Chart indicates the available time-scale units when zoom-in zoom-out is performed. For instance, if the Label(exHour) property is empty, the exHour time-scale unit is not available, else If the Label(exHour) is not empty, the exHour time scale unit is available. In other words, when zoom-in/zoom-out is performed the control's chart can be zoomed to exHour only if the Chart.Label(exHour) property is not empty.

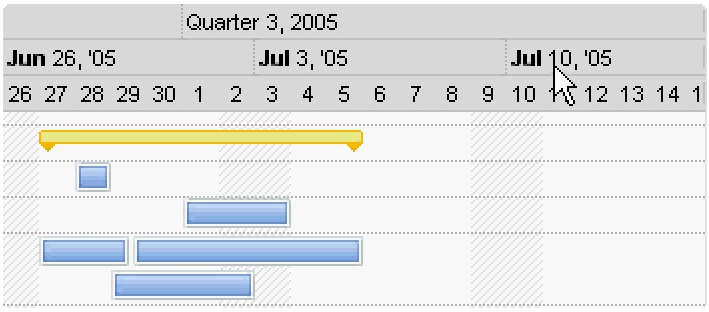
property Chart.AllowResizeInsideZoom as Boolean

Specifies whether the user can resize the inside zoom unit.

Type	Description
Boolean	A boolean expression that specifies whether the user can magnify a time unit, by resizing it in the chart's base level.

By default, the AllowResizeInsideZoom property is True. If the AllowResizeInsideZoom property is True, the resizing cursor is shown once it hovers the base level area in the chart area. The inside zoom units are shown ONLY if the AllowInsideZoom property is True. The inside zoom feature allows displaying portions of the chart with different time scale units. For instance, you can display the bars on hours, while the chart still displays days. Use the CondInsideZoom property to specify the dates that can be magnified by resizing the chart's base level. Use the DefaultInsideZoomFormat property to specify the format of the dates being magnified. The InsideZoomOnDblClick property specifies whether the user can magnify dates by double clicking them in the chart's base level. The InsideZooms property retrieves the collection of inside zoom units. The SplitBaseLevel property specifies whether the base level is expanded once the chart displays inside zoom units.

The following animation shows resizing the bars, using the inside zoom feature:



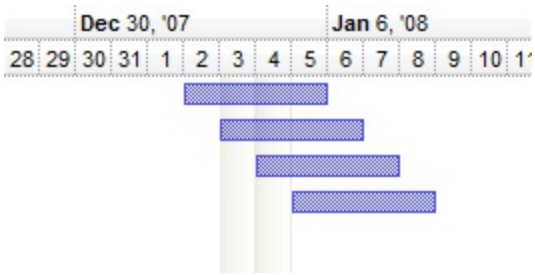
property Chart.AllowSelectDate as SelectDateEnum

Specifies whether the user selects dates at runtime.

Type	Description
SelectDateEnum	A SelectDateEnum expression that specifies whether the user can select dates at runtime, by clicking the chart's header.

By default, the AllowSelectDate property is True, which means that the user can select dates by clicking the chart's header. If the chart displays the histogram, it can select new dates by clicking the histogram. The selected dates are shown using the MarkSelectDateColor property, and it is different than chart's background color, BackColor property. The SelectedDates property can be used to retrieve all selected dates, or to select a collection of dates. Use the SelectDate property to select dates programmatically, no matter of AllowSelectDate property . The MarkTodayColor property specifies the color to mark the today date. Use the LevelFromPoint property to get the index of the level from the cursor. Use the DateFromPoint property to retrieve the date from the cursor. The ChartEndChanging(exSelectDate) event notifies your application when the user selects a new date by clicking the header of the chart. You can show the selected dates on the overview-map part of the control by setting the OverviewShowSelectDates property on True.

The following screen shot shows the selected dates (*Dec 2 and Dec 4*) being colored with this EBN file:



The following VB sample shows how you can use skin / ebn files to display the selected dates:

```
With G2antt1
   .BeginUpdate
   With .VisualAppearance
      .Add 1,"c:\exontrol\images\normal.ebn"
   End With
   With .Chart
      .FirstVisibleDate = #1/1/2008#
```

```
    .MarkTodayColor = .BackColor
    .LevelCount = 2
    .MarkSelectDateColor = 0x1000000
    .SelectLevel = 1
    .SelectDate(#1/3/2008#) = True
    .SelectDate(#1/4/2008#) = True
  End With
  .Columns.Add "Default"
  With .Items
    .AddBar .AddItem("Item 1"),"Task",#1/2/2008#,#1/6/2008#
    .AddBar .AddItem("Item 2"),"Task",#1/3/2008#,#1/7/2008#
    .AddBar .AddItem("Item 3"),"Task",#1/4/2008#,#1/8/2008#
    .AddBar .AddItem("Item 4"),"Task",#1/5/2008#,#1/9/2008#
  End With
  .EndUpdate
End With
```

The following VB.NET sample shows how you can use skin / ebn files to display the selected dates:

```
With AxG2antt1
  .BeginUpdate
  With .VisualAppearance
    .Add 1,"c:\exontrol\images\normal.ebn"
  End With
  With .Chart
    .FirstVisibleDate = #1/1/2008#
    .MarkTodayColor = .BackColor
    .LevelCount = 2
    .MarkSelectDateColor = &H1000000
    .SelectLevel = 1
    .SelectDate(#1/3/2008#) = True
    .SelectDate(#1/4/2008#) = True
  End With
  .Columns.Add "Default"
  With .Items
    .AddBar .AddItem("Item 1"),"Task",#1/2/2008#,#1/6/2008#
    .AddBar .AddItem("Item 2"),"Task",#1/3/2008#,#1/7/2008#
```

```
    .AddBar .AddItem("Item 3"),"Task",#1/4/2008#,#1/8/2008#
    .AddBar .AddItem("Item 4"),"Task",#1/5/2008#,#1/9/2008#
  End With
  .EndUpdate
End With
```

The following C++ sample shows how you can use skin / ebn files to display the selected dates:

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
EXG2ANTTLib::IAppearancePtr var_Appearance = spG2antt1->GetVisualAppearance();
   var_Appearance->Add(1,"c:\\exontrol\\images\\normal.ebn");
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutFirstVisibleDate("1/1/2008");
   var_Chart->PutMarkTodayColor(var_Chart->GetBackColor());
   var_Chart->PutLevelCount(2);
   var_Chart->PutMarkSelectDateColor(0x1000000);
   var_Chart->PutSelectLevel(1);
   var_Chart->PutSelectDate("1/3/2008",VARIANT_TRUE);
   var_Chart->PutSelectDate("1/4/2008",VARIANT_TRUE);
spG2antt1->GetColumns()->Add(L"Default");
EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems();
   var_Items->AddBar(var_Items->AddItem("Item
1"),"Task","1/2/2008","1/6/2008",vtMissing,vtMissing);
   var_Items->AddBar(var_Items->AddItem("Item
2"),"Task","1/3/2008","1/7/2008",vtMissing,vtMissing);
   var_Items->AddBar(var_Items->AddItem("Item
3"),"Task","1/4/2008","1/8/2008",vtMissing,vtMissing);
   var_Items->AddBar(var_Items->AddItem("Item
```

```
4"),"Task","1/5/2008","1/9/2008",vtMissing,vtMissing);
spG2antt1->EndUpdate();
```

The following C# sample shows how you can use skin / ebn files to display the selected dates:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Appearance var_Appearance = axG2antt1.VisualAppearance;
   var_Appearance.Add(1,"c:\\exontrol\\images\\normal.ebn");
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.FirstVisibleDate = "1/1/2008";
   var_Chart.MarkTodayColor = var_Chart.BackColor;
   var_Chart.LevelCount = 2;
   var_Chart.MarkSelectDateColor = 0x1000000;
   var_Chart.SelectLevel = 1;
   var_Chart.set_SelectDate("1/3/2008",true);
   var_Chart.set_SelectDate("1/4/2008",true);
axG2antt1.Columns.Add("Default");
EXG2ANTTLib.Items var_Items = axG2antt1.Items;
   var_Items.AddBar(var_Items.AddItem("Item 1"),"Task","1/2/2008","1/6/2008",null,null);
   var_Items.AddBar(var_Items.AddItem("Item 2"),"Task","1/3/2008","1/7/2008",null,null);
   var_Items.AddBar(var_Items.AddItem("Item 3"),"Task","1/4/2008","1/8/2008",null,null);
   var_Items.AddBar(var_Items.AddItem("Item 4"),"Task","1/5/2008","1/9/2008",null,null);
axG2antt1.EndUpdate();
```

The following VFP sample shows how you can use skin / ebn files to display the selected dates:

```
with thisform.G2antt1
   .BeginUpdate
   with .VisualAppearance
      .Add(1,"c:\exontrol\images\normal.ebn")
   endwith
   with .Chart
      .FirstVisibleDate = {^2008-1-1}
      .MarkTodayColor = .BackColor
      .LevelCount = 2
      .MarkSelectDateColor = 0x1000000
```

```
    .SelectLevel = 1
    .SelectDate({^2008-1-3}) = .T.
    .SelectDate({^2008-1-4}) = .T.
  endwith
  .Columns.Add("Default")
  with .Items
    .AddBar(.AddItem("Item 1"),"Task",{^2008-1-2},{^2008-1-6})
    .AddBar(.AddItem("Item 2"),"Task",{^2008-1-3},{^2008-1-7})
    .AddBar(.AddItem("Item 3"),"Task",{^2008-1-4},{^2008-1-8})
    .AddBar(.AddItem("Item 4"),"Task",{^2008-1-5},{^2008-1-9})
  endwith
  .EndUpdate
endwith
```

property Chart.AllowSelectObjects as SelectObjectsEnum

Sets or gets a value that indicates whether the user can select objects in the chart.

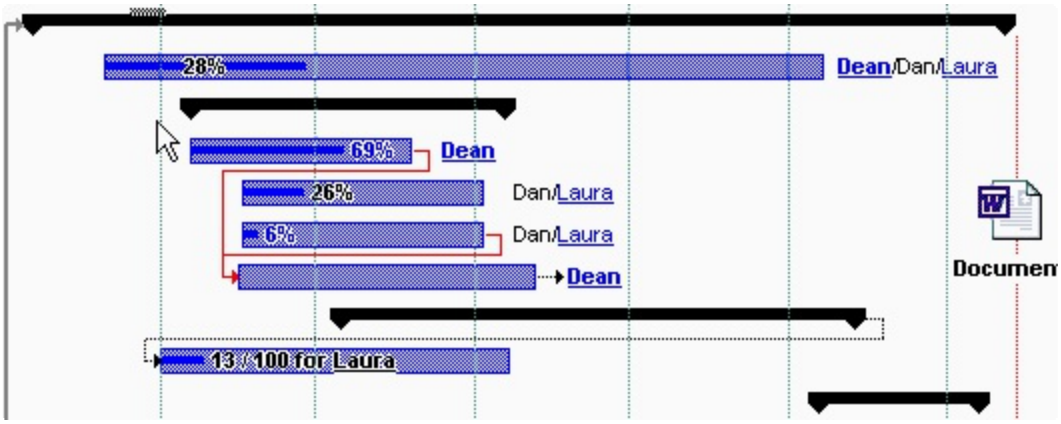
Type	Description
SelectObjectsEnum	A combination of SelectObjectsEnum values that indicates the objects to be selected in the chart.

By default, the AllowSelectObjects property is exSelectObjects. The AllowSelectObjects property allows users to select at runtime the bars and links in the chart area. Use the AllowSelectObjects property to disable selecting bars and links in the chart area using the mouse. For instance, if the AllowSelectObjects property is

- exNoSelectObjects, the selection of objects in the chart is disabled.
- exSelectBarsOnly the user can select bars only.
- exSelectLinksOnly the user can select links only.
- exSelectObjects the user can select links and bars as well.

Also, if you want to enable selecting a single bar in your chart, the exSelectSingleObject value should be added to one of these: exSelectBarsOnly, exSelectLinksOnly and exSelectObjects. So, for instance, you need to select a single bar, the AllowSelectObjects property should be exSelectBarsOnly Or exSelectSingleObject, and if you need to select only a single link, the AllowSelectObjects property should be exSelectLinksOnly Or exSelectSingleObject

The ChartSelectionChanged event is fired when the selection in the chart is changed. Use the SelectedObject property to retrieve a collection of selected bars or/and links. You can use the selection to move all selected objects as you would move them individually. The user can select a bar or a link by clicking it. The user can use the CTRL key to select or unselect the bar or the link from the cursor. Also, the user can right click the chart area, to start selecting the bars and links that intersect the dragging rectangle. The SelBackColor property specifies the color to draw the frame around the selected bar or link. Use the ItemBar(exBarSelected) property to select or unselect programmatically a bar. Use the Link(exLinkSelected) property to select or unselect programmatically a link. Use the RemoveSelection property to remove objects in the chart's selection.


28%
Dean/Dan/Laura
69%
Dean
26%
Dan/Laura
6%
Dan/Laura
Dean
13 / 100 for Laura

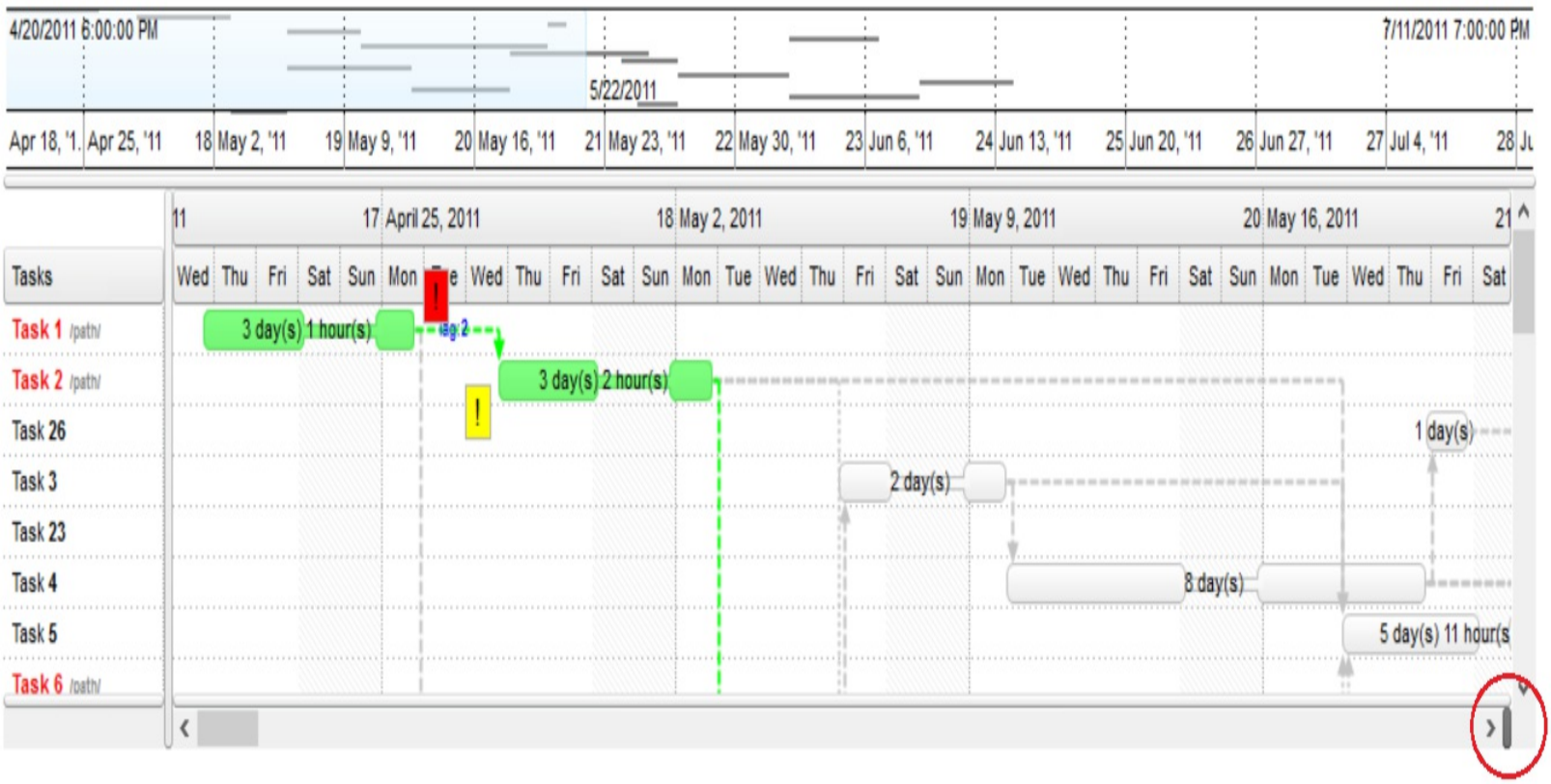
property Chart.AllowSplitPane as AllowSplitPaneEnum

Specifies whether the chart panel supports splitting.

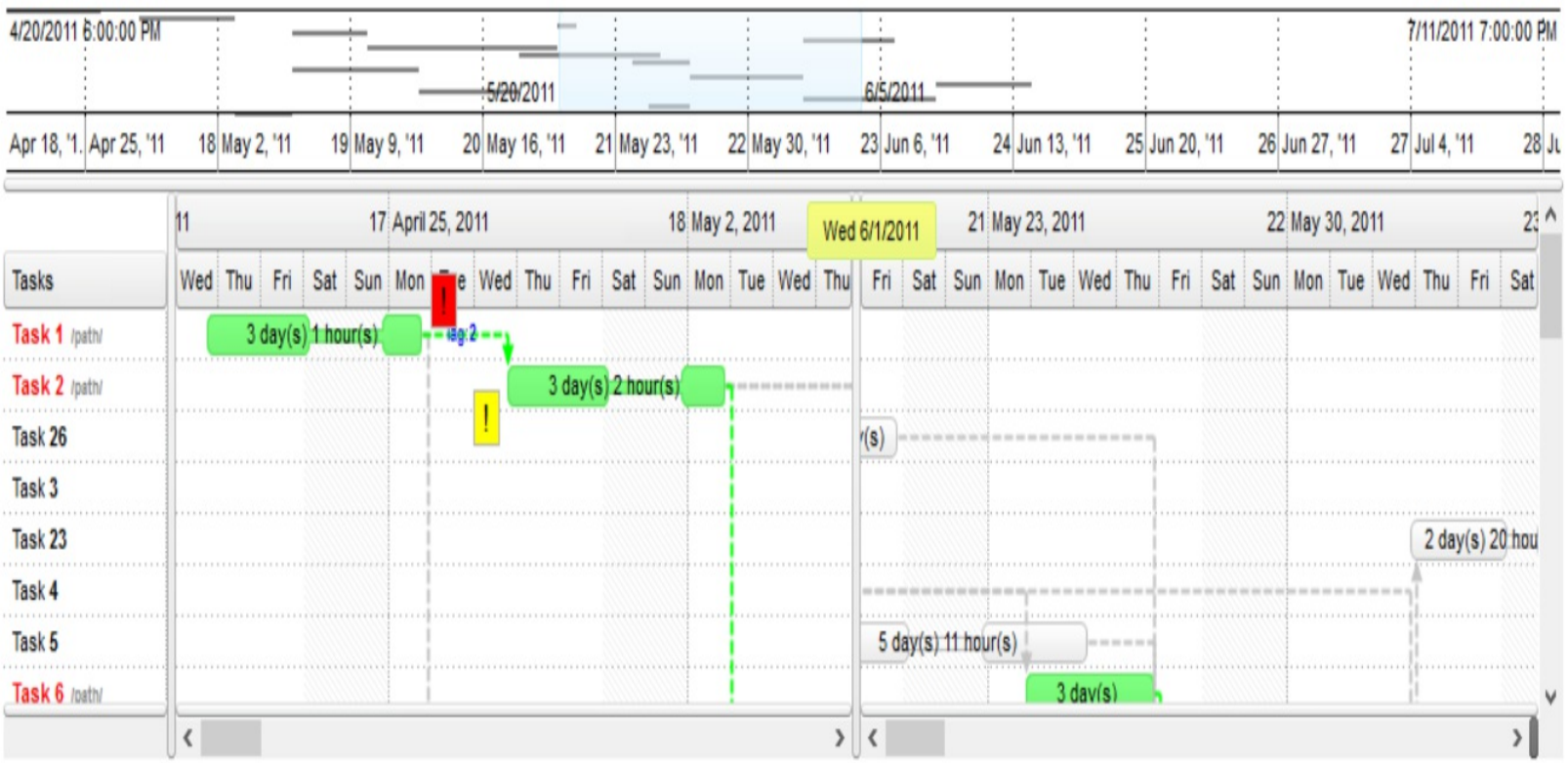
Type	Description
AllowSplitPaneEnum	An AllowSplitPaneEnum expression that specifies the number of split panels, the user can divide the control's chart.

By default, the AllowSplitPane property is exNoSplitPane, which specifies that user can't split the control's chart. The AllowSplitPane property specifies whether the chart panel supports splitting. Once the AllowSplitPane property is set, the user can click the lower-right split bar, and drag to a new position to add a new split to the current chart. The Background(exCSplitBar) property specifies the visual appearance of the chart's split bar. The exDisableSplitPane flag of OnResizeControl property specifies whether the user can drag the split bar at runtime. The SplitPaneWidth property specifies the width of split panels, separated by comma. The ChartStartChanging(exSplitPaneChange) / ChartEndChanging(exSplitPaneChange) events notify that the user splits/resizes the chart's panel into multiple-views.

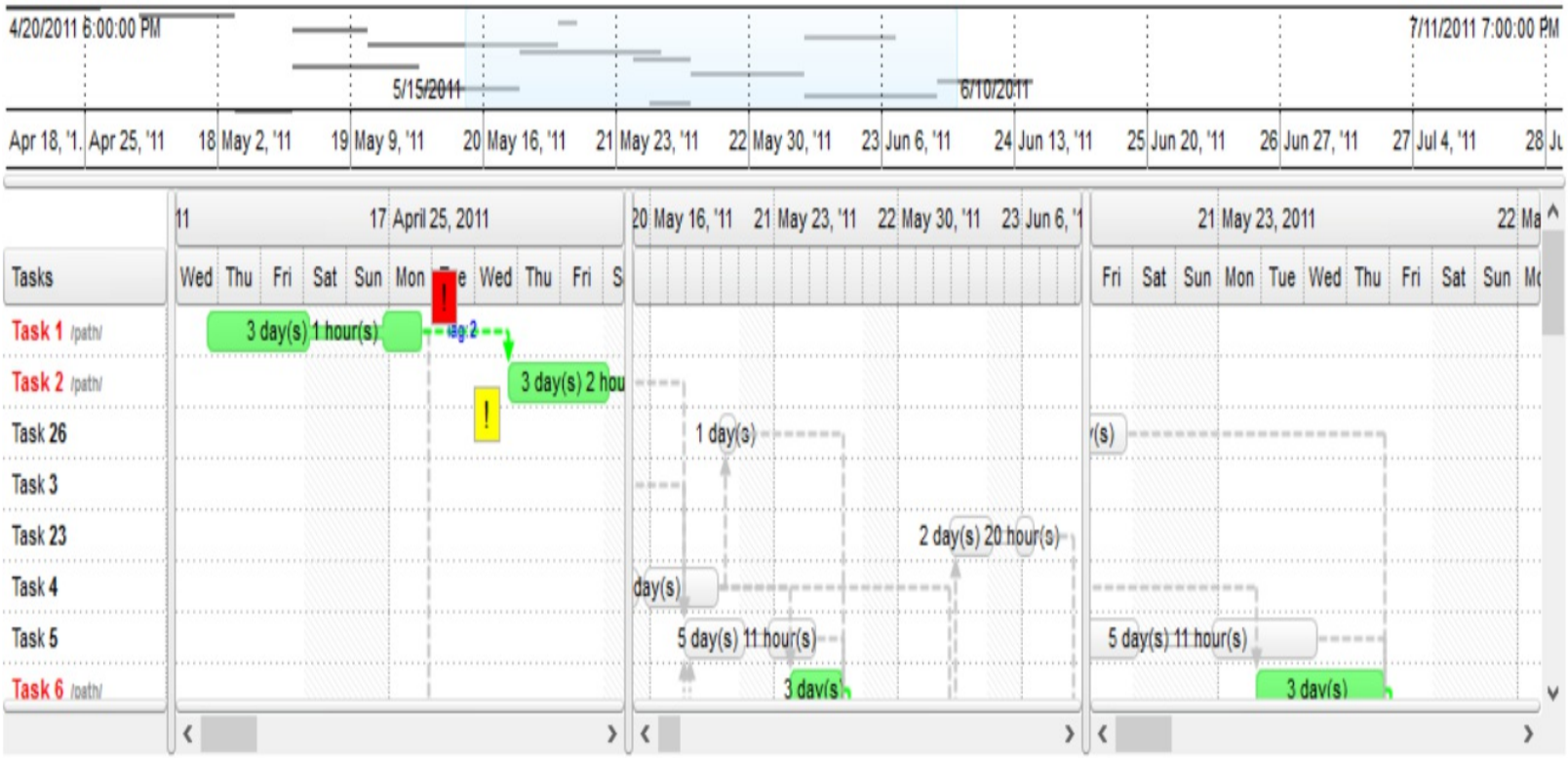
The following screen shot shows the control's split bar:



The following screen shot shows the chart divided in two parts:



The following screen shot shows the chart divided in three parts:



property Chart.AllowUndoRedo as Boolean

Enables or disables the Undo/Redo feature.

Type	Description
Boolean	A Boolean expression that specifies whether the Undo/Redo operations are enabled or disabled.

By default, the AllowUndoRedo property is FalseThe Undo and Redo features let you remove or repeat single or multiple actions, but all actions must be undone or redone in the order you did or undid them you cant skip actions. For example, if you change the value of three cells in an item and then decide you want to undo the first change you made, you must undo all three changes. To undo an action you need to press Ctrl+Z, while for to redo something you've undone, press Ctrl+Y. The CanUndo property retrieves a value that indicates whether the chart may perform the last Undo operation. The CanRedo property retrieves a value that specifies whether the chart can execute the next operation in the chart's Redo queue. Call the Undo method to Undo the last chart operation. The Redo redoes the next action in the chart's redo queue. The UndoRedoQueueLength property gets or sets the maximum number of Undo/Redo actions that may be stored to the chart's queue, or in other words how many operations the chart's Undo/Redo manager may store.

The records of the Undo/Redo queue may contain actions in the following format:

- "**AddBar**;ITEMINDEX;KEY", indicates that a new bar has been created
- "**RemoveBar**;ITEMINDEX;KEY", indicates that a bar has been removed
- "**MoveBar**;ITEMINDEX;KEY", indicates that a bar has been moved or resized
- "**PercentChange**;ITEMINDEX;KEY", indicates that the bar's percent has been changed
- "**UpdateBar**;ITEMINDEX;KEY", indicates that one or more properties of the bar has been updated (ItemBar property, this operation can be added only using the StartUpdateBar / EndUpdateBar methods)
- "**ParentChangeBar**;ITEMINDEX;KEY", indicates that the bar's parent has been changed
- "**GroupBars**;ITEMINDEXA;KEYA;STARTA;ITEMINDEXB;KEYB;STARTB", specifies that two bars has been grouped
- "**UngroupBars**;ITEMINDEXA;KEYA;STARTA;ITEMINDEXB;KEYB;STARTB", specifies that two bars has been ungrouped
- "**DefineSummaryBars**;SUMMARYITEMINDEX;SUMMARYKEY;ITEMINDEX;KEY", indicates that a bar has been defined as a child of a summary bar
- "**UndefineSummaryBars**;SUMMARYITEMINDEX;SUMMARYKEY;ITEMINDEX;KEY", indicates that a bar has been removed from the summary bar's children
- "**AddLink**;KEY", indicates that a new link has been created
- "**RemoveLink**;KEY", indicates that a link has been removed

- "**UpdateLink**;KEY", specifies that one of more properties of the link has been updated (Link property, this operation can be added only using the StartUpdateLink / EndUpdateLink methods)

The records of the Undo/Redo queue may contain actions in the following format (available starting from 23.0):

- "**AddItem**;ITEMINDEX", indicates that a new item has been created
- "**RemoveItem**;ITEMINDEX", indicates that an item has been removed
- "**ChangeItemPos**;ITEMINDEX", indicates that an item changes its position or / and parent
- "**ChangeCellValue**;ITEMINDEX;CELLINDEX", indicates that the cell's value has been changed
- "**ChangeCellState**;ITEMINDEX;CELLINDEX", indicates that the cell's state has been changed

Also, the Undo/Redo queue may include:

- "**StartBlock**", specifies that a block of operations begins
- "**EndBlock**", specifies that a block of operations ends

The ChartStartChanging(exUndo/exRedo) / ChartEndChanging(exUndo/exRedo) event notifies your application whenever an Undo/Redo operation is performed. The UndoListAction property lists the Undo actions that can be performed in the chart. The RedoListAction property lists the Redo actions that can be performed in the chart. Use the UndoRemoveAction method to remove the last actions from the undo queue. The RedoRemoveAction method removes the first action to be performed if the Redo method is invoked.

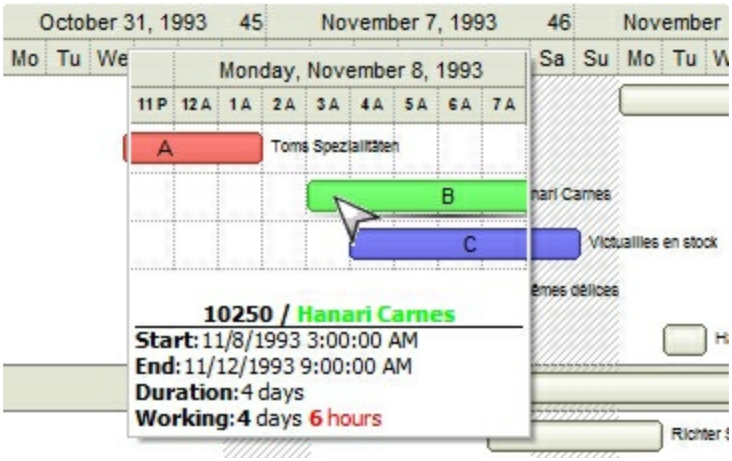
property Chart.AllowZoomOnFly as ZoomOnFlyEnum

Magnifies the bar from the cursor, when the user presses the CTRL / SHIFT key combination.

Type	Description
ZoomOnFlyEnum	A ZoomOnFlyEnum expression, or combination of the giving flags, that specifies the way the Zoom-OnFly view is displayed. For instance, if the AllowZoomOnFly is exZoomOnFly, the Zoom-OnFly view is show once the user presses the SHIFT + CTRL keys combination on the chart.

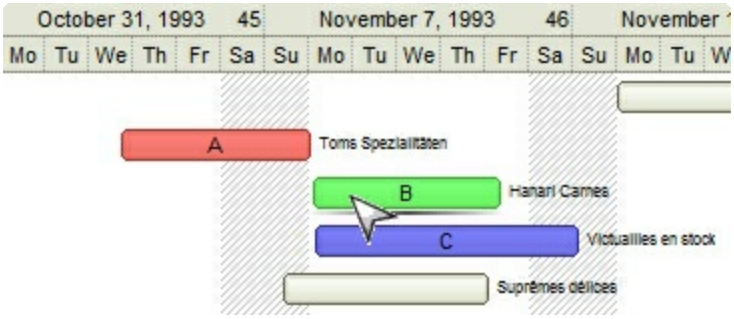
By default, the AllowZoomOnFly property is 0, exNoZoomOnFly. In other words, the Zoom-OnFly view is never displayed by default. The Zoom-OnFly view was provided to let you magnify a portion of the chart without affecting the chart's scale. The Zoom-OnFly view displays the item and its neighbors from the cursor, and additional information about the bar from the cursor. The Zoom-OnFly scale is indicating by the chart's ResizeUnitScale property. If the chart's ResizeUnitScale property is not specified, the Zoom-OnFly uses the chart's UnitScale property to indicate the inside scale. The ZoomOnFlyCaption property indicates the HTML caption to be displayed as addition information for the bar/task from the cursor. The BackColorZoomOnFly property indicates the Zoom-OnFly's background color. The Zoom-OnFly view is shown on the chart once the user presses the CTRL +/ SHIFT keys and the the chart is active/focused. The Label property indicates the label to be shown in the Zoom-OnFly view. Set the ZoomOnFlyCaption property on empty, to display no addition information about the bar from the cursor. Use the SelectOnClick property to prevent selecting a row / item when clicking the chart portion of the control.

The following screen shot shows the Zoom-OnFly view, after the user presses the CTRL + SHIFT keys combination (You can notice that the inside scale displays hours, while the master chart displays days):



The following screen shot shows the chart before pressing the CTRL + SHIFT keys (before

showing the Zoom-OnFly view):



This feature could be very useful:

- if you require to display additional information about bars.
- if you need to align different bars based on other bars
- refines moving or sizing the bars based on another scale. For instance, if the main chart displays days, you can have the Zoom-OnFly view displaying hours, so resizing is very easily to be precise.

Click here to watch a movie on how you can resize or move bars if precision is required.

Here's a few samples on how to use the AllowZoomOnFly property:

- **= exZoomOnFly** (24), the Zoom-OnFly view is shown if the cursor hovers any part of the control's chart, and presses the CTRL + SHIFT combination. The view shows the item from the cursor including the bar from the cursor.
- **= exZoomOnFly + exZoomOnFlyBarsOnly** (56), the Zoom-OnFly view is shown if the cursor hovers any BAR of the control's chart, and presses the CTRL + SHIFT combination. The view shows the item from the cursor including the bar from the cursor, and shows nothing if there is no bar at the cursor position.
- **= exZoomOnFlyShift + exZoomOnFly** (25), same as the exZoomOnFly excepts that the view can be shown if the SHIFT is pressed, not requiring pressing the CTRL + SHIFT
- **= exZoomOnFlyCtrl + exZoomOnFly** (26), same as the exZoomOnFly excepts that the view can be shown if the CTRL is pressed, not requiring pressing the CTRL + SHIFT
- **= exZoomOnFly + exZoomOnFlyIncludeNeighborItems** (280), same as the exZoomOnFly, excepts that the view displays the previously visible item, and the next visible item, if they exists. Use this option to display more than one item in the view for better alignment of the bars based on their neighbors.
- **= exZoomOnFly + exZoomOnFlyIncludeSelectedItems** (792), same as exZoomOnFly + exZoomOnFlyIncludeNeighborItems, excepts that the view displays the previously selected item, and next selected item. Use this option to include the next and previously selected items for a better comparing between selection and the item from the cursor.

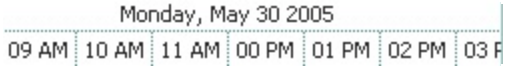
- **= exZoomOnFlyShift + exZoomOnFly + exZoomOnFlyIncludeSelectedItems** (793), same as exZoomOnFly + exZoomOnFlyIncludeSelectedItems, excepts that the view can be shown if the SHIFT is pressed, not requiring pressing the CTRL + SHIFT
- **= exAllowInfoOnFly** (16), the Zoom-OnFly view is shown if the cursor hovers any part of the control's chart, and presses the CTRL + SHIFT combination. The view shows the item from the cursor including the bar from the cursor. The view is closed as soon as the cursor hovers another part, or if the user presses the mouse.
- **= exZoomOnFlyCtrl + exAllowInfoOnFly** (18), same as exAllowInfoOnFly, excepts that the view can be shown if the SHIFT is pressed, not requiring pressing the CTRL + SHIFT
- = **exAllowRefineOnFly** (8), the Zoom-OnFly view is shown only if the user clicks while pressing the CTRL + SHIFT combination.

property Chart.AMPM as String

Specifies the AM and PM indicators.

Type	Description
String	A String expression that indicates the AM PM indicators, separated by space.

By default, the AMPM property is "AM PM". The AMPM property specifies the indicators being displayed when the Label or ToolTip property includes the <%AM/PM%> tag. Use the UnitScale property to change the chart's time unit. Use the MonthNames property to specify the name of the months being displayed in the chart's header. Use the WeekDays property to specify the name for each day in a week. Use the UnitWidth property to specify the width of the time unit



property Chart.BackColor as Color

Retrieves or sets a value that indicates the chart's background color.

Type	Description
Color	A Color expression that indicates the chart's background color.

Use the BackColor property to specify the chart's background color. Use the ForeColor property to specify the chart's foreground color. Use the BackColorLevelHeader property to specify the background color of the chart's header. Use the ForeColorLevelHeader property to specify the foreground color of the chart's header. Use the BackColor property to specify the background color for a specified level. Use the ForeColor property to specify the foreground color for a specified level. Use the ItemBackColor property to change the item's background color. Use the NonworkingDaysColor property the color of the brush to fill the nonworking days area. Use the Picture property to specify the picture being displayed on the chart's area. The OverviewBackColor property specifies the background color of the chart's overview. The HistogramBackColor property is changed to BackColor property as soon as it is changed. Use the SelBackColor property to specify the background color for selected items in the chart area.

The following VB sample changes the chart's background color:

```
With G2antt1.Chart
   .BackColor = RGB(&H80, &H80, &H80)
End With
```

The following C++ sample changes the chart's background color:

```
m_g2antt.GetChart().SetBackColor( RGB(0x80,0x80,0x80) );
```

The following VB.NET sample changes the chart's background color:

```
With AxG2antt1.Chart
   .BackColor = ToUInt32(Color.FromArgb(&H80, &H80, &H80))
End With
```

where the ToUInt32 function converts a Color expression to an OLE_COLOR type:

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
   Dim i As Long
   i = c.R
```

```
    i = i + 256 * c.G
    i = i + 256 * 256 * c.B
    ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample changes the chart's background color:

```
axG2antt1.Chart.BackColor = ToUInt32(Color.FromArgb(0x80, 0x80, 0x80));
```

where the ToUInt32 function converts a Color expression to an OLE_COLOR type:

```
private UInt32 ToUInt32(Color c)
{
    long i;
    i = c.R;
    i = i + 256 * c.G;
    i = i + 256 * 256 * c.B;
    return Convert.ToUInt32(i);
}
```

The following VFP sample changes the chart's background color:

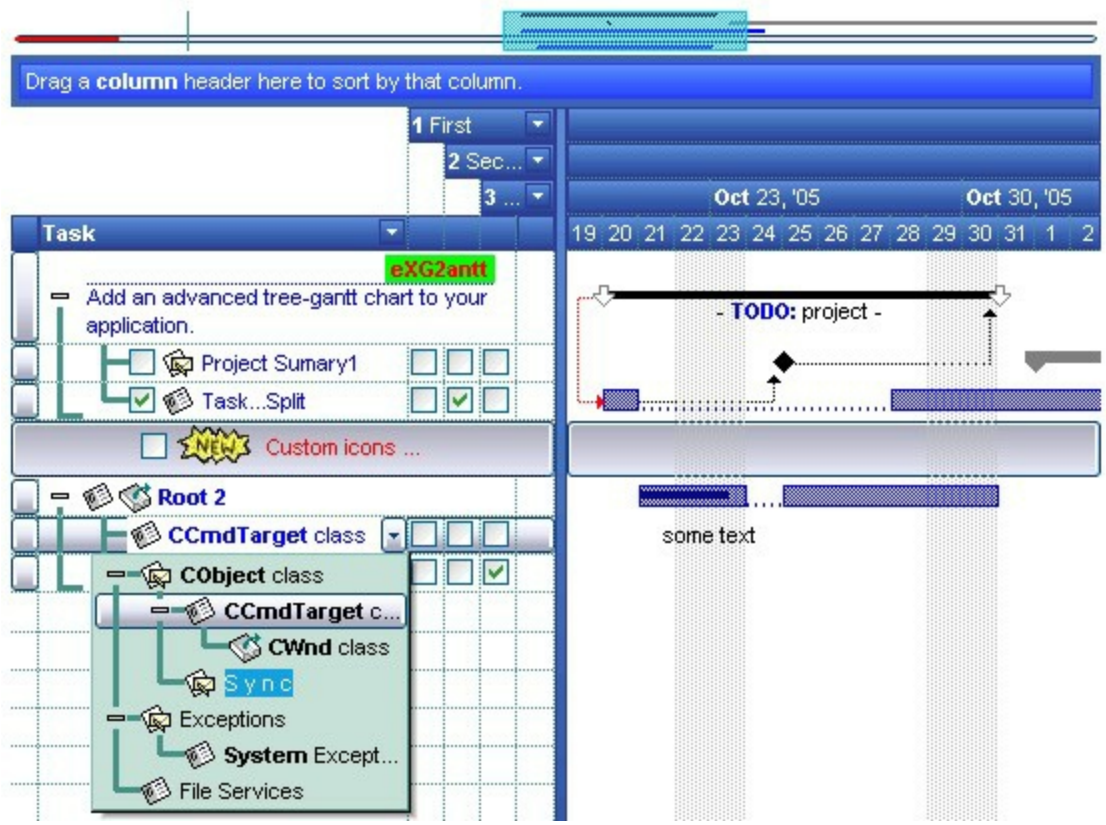
```
With thisform.G2antt1.Chart
    .BackColor = RGB(128, 128, 128)
EndWith
```

property Chart.BackColorLevelHeader as Color

Specifies the background color for the chart's levels.

Type	Description
Color	A Color expression that indicates the background color for the chart's header. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

Use the BackColorLevelHeader property to specify the background color of the chart's header. Use the ForeColorLevelHeader property to specify the foreground color of the chart's header. Use the LevelCount property to specify the number of levels in the chart's header. Use the Level property to access a level. Use the BackColor property to specify the background color for a specified level. Use the ForeColor property to specify the foreground color for a specified level. Use the BackColor property to specify the chart's background color. Use the ForeColor property to specify the chart's foreground color. Use the ItemBackColor property to change the item's background color. Use the NonworkingDaysColor property the color of the brush to fill the nonworking days area. Use the Picture property to specify the picture being displayed on the chart's area.



The following VB sample changes the chart's header background color:

```
With G2antt1.Chart
   .BackColorLevelHeader = RGB(&H80, &H80, &H80)
End With
```

The following C++ sample changes the chart's header background color:

```
m_g2antt.GetChart().SetBackColorLevelHeader( RGB(0x80,0x80,0x80) );
```

The following VB.NET sample changes the chart's header background color:

```
With AxG2antt1.Chart
   .BackColorLevelHeader = ToUInt32(Color.FromArgb(&H80, &H80, &H80))
End With
```

where the ToUInt32 function converts a Color expression to an OLE_COLOR type:

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
   Dim i As Long
   i = c.R
   i = i + 256 * c.G
   i = i + 256 * 256 * c.B
   ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample changes the chart's header background color:

```
axG2antt1.Chart.BackColorLevelHeader = ToUInt32(Color.FromArgb(0x80, 0x80, 0x80));
```

where the ToUInt32 function converts a Color expression to an OLE_COLOR type:

```
private UInt32 ToUInt32(Color c)
{
   long i;
   i = c.R;
   i = i + 256 * c.G;
   i = i + 256 * 256 * c.B;
   return Convert.ToUInt32(i);
}
```

The following VFP sample changes the chart's header background color:

```
With thisform.G2antt1.Chart
    .BackColorLevelHeader = RGB(128, 128, 128)
EndWith
```

property Chart.BackColorZoomOnFly as Color

Specifies the background color for the zoom-on-fly panel.

Type	Description
Color	A Color expression that specifies the background color for the Zoom-OnFly view.

By default, the BackColorZoomOnFly property specifies the Zoom-OnFly's background color. By default, the BackColorZoomOnFly property is white, (0xFFFFFF or RGB(255,255,255)). Use the BackColorZoomOnFly property to specify a different background color for the Zoom-OnFly panel. The BackColor property indicates the chart's background color. The AllowZoomOnFly property specifies whether the user can display the Zoom-OnFly panel, when the cursor hovers the chart part of the area, and the user presses the CTRL + SHIFT keys combination. The ZoomOnFlyCaption property indicates the HTML caption to be displayed as addition information for the bar/task from the cursor.

property Chart.BarFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Variant

Retrieves the bar from point.

Type	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in client coordinates.
Variant	A VARIANT expression that indicates the key of the bar from the cursor.

The BarFromPoint property gets the bar from point. **If the X parameter is -1 and Y parameter is -1 the BarFromPoint property determines the key of the bar from the cursor.** Use the ItemBar property to access properties of the bar from the point. The DateFromPoint property retrieves the date from the cursor, only if the cursor hovers the chart's area. Use the ItemFromPoint property to get the cell/item from the cursor. Use the ColumnFromPoint property to retrieve the column from cursor. Use the LinkFromPoint property to get the link from the point. Use the FormateDate property to format a date. Use the DrawDateTicker property to draw a ticker as cursor hovers the chart's area. Use the LevelFromPoint property to retrieve the index of the level from the cursor. The ItemBar(exBarSelectable) property specifies whether a bar is selectable or not. The BarFromPoint property can returns only selectable bars. By default, all bars are selectable. The NoteFromPoint property retrieves the note/box from the cursor.

The following VB sample displays the handle of the item and the key of the bar from cursor:

```
Private Sub G2antt1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
  Dim h As HITEM, c As Long, hit As HitTestInfoEnum
  With G2antt1
    h = .ItemFromPoint(-1, -1, c, hit)
    If (h <> 0) Then
      Dim k As Variant
      k = .Chart.BarFromPoint(-1, -1)
      If Not IsEmpty(k) Then
        Debug.Print h & " " & k
      End If
```

```
    End If
  End With
End Sub
```

The following Access sample displays the handle of the item and the key of the bar from cursor (please notice that the X and Y parameters of the MouseMove event are declared as Long):

```
Private Sub G2antt1_MouseMove(ByVal Button As Integer, ByVal Shift As Integer, ByVal X As Long, ByVal Y As Long)
  Dim h As HITEM, c As Long, hit As HitTestInfoEnum
  With G2antt1
    h = .ItemFromPoint(-1, -1, c, hit)
    If (h <> 0) Then
      Dim k As Variant
      k = .Chart.BarFromPoint(-1, -1)
      If Not IsEmpty(k) Then
        Debug.Print h & " " & k
      End If
    End If
  End With
End Sub
```

or:

```
Private Sub G2antt1_MouseMove(ByVal Button As Integer, ByVal Shift As Integer, ByVal X As Long, ByVal Y As Long)
  Dim h As Long, c As Long, hit As Long
  With G2antt1
    h = .ItemFromPoint(-1, -1, c, hit)
    If (h <> 0) Then
      Dim k As Variant
      k = .Chart.BarFromPoint(-1, -1)
      If Not IsEmpty(k) Then
        MsgBox h & " " & k
      End If
    End If
  End With
```

```
End Sub
```

The second sample uses the Long type instead HITEM and HitTestInfoEnum which are equivalents.

The following VB sample displays the key of the bar from the cursor:

```
Private Sub G2antt1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    With G2antt1.Chart
        Debug.Print .BarFromPoint(-1, -1)
    End With
End Sub
```

The following VB sample displays the start data of the bar from the point:

```
Private Sub G2antt1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    With G2antt1
        Dim h As HITEM, c As Long, hit As HitTestInfoEnum
        h = .ItemFromPoint(-1, -1, c, hit)
        If Not (h = 0) Then
            Dim k As Variant
            k = .Chart.BarFromPoint(-1, -1)
            If Not IsEmpty(k) Then
                Debug.Print .Items.ItemBar(h, k, exBarStart)
            End If
        End If
    End With
End Sub
```

The following VB sample displays the keys of the bars from the cursor (*in case several bars covers each other*):

```
Private Sub G2antt1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Dim h As HITEM, c As Long, hit As HitTestInfoEnum
    With G2antt1
        h = .ItemFromPoint(-1, -1, c, hit)
```

```
    If (h <> 0) Then
      Dim vKey As Variant, vKeys As New Collection

      vKey = .Chart.BarFromPoint(-1, -1)
      While (Not VarType(vKey) = vbEmpty)
        vKeys.Add vKey
        .Items.ItemBar(h, vKey, exBarSelectable) = False
        vKey = .Chart.BarFromPoint(-1, -1)
      Wend

      If (vKeys.Count > 0) Then
        Debug.Print "Bar(s) from the cursor: "
        Dim v As Variant
        For Each v In vKeys
          .Items.ItemBar(h, v, exBarSelectable) = True
          Debug.Print v
        Next
        Else
        Debug.Print "No bar at the cursor."
      End If

      Set vKeys = Nothing
    End If
  End With
End Sub
```

The following C++ sample displays the start data of the bar from the point:

```
#include "Items.h"
#include "Chart.h"

CString V2Date( VARIANT* pvtValue )
{
  COleVariant vtDate;
  vtDate.ChangeType( VT_BSTR, pvtValue );
  return V_BSTR( &vtDate );
```

```
}

void OnMouseDownG2antt1(short Button, short Shift, long X, long Y)
{
   long c = 0, hit = 0, h = m_g2antt.GetItemFromPoint( -1, -1, &c, &hit );
   if ( h != 0 )
   {
      COleVariant vtKey = m_g2antt.GetChart().GetBarFromPoint( -1, -1 );
      if ( V_VT( &vtKey ) != VT_EMPTY )
      {
         COleVariant vtStart = m_g2antt.GetItems().GetItemBar( h, vtKey, 1 /*exBarStart*/ );
         OutputDebugString( V2Date( &vtStart ) );
      }
   }
}
```

The following VB.NET sample displays the start data of the bar from the point:

```
Private Sub AxG2antt1_MouseDownEvent(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_MouseDownEvent) Handles
AxG2antt1.MouseDownEvent
   With AxG2antt1
      Dim c As Long, hit As EXG2ANTTLib.HitTestInfoEnum, h As Integer =
.get_ItemFromPoint(-1, -1, c, hit)
      If Not (h = 0) Then
         Dim k As Object
         k = .Chart.BarFromPoint(-1, -1)
         If Not k Is Nothing Then
            System.Diagnostics.Debug.WriteLine(.Items.ItemBar(h, k,
EXG2ANTTLib.ItemBarPropertyEnum.exBarStart))
         End If
      End If
   End With
End Sub
```

The following VB.NET /NET Assembly sample displays the bars from the point (*in case several bars covers each other*):

```
Private Sub Exg2antt1_MouseMoveEvent(ByVal sender As System.Object, ByVal Button As
System.Int16, ByVal Shift As System.Int16, ByVal X As System.Int32, ByVal Y As
System.Int32) Handles Exg2antt1.MouseMoveEvent
   With Exg2antt1
      Dim h As Integer = .get_ItemFromPoint(-1, -1)
      If (h <> 0) Then
         Dim vKey As Object = Exg2antt1.Chart.get_BarFromPoint(-1, -1)
         Dim vKeys As List(Of Object) = New List(Of Object)

         While Not vKey Is Nothing
            vKeys.Add(vKey)
            .Items.set_BarSelectable(h, vKey, False)
            vKey = .Chart.get_BarFromPoint(-1, -1)
         End While

         System.Diagnostics.Debug.Print("Bars from point: " + vKeys.Count.ToString())
         Dim v As Object
         For Each v In vKeys
            System.Diagnostics.Debug.Print(v.ToString())
            Exg2antt1.Items.set_BarSelectable(h, v, True)
         Next
      End If
   End With
End Sub
```

The following C# sample displays the start data of the bar from the point:

```
private void axG2antt1_MouseDownEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_MouseDownEvent e)
{
   int c = 0;
   EXG2ANTTLib.HitTestInfoEnum hit = EXG2ANTTLib.HitTestInfoEnum.exHTCell;
   int h = axG2antt1.get_ItemFromPoint(-1, -1, out c, out hit);
   if (h != 0)
   {
      object k = axG2antt1.Chart.get_BarFromPoint(-1, -1);
      if (k != null)
```

```
        System.Diagnostics.Debug.WriteLine( axG2antt1.Items.get_ItemBar( h, k,
EXG2ANTTLib.ItemBarPropertyEnum.exBarStart ) );
    }
}
```

The following C# /NET Assembly sample displays the bars from the point (*in case several bars covers each other*):

```
private void exg2antt1_MouseMoveEvent(object sender, short Button, short Shift, int X,
int Y)
{
    int h = exg2antt1.get_ItemFromPoint(-1, -1);
    if (h != 0)
    {
        object vKey = exg2antt1.Chart.get_BarFromPoint(-1, -1);
        List<object> vKeys = new List<object>();
        while (vKey != null)
        {
            vKeys.Add(vKey);
            exg2antt1.Items.set_BarSelectable(h, vKey, false);
            vKey = exg2antt1.Chart.get_BarFromPoint(-1, -1);
        }
        System.Diagnostics.Debug.Print("Bars from point: " + vKeys.Count.ToString());
        foreach (object v in vKeys)
        {
            System.Diagnostics.Debug.Print(v.ToString());
            exg2antt1.Items.set_BarSelectable(h, v, true);
        }
    }
}
```

The following VFP sample displays the start data of the bar from the point:

```
*** ActiveX Control Event ***
LPARAMETERS button, shift, x, y

With thisform.G2antt1
    local h, c, hit
```

```
   h = .ItemFromPoint(-1, -1, c, hit)
   If (h # 0) Then
      local k
      k = .Chart.BarFromPoint(-1, -1)
      If !Empty(k) Then
         ? .Items.ItemBar(h, k, 1)
      EndIf
   EndIf
EndWith
```

property Chart.Bars as Bars

Retrieves the Bars collection.

Type	Description
Bars	A Bars collection that holds Bar objects.

Use the Bars property to access the control's Bars collection. Use the Add or Copy property to add new type of bars to the control. Use the AddBar method to add new bars to an item. Use the Chart property to access the Chart object.

By default, the Bars collection includes the following predefined bars:

- "Deadline": 
- "Project Summary": 
- "Summary": 
- "Milestone": 
- "Progress":
- "Split":
- "Task": 

property Chart.BarsAllowSizing as Boolean

Specifies whether bars can be resized at run-time.

Type	Description
Boolean	A boolean expression that indicates whether the control allows resizing or moving the bars in the chart area.

Use the BarsAllowSizing property to specify whether the control allows resizing or moving the bars in the chart area. By default, the BarsAllowSizing property is True. Use the ItemBar(,,exBarCanResize) property to specify whether the bar is resizable. By default, all bars are resizable. The control displays a resizing cursor while the user hovers the mouse over the bar. The user may start the resizing/moving the operation by clicking the bar and moving it to a new position. The control is scrolled if required. The DateTickerLabel property specifies the label (being displayed across the ticker) that shows the start and end dates of the moved or resized bar.

property Chart.CanRedo as Boolean

Retrieves a value that indicates whether the chart can perform a Redo operation.

Type	Description
Boolean	A Boolean expression that specifies whether the chart can perform the next action in the chart's Redo queue.

For instance, you can use the CanRedo property to update the Redo button in your toolbar, so the user knows that Redo operations in the chart may be performed. The Redo redoes the next action in the chart's redo queue. If the AllowUndoRedo property is True, the CTRL+Y redoes the next action in the chart's Redo queue. The RedoListAction property lists the Redo actions that can be performed in the chart.

property Chart.CanUndo as Boolean

Retrieves a value that indicates whether the chart can perform an Undo operation.

Type	Description
Boolean	A Boolean expression that specifies whether the chart can perform the last Undo operation.

For instance, you can use the CanUndo property to update the Undo button in your toolbar, so the user knows that Undo operations in the chart may be performed. Call the Undo method to Undo the last chart operation. By default, the if the AllowUndoRedo property is True, the CTRL+Z performs the last Undo operation. The CanRedo property retrieves a value that specifies whether the chart can execute the next operation in the chart's Redo queue. The Redo redoes the next action in the chart's redo queue. If the AllowUndoRedo property is True, the CTRL+Y redoes the next action in the chart's Redo queue. The UndoListAction property lists the Undo actions that can be performed in the chart. The RedoListAction property lists the Redo actions that can be performed in the chart.

method Chart.ClearItemBackColor (Item as HITEM)

Clears the item's background color in the chart area.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.

The ClearItemBackColor method clears the item's background color when ItemBackColor property is used (chart part only). The ClearItemBackColor method clears the item's background color when ItemBackColor property is used (items/columns part only).

method Chart.ClearNonworkingDates ()

Clears nonworking dates.

Type	Description

Use the ClearNonworkingDates method to remove all nonworking dates. Use the ShowNonworkingDates property to show or hide the nonworking dates. Use the RemoveNonworkingDate method to unmark a specified nonworking date, being previously added using the AddNonworkingDate method. Use the IsDateVisible property to specify whether a date fits the chart's area. Use the IsNonworkingDate property to check whether the date is already highlighted as nonworking day. The NonworkingDays property specifies the days being marked as nonworking in a week. Use the NonworkingDaysPattern property to specify the pattern being used to fill non-working days. The NonworkingDaysColor property specifies the color being used to fill the non-working days.

property Chart.ColumnsFont as IFontDisp

Retrieves or sets the font to display the columns in the chart section.

Type	Description
IFontDisp	A Font object to be used when showing the ColumnsFormatLevel property.

By default, the ColumnsFont property is empty, and so the control's Font is used to show the columns in the chart section. Use the ColumnsFont property to use a different font when showing the columns in the chart part of the control. Use the Def(exCellForeColor) property to specify the column's foreground color, so in case you need to change the foreground color of the columns. The ColumnsFormatLevel property may display any visible or hidden column. The Visible property indicates whether a column is visible or hidden in the items section. The ShowTransparentBars property to specify a transparency to displays all bars in the chart. The ColumnsTransparent property specifies the percent of the transparency to display the columns in the chart.

property Chart.ColumnsFormatLevel as String

Specifies the CRD format layout to display the columns in the chart section.

Type	Description
String	A String expression that indicates the CRD format to display the columns in the chart part of the control. You can use the ExCRD tool to generate the CRD strings.

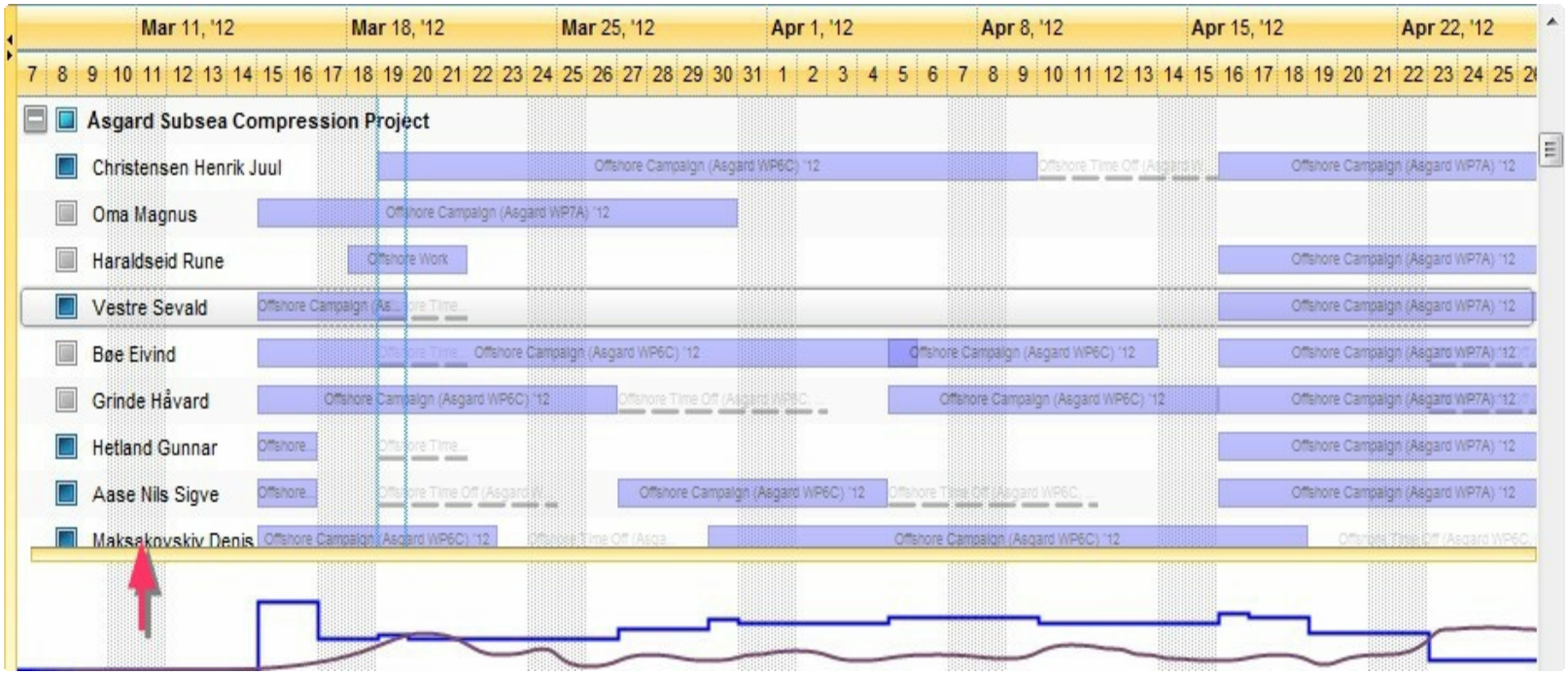
By default, the ColumnsFormatLevel property is empty, so no columns are displayed on the chart. Use the ColumnsFormatLevel property to specify the columns to be displayed on the chart's section. The ColumnsFormatLevel property may display any visible or hidden column. For instance, the ColumnsFormatLevel = "0" displays the first column (the column with the index 0), on the chart. The ColumnsFont property indicates the font to be used when showing the columns on the chart part of the control. The ColumnsTransparent property specifies the percent of the transparency to display the columns in the chart. The Visible property indicates whether a column is visible or hidden in the items section. The ShowTransparentBars property to specify a transparency to displays all bars in the chart. The SelBackColor/SelForeColor property indicates whether the selected item is highlighted in the chart section.

The columns section displays the expanding +/- buttons only if:

- LinesAtRoot
- HasButtons
- Indent

are NOT set on 0. If any of these is 0, the expanding +/- buttons in the chart is not shown.

The following screen shot shows columns on the chart part of the control:



Here's a few samples of using the CRD syntax:

- "**1**:52", displays the column with the index 1 on a 52 pixels width.
- "|,**1**:52", aligns the column with the index 1 on the right side of the chart
- "**1**:52,\"\"[bg=255]:2", displays a red border to the right of the column with the index 1
- "**1**[bg=255]:52", displays the column with the index 1 with different background color (red for 255 or RGB(255,0,0))

property Chart.ColumnsTransparent as Long

Specifies the percent of the transparency to display the columns in the chart.

Type	Description
Long	A Long expression, from 0 to 100, that indicates the percent of transparency that's used to paint the columns in the chart part of the control. 0 means opaque, 100 means hidden, or 100% transparent. 50 means semi-transparent.

By default, the ColumnsTransparent property is 0. Use the ColumnsFormatLevel property to specify the columns to be displayed on the chart's section. For instance, the ColumnsFormatLevel = "0" displays the first column (the column with the index 0), on the chart. The ColumnsFont property indicates the font to be used when showing the columns on the chart part of the control. The ShowTransparentBars property to specify a transparency to displays all bars in the chart. The ColumnsFormatLevel property may display any visible or hidden column. The Visible property indicates whether a column is visible or hidden in the items section.

The following screen shot shows the a check-box column in the right side of the chart, using a semi-transparent color:

property Chart.CondInsideZoom as String

Specifies the formula that indicates the dates that can be zoomed at runtime.

Type	Description
String	A String expression that defines the dates that can be magnified by double clicking the base level, or resizing it. If empty, the CondInsideZoom property has no effect. If not empty and the expression is valid, it indicates the dates that can be magnified. For instance, if the ConsInsideZoom property is "*month(value)= 5*" specifies that user can zoom only dates in May.

By default, the CondnsideZoom property is empty. If empty, the CondInsideZoom property has no effect. If the CondInsideZoom property is not empty and valid, it indicates the dates that can be magnified by double clicking the time unit in the base level, if the InsideZoomOnDblClick property is True, or by resizing the time unit, if the AllowResizeInsideZoom property is True. In other words, the CondInsideZoom property has effect ONLY if the AllowResizeInsideZoom or InsideZoomOnDblClick properties. So, it has no effect when adding the inside zoom units by code, using the Add method.

The expression may be a combination of variables, constants, strings, dates and operators, and value. The ***value*** operator gives the date-time expression being checked. A string is delimited by ", ` or ' characters, and inside they can have the starting character preceded by \ character, ie "\"This is a quote\"". A date is delimited by # character, ie #1/31/2001 10:00# means the January 31th, 2001, 10:00 AM.

For instance, if the CondInsideZoom property is "*weekday(value) = 0*" means that you allow zooming only for Sundays, or if it is "*not(weekday(value) = 0 or weekday(value) = 6)*" the control allow zooming only for working dates, as Monday to Friday.

The **value** keyword in the CondInsideZoom property indicates the date-time expression being checked.

This property/method supports predefined constants and operators/functions as described here.

property Chart.CountVisibleUnits ([Start as Variant], [End as Variant]) as Long

Counts the number of units within the specified range.

Type	Description
Start as Variant	A DATE expression that specifies the starting date, if missing, the StartPrintDate value is used.
End as Variant	A DATE expression that specifies the ending date, if missing, the EndPrintDate value is used.
Long	A long expression that specifies the number of units within the specified range.

Use the CountVisibleUnits property to count the number of units within the specified range. The UnitScale property indicates the time-unit scale being displayed by the chart's header. Use the CountVisibleUnits property to count the number of units so the entire chart is displayed on a specified size. Use the UnitWidth property specifies the width in pixels for the minimal time-unit. Use the CountVisibleUnits property and the ClientWidth property of the eXPrint component (Retrieves the width in pixels, of the drawing area of the printer page) to specify that you need to display the chart on a single page. The StartPrintDate and EndPrintDate property specifies range of dates within the chart is printed.

When computing the UnitWidth property for printing to a page (as shown in the following sample), you can still use the Count property of the Level object to display more units instead one.

The following VB sample changes the UnitWidth property of the eXG2ant's Chart object so, the entire chart is printed to the page:

```
With Print1
   Dim l As Long
   With G2antt1.Chart
      l = .UnitWidth
      .UnitWidth = (Print1.ClientWidth - .PaneWidth(False)) / .CountVisibleUnits()
   End With
   Set .PrintExt = G2antt1.Object
   .Preview
   G2antt1.Chart.UnitWidth = l
End With
```

The sample has the disadvantage that once the user changes the Page's setup during

Previewing the code is not re-executed, so the chart is displayed as it is on the screen. In order to update the UnitWidth property once the page's setup is changed, we need to handle the Refreshing and Refresh events of the eXPrint component as shown in the following VB sample:

```
Dim nUnitWidth As Long

Private Sub Print1_Refreshing()
   With G2antt1.Chart
      nUnitWidth = .UnitWidth
      .UnitWidth = (Print1.ClientWidth - .PaneWidth(False)) / .CountVisibleUnits()
   End With
End Sub

Private Sub Print1_Refresh()
   G2antt1.Chart.UnitWidth = nUnitWidth
End Sub

Private Sub Preview_Click()
   With Print1
      Set .PrintExt = G2antt1.Object
      .Preview
   End With
End Sub
```

The sample changes the UnitWidth property of the Chart during the Refreshing event, so the chart fits to page, and restores the UnitWidth's value when the Refresh event is invoked.

The following VB/NET sample changes the UnitWidth property so the chart fits to page:

```
Dim nUnitWidth As Long

Private Sub Exprint1_RefreshingEvent(ByVal sender As System.Object) Handles
Exprint1.RefreshingEvent
   With Exg2antt1.Chart
      nUnitWidth = .UnitWidth
      .UnitWidth = (Exprint1.ClientWidth - .get_PaneWidth(False)) / .CountVisibleUnits()
   End With
```

```
End Sub

Private Sub Exprint1_RefreshEvent(ByVal sender As System.Object) Handles
Exprint1.RefreshEvent
    Exg2antt1.Chart.UnitWidth = nUnitWidth
End Sub

Private Sub Preview_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Preview.Click
    Exprint1.PrintExt = Exg2antt1
    Exprint1.Preview()
End Sub
```

property Chart.DateFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Date

Retrieves the date from the cursor.

Type	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in client coordinates.
Date	A Date expression that indicates the date from the cursor or 0 if no date found.

The DateFromPoint property gets the date from point. The DateFromPoint property retrieves the date from the cursor, only if the cursor hovers the chart's area. Use the ItemFromPoint property to get the cell/item from the cursor. Use the ColumnFromPoint property to retrieve the column from cursor. Use the FormateDate property to format a date. Use the DrawDateTicker property to draw a ticker as cursor hovers the chart's area. Use the BarFromPoint property to get the bar from the point. Use the LinkFromPoint property to get the link from the point. Use the LevelFromPoint property to retrieve the index of the level from the cursor.

The DateFromPoint property retrieves the value based on the X and Y parameters as follows:

- **if X = -1 and Y = -1, the DateFromPoint property retrieves the date from the cursor, shortly the DateFromPoint(-1,-1) returns the date from the cursor**
- if X = 0 and Y = -1, the DateFromPoint property retrieves the first visible date in the chart, the same as FirstVisibleDate property.
- if X = 1 and Y = -1, the DateFromPoint property retrieves the last visible date in the chart.

The following VB sample displays the date from the cursor:

```
Private Sub G2antt1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
  With G2antt1.Chart
    Dim d As Date
    d = .DateFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
    Debug.Print .FormatDate(d, "<%m%>/<%d%>/<%yyyy%>")
```

```
    End With
End Sub
```

The following C++ sample displays the date from the point:

```
void OnMouseMoveG2antt1(short Button, short Shift, long X, long Y)
{
    CChart chart = m_g2antt.GetChart();
    DATE d = chart.GetDateFromPoint( X, Y );
    CString strFormat = chart.GetFormatDate( d, "<%m%>/<%d%>/<%yyyy%>" );
    OutputDebugString( strFormat );
}
```

The following VB.NET sample displays the date from the point:

```
Private Sub AxG2antt1_MouseMoveEvent(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_MouseMoveEvent) Handles AxG2antt1.MouseMoveEvent
    With AxG2antt1.Chart
        Dim d As Date
        d = .DateFromPoint(e.x, e.y)
        Debug.Write(.FormatDate(d, "<%m%>/<%d%>/<%yyyy%>"))
    End With
End Sub
```

The following C# sample displays the date from the point:

```
private void axG2antt1_MouseMoveEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_MouseMoveEvent e)
{
    DateTime d = axG2antt1.Chart.get_DateFromPoint(e.x, e.y);
    System.Diagnostics.Debug.Write(axG2antt1.Chart.get_FormatDate(d, "
<%m%>/<%d%>/<%yyyy%>"));
}
```

The following VFP sample displays the date from the point:

```
*** ActiveX Control Event ***
LPARAMETERS button, shift, x, y
```

```
with thisform.G2antt1.Chart
   d = .DateFromPoint(x,y)
   wait window nowait .FormatDate(d, "<%m%>/<%d%>/<%yyyy%>" )
endwith
```

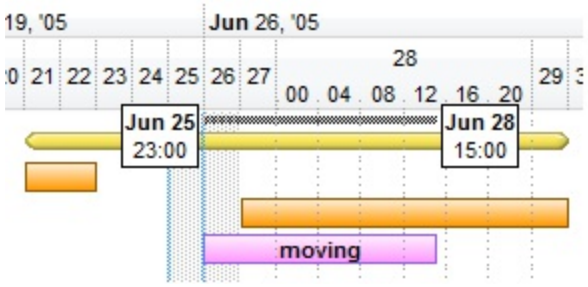
property Chart.DateTickerLabel as String

Retrieves or sets a value that indicates the format to display the bar's start and end date while creating, moving or resizing it.

Type	Description
String	A String expression that specifies the format of the label being displayed while the item is moved or resized. It supports HTML format, <%%> tags and <%=formula%> expressions as explained below

By default, the DateTickerLabel property is empty. The DateTickerLabel property shows the start and end date of the bar being created, moved or resized at runtime using the mouse. Use the
 HTML tag to break the lines, in case you need to display the label using multiple lines. The label always ensure that can be displayed in the chart's area.

The following screen shot shows the ticker and it's label that displays the start and and time of the "**moving**" bar



For instance:

- "<%mmm%> <%d%>
<%hh%>:<%nn%>", displays the month (three letters) and the day in the first line, while the hour and minute on the second line
- "<%=shortdate(value=end?value-1:value)%>", displays the end-margin with one day before
- "<%mmm%> <%d%><fgcolor 808080><%=value=end?` (`+(end - start) + `)`:``%>" displays the month, the day and for the end-margin includes the number of days of the bar being created, resized or moved
- "<%=value=start?``:value%>" specifies that only end-margin of the bar is being shown

The DateTickerLabel supports <%=formula%> expressions, where formula. The formula supports the following keywords:

- **value**, specifies the date being displayed (could be the **start** or the **end** margin of the bar, DATE type)
- **start**, specifies the start-margin of the bar as DATE type. The **start** result is equivalent of the ItemBar(exBarStart) property.

- **end**, specifies the end-margin of the bar as DATE type. The **end** result is equivalent of the ItemBar(exBarEnd) property.
- **wcount**, returns the working-count between start and end margins, as a number (1 indicates one-day). The **wcount** result is equivalent of the ItemBar(exBarWorkingCount) property. The NonworkingDays, NonworkingHours, ItemNonworkingUnits properties define the non-working portion of the chart. The **end - start** gets the duration (in days) of the task. The **end - start -wcount** gets the bar's non-working length.

Also, the formula supports predefined constants and operators/functions as described here.

The DateTickerLabel supports the following built-in tags:

- <%**d**%> - Day of the month in one or two numeric digits, as needed (1 to 31).
- <%**dd**%> - Day of the month in two numeric digits (01 to 31).
- <%**d1**%> - First letter of the weekday (S to S). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**d2**%> - First two letters of the weekday (Su to Sa). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**d3**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**ddd**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays property to specify the name of the days in the week). You can use the <%**loc_ddd**%> that indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%**loc_ddd**%> - Indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%**dddd**%> - Full name of the weekday (Sunday to Saturday). (Use the WeekDays property to specify the name of the days in the week). You can use the <%**loc_dddd**%> that indicates day of week as its full name using the current user regional and language settings.
- <%**loc_dddd**%> - Indicates day of week as its full name using the current user regional and language settings.
- <%**i**%> - Displays the number instead the date. For instance, you can display numbers as 1000, 1001, 1002, 1003, instead dates. (the valid range is from -647,434 to 2,958,465)
- <%**w**%> - Day of the week (1 to 7).
- <%**ww**%> - Week of the year (1 to 53).

- <%**m**%> - Month of the year in one or two numeric digits, as needed (1 to 12).
- <%**mr**%> - Month of the year in Roman numerals, as needed (I to XII).
- <%**mm**%> - Month of the year in two numeric digits (01 to 12).
- <%**m1**%> - First letter of the month (J to D). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**m2**%> - First two letters of the month (Ja to De). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**m3**%> - First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**mmm**%> - First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmm**%> that indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%**loc_mmm**%> - Indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%**mmmm**%> - Full name of the month (January to December). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmmm**%> that indicates month as its full name using the current user regional and language settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user regional and language settings.
- <%**q**%> - Date displayed as the quarter of the year (1 to 4).
- <%**y**%> - Number of the day of the year (1 to 366).
- <%**yy**%> - Last two digits of the year (01 to 99).
- <%**yyyy**%> - Full year (0100 to 9999).
- <%**hy**%> - Date displayed as the half of the year (1 to 2).
- <%**loc_gg**%> - Indicates period/era using the current user regional and language settings.
- <%**loc_sdate**%> - Indicates the date in the short format using the current user regional and language settings.
- <%**loc_ldate**%> - Indicates the date in the long format using the current user regional and language settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user regional and language settings (/).
- <%**h**%> - Hour in one or two digits, as needed (0 to 23).
- <%**hh**%> - Hour in two digits (00 to 23).
- <%**h12**%> - Hour in 12-hour time format, in one or two digits - [0(12),11]

- <%**hh12**%> - hour in 12-hour time format, in two digits - [00(12),11]
- <%**n**%> - Minute in one or two digits, as needed (0 to 59).
- <%**nn**%> - Minute in two digits (00 to 59).
- <%**s**%> - Second in one or two digits, as needed (0 to 59).
- <%**ss**%> - Second in two digits (00 to 59).
- <%**AM/PM**%> - Twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate. (Use the AMPM property to specify the name of the AM and PM indicators). You can use the <%**loc_AM/PM**%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%**loc_A/P**%> that indicates the one character time marker such as A or P using the current user regional and language settings
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user regional and language settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user regional and language settings.
- <%**loc_time**%> - Indicates the time using the current user regional and language settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%**loc_tsep**%> - indicates the time separator using the current user regional and language settings (:)
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

The following tags are displayed based on the user's Regional and Language Options:

- <%**loc_sdate**%> - Indicates the date in the short format using the current user settings.
- <%**loc_ldate**%> - Indicates the date in the long format using the current user settings.
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.

- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**loc_ddd**%> - Indicates day of week as a three-letters abbreviation using the current user settings.
- <%**loc_dddd**%> - Indicates day of week as its full name using the current user settings.
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**loc_mmm**%> - Indicates month as a three-letters abbreviation using the current user settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user settings.
- <%**loc_gg**%> - Indicates period/era using the current user settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user settings.
- <%**loc_time**%> - Indicates the time using the current user settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user settings.
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user settings.
- <%**loc_tsep**%> - Indicates the time separator using the current user settings.
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

The DateTickerLabel property supports the following built-in HTML tags:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a

piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.

The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using <a ;exp=> or <a ;e64=> anchor tags. The exp/e64 field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption.

- exp, stores the plain text to be shown once the user clicks the anchor, such as <a ;exp=show lines>
- e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as <a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu that displays show lines- in gray when the user clicks the + anchor. The gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY/ string encodes the <fgcolor 808080>show lines<a>-</fgcolor> The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, <solidline>Header</solidline>
Line1<r> <a ;exp=show lines>+
Line2
Line3 shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The show lines is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** displays portions of text with a different font and/or different size. For instance, the bit draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, bit displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.

- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: Text with <**off** 6>subscript displays the text such as: Text with $_{\text{subscript}}$ The Text with <**off** -6>superscript displays the text such as: Text with $^{\text{subscript}}$
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the <**gra** FFFFFF;1;1>gradient-center</**gra**> generates the following picture:

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the <**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**> generates the following picture:

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the <**sha**>shadow</**sha**> generates the following picture:

or *<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>* gets:

Your application can provide some options to help user while performing moving or resizing at runtime as follow:

- grid lines, that can be shown only when moving or resizing, using the ChartStartChanging and ChartEndChanging events
- select date, to specify the margins of the are you what to highlight
- ticker, that shows the cursor's position in the chart, or while resizing, it shows the size and the position of the bar
- ability to specify a resizing/moving unit, different that the displayed one ie while the chart displays days, you can specify the resizing unit on hours.
- inside zoom, that can be used to magnify the portion of the chart being selected

*Added: The Chart.DateTickerLabel property property supports <%=formula%> expression to customize the label to be shown for start and/or end margins of the bar being created, moved or resized. The expression (after the = character) supports keywords such as "value" that defines the DATE being displayed (could be start or end), "start" that defines the start-margin of the bar as a DATE type and "end" that defines the end-margin of the bar

as a DATE type. For instance,

property Chart.DefaultInsideZoomFormat as InsideZoomFormat

Retrieves the format of the inside zoom units.

Type	Description
InsideZoomFormat	An InsideZoomFormat object that defines the look and feel for newly added inside zoom units.

The DefaultInsideZoomFormat property defines the format of the inside zoom units. By default, the inside zoom units displays hours. The DefaultInsideZoomFormat property is applied to all inside zoom units, that has AllowCustomFormat property on False (by default). The InsideLabel property defines the label of the units once they get magnified. The control fires the InsideZoom event once the user magnifies a date. The inside zoom units are displayed only if the AllowInsideZoom property is True (by default, the AllowInsideZoom property is False).

The following VB sample shows how can I change the scale unit when doing inside zoom (the chart displays weeks, and we want week days):

```
With G2antt1
   .BeginUpdate
   With .Chart
      .ShowNonworkingDates = False
      .PaneWidth(0) = 0
      .LevelCount = 2
      With .Level(0)
         .Label = "<%mmmm%>"
         .Unit = exMonth
      End With
      With .Level(1)
         .Label = "<%ww%>"
         .Unit = exWeek
      End With
      .FirstVisibleDate = #1/1/2008#
      .AllowInsideZoom = True
      With .DefaultInsideZoomFormat
         .OwnerLabel = "<font ;7><%mmm%> Week: <%ww%>"
         .InsideLabel = "<font ;7><b><%d1%></b>"
         .InsideUnit = exDay
      End With
```

```
      With .InsideZooms
         .SplitBaseLevel = False
         .Add #2/3/2008#
      End With
   End With
   .EndUpdate
End With
```

The following VB.NET sample shows how can I change the scale unit when doing inside zoom (the chart displays weeks, and we want week days):

```
With AxG2antt1
   .BeginUpdate
   With .Chart
      .ShowNonworkingDates = False
      .PaneWidth(0) = 0
      .LevelCount = 2
      With .Level(0)
         .Label = "<%mmmm%>"
         .Unit = EXG2ANTTLib.UnitEnum.exMonth
      End With
      With .Level(1)
         .Label = "<%ww%>"
         .Unit = EXG2ANTTLib.UnitEnum.exWeek
      End With
      .FirstVisibleDate = #1/1/2008#
      .AllowInsideZoom = True
      With .DefaultInsideZoomFormat
         .OwnerLabel = "<font ;7><%mmm%> Week: <%ww%>"
         .InsideLabel = "<font ;7><b><%d1%></b>"
         .InsideUnit = EXG2ANTTLib.UnitEnum.exDay
      End With
      With .InsideZooms
         .SplitBaseLevel = False
         .Add #2/3/2008#
      End With
   End With
   .EndUpdate
```

End With

The following C++ sample shows how can I change the scale unit when doing inside zoom (the chart displays weeks, and we want week days):

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)->GetControlUnknown();
spG2antt1->BeginUpdate();
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutShowNonworkingDates(VARIANT_FALSE);
   var_Chart->PutPaneWidth(0,0);
   var_Chart->PutLevelCount(2);
   EXG2ANTTLib::ILevelPtr var_Level = var_Chart->GetLevel(0);
      var_Level->PutLabel("<%mmmm%>");
      var_Level->PutUnit(EXG2ANTTLib::exMonth);
   EXG2ANTTLib::ILevelPtr var_Level1 = var_Chart->GetLevel(1);
      var_Level1->PutLabel("<%ww%>");
      var_Level1->PutUnit(EXG2ANTTLib::exWeek);
   var_Chart->PutFirstVisibleDate("1/1/2008");
   var_Chart->PutAllowInsideZoom(VARIANT_TRUE);
   EXG2ANTTLib::IInsideZoomFormatPtr var_InsideZoomFormat = var_Chart->GetDefaultInsideZoomFormat();
      var_InsideZoomFormat->PutOwnerLabel(L"<font ;7><%mmm%> Week: <%ww%>");
      var_InsideZoomFormat->PutInsideLabel(L"<font ;7><b><%d1%></b>");
      var_InsideZoomFormat->PutInsideUnit(EXG2ANTTLib::exDay);
   EXG2ANTTLib::IInsideZoomsPtr var_InsideZooms = var_Chart->GetInsideZooms();
      var_InsideZooms->PutSplitBaseLevel(VARIANT_FALSE);
      var_InsideZooms->Add("2/3/2008");
spG2antt1->EndUpdate();
```

The following C# sample shows how can I change the scale unit when doing inside zoom (

the chart displays weeks, and we want week days):

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.ShowNonworkingDates = false;
   var_Chart.set_PaneWidth(0 != 0,0);
   var_Chart.LevelCount = 2;
   EXG2ANTTLib.Level var_Level = var_Chart.get_Level(0);
      var_Level.Label = "<%mmmm%>";
      var_Level.Unit = EXG2ANTTLib.UnitEnum.exMonth;
   EXG2ANTTLib.Level var_Level1 = var_Chart.get_Level(1);
      var_Level1.Label = "<%ww%>";
      var_Level1.Unit = EXG2ANTTLib.UnitEnum.exWeek;
   var_Chart.FirstVisibleDate = "1/1/2008";
   var_Chart.AllowInsideZoom = true;
   EXG2ANTTLib.InsideZoomFormat var_InsideZoomFormat =
var_Chart.DefaultInsideZoomFormat;
      var_InsideZoomFormat.OwnerLabel = "<font ;7><%mmm%> Week: <%ww%>";
      var_InsideZoomFormat.InsideLabel = "<font ;7><b><%d1%></b>";
      var_InsideZoomFormat.InsideUnit = EXG2ANTTLib.UnitEnum.exDay;
   EXG2ANTTLib.InsideZooms var_InsideZooms = var_Chart.InsideZooms;
      var_InsideZooms.SplitBaseLevel = false;
      var_InsideZooms.Add("2/3/2008");
axG2antt1.EndUpdate();
```

The following VFP sample shows how can I change the scale unit when doing inside zoom (the chart displays weeks, and we want week days):

```
with thisform.G2antt1
   .BeginUpdate
   with .Chart
      .ShowNonworkingDates = .F.
      .PaneWidth(0) = 0
      .LevelCount = 2
      with .Level(0)
         .Label = "<%mmmm%>"
         .Unit = 16
      endwith
```

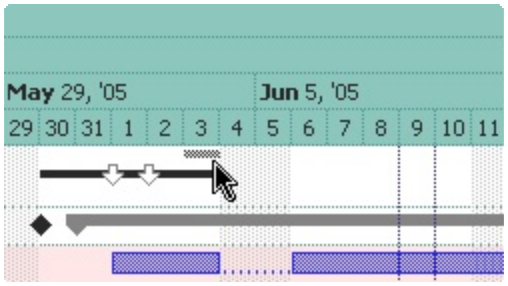
```
      with .Level(1)
         .Label = "<%ww%>"
         .Unit = 256
      endwith
      .FirstVisibleDate = {^2008-1-1}
      .AllowInsideZoom = .T.
      with .DefaultInsideZoomFormat
         .OwnerLabel = "<font ;7><%mmm%> Week: <%ww%>"
         .InsideLabel = "<font ;7><b><%d1%></b>"
         .InsideUnit = 4096
      endwith
      with .InsideZooms
         .SplitBaseLevel = .F.
         .Add({^2008-2-3})
      endwith
   endwith
   .EndUpdate
endwith
```

property Chart.DrawDateTicker as Boolean

Retrieves or sets a value that indicates whether the control draws a ticker around the current date while cursor hovers the chart's client area.

Type	Description
Boolean	A Boolean expression that indicates whether the date ticker is visible or hidden.

By default, the DrawDateTicker property is False. Use the DrawDateTicker property to show or hide the ticker that shows up while the cursor hovers the chart's area. The ticker indicates the size and position of the focused bar while it is resized or moved. The DateTickerLabel property specifies the label (being displayed across the ticker) that shows the start and end dates of the moved or resized bar. Use the DateFromPoint property to retrieve the date from the cursor. Use the NonworkingDays property to specify the nonworking days. Use the MarkTodayColor property to specify whether the today date is marked. Use the DrawTickLines property to specify whether the grid lines between time units in the level are visible or hidden. The DateTickerLabel property shows the start and end date of the bar being moved or resized.



Your application can provide some options to help user while performing moving or resizing the bars at runtime as follow:

- grid lines, that can be shown only when moving or resizing, using the ChartStartChanging and ChartEndChanging events
- select date, to specify the margins of the area you want to highlight
- ticker, that shows the cursor's position in the chart, or while resizing, it shows the size and the position of the bar
- ability to specify a resizing/moving unit, different that the displayed one ie while the chart displays days, you can specify the resizing unit on hours.
- inside zoom, that can be used to magnify the portion of the chart being selected

property Chart.DrawGridLines as GridLinesEnum

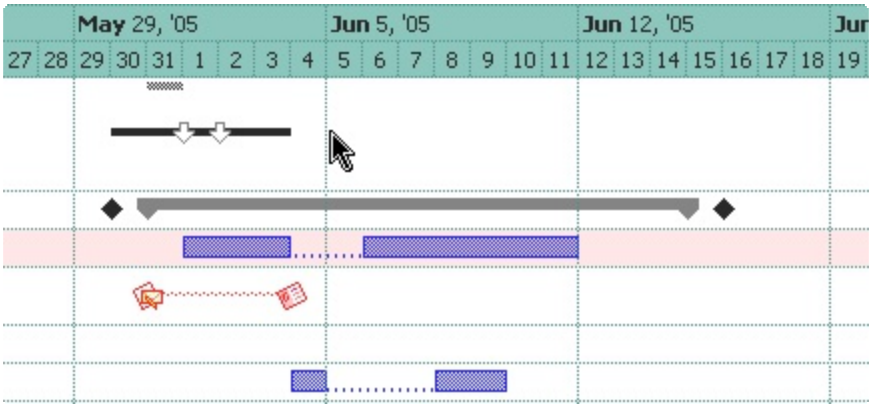
Retrieves or sets a value that indicates whether the grid lines are visible or hidden.

Type	Description
GridLinesEnum	A GridLinesEnum expression that indicates whether the control draws the grid lines in the chart's area.

By default, the DrawGridLines property is exNoLines. Use the DrawGridLines property to specify whether the control draws the grid lines in the chart's area. Use the GridLineStyle property of the Level object to specify the style for vertical grid lines in the chart area. Use the GridLineColor property to specify the color for grid lines. Use the DrawGridLines property to specify whether the control draws the grid lines in the items area. Use the DrawLevelSeperator property to draw lines between levels inside the chart's header. Use the DrawTickLines property to specify whether the grid lines between time units in the level are visible or hidden. Use the MarkTodayColor property to specify the color to mark the today date. Use the DrawGridLines property to draw grid lines for a specified level. Use the NonworkingDays property to specify the nonworking days. Use the NonworkingDaysPattern property to specify the brush to fill the nonworking days area.

In conclusion, the following properties are related to the control's gridlines:

- DrawGridLines specifies whether the gridlines are shown in the column/list part of the control. The gridlines in the chart part of the control are handled by the Chart.DrawGridLines property.
- GridLineColor specifies the color to show the horizontal grid line, and vertical grid lines for the columns/list part of the control. The color for vertical grid lines in the chart view part is handled by the Level.GridLineColor property.
- GridLineStyle specifies the style for horizontal grid lines and vertical grid lines in the columns/list part of the control. The Level.GridLineStyle property specifies the style for vertical grid lines in the chart area.
- Chart.DrawGridLines (belongs to Chart object) indicates whether gridlines are shown in the chart view.
- Level.DrawGridLines (belongs to Level object) specifies whether the level shows vertical gridlines in the chart part of the control.
- Level.GridLineColor (belongs to Level object) indicates the color for vertical gridlines in the chart view.
- Level.GridLineStyle (belongs to Level object) specifies the style to show the vertical gridlines in the chart part area of the control.

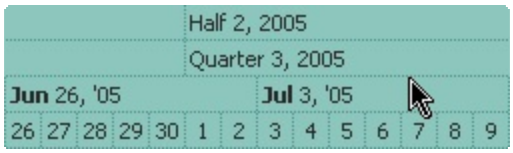

May 29, '05
Jun 5, '05
Jun 12, '05
Jur
27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

property Chart.DrawLevelSeparator as LevelLineEnum

Retrieves or sets a value that indicates whether lines between levels are shown or hidden.

Type	Description
LevelLineEnum	A LevelLineEnum value that specifies the type of line being shown between levels.

By default, the DrawLevelSeparator property is exLevelDefaultLine (dotted line) Use the DrawLevelSeperator property to draw lines between levels inside the chart's header. The DrawTickLines / DrawTickLinesFrom property always draw the vertically lines in the level, while the DrawLevelSeparator property draws the horizontally lines in the level. Use the DrawTickLines property to specify whether the grid lines between time units in the level are visible or hidden. Use the DrawGridLines property to specify whether the control draws the grid lines in the chart's area. Use the GridLineColor property to specify the color for grid lines. Use the DrawGridLines property to specify whether the control draws the grid lines in the items area. Use the DrawGridLines property to draw grid lines for a specified level. Use the NonworkingDays property to specify the nonworking days. Use the NonworkingDaysPattern property to specify the brush to fill the nonworking days area. Use the MarkTodayColor property to specify the color to mark the today date.



method Chart.EndBlockUndoRedo ()

Ends recording the UI operations and adds the undo/redo operations as a block, so they all can be restored at once, if Undo method is performed.

Type	Description

The StartBlockUndoRedo method starts recording the UI operations as a block on undo/redo operations The method has effect only if the AllowUndoRedo property is True. The EndBlockUndoRedo method collects all undo/redo operations since StartBlockUndoRedo method was called and add them to the undo/redo queue as a block. This way the next call on a Undo operation, the entire block is restored, so all UI operations are restored. For instance, if you have a procedure that moves several bars, and want all of them being grouped, you can use StartBlockUndoRedo to start recording the operations as a block, and call the EndBlockUndoRedo when procedure ends, so next call of an undo operation the bars are restored to their original position. The EndBlockUndoRedo method must be called the same number of times as the StartBlockUndoRedo method was called. For instance, if you have called the StartBlockUndoRedo twice the EndBlockUndoRedo method must be called twice too, and the collected operations are added to the chart's queue of undo/redo operations at the end.

property Chart.EndPrintDate as Variant

Retrieves or sets a value that indicates the printing end date.

Type	Description
Variant	A DATE expression that specifies the ending date to print the chart. The get method always retrieves a DATE expression. When calling the set method of the EndPrintDate property, it can be a string, a DATE or any other expression that can be converted to a date.

The EndPrintDate property indicates the date the chart ends, when:

- printing the control's content using the exprint component
- coping the control's content using the CopyTo method (since 22.0.1.5)

By default, the EndPrintDate property computes the required end date so the entire chart is displayed, *if the EndPrintDate was not specified before*. For instance, if you set the EndPrintDate property on "Dec 31 2001", the EndPrintDate property retrieves the "Dec 31 2001" date and does not compute the required end date. If you have specified a value for the EndPrintDate but you still need to get the required end date being computed, set the EndPrintDate property on 0, and calling the next method get of EndPrintDate property computes the required end date to print the chart. The StartPrintDate property indicates the starting date to print the chart. Use the CountVisibleUnits property to count the number of units within the specified range.

property Chart.FirstVisibleDate as Variant

Retrieves or sets a value that indicates the first visible date.

Type	Description
Variant	A Date expression that indicates the first visible date in the chart.

The FirstVisibleDate property indicates the first visible date in the chart. The control fires the DateChange event when the first visible date is changed. The DateFromPoint(1,-1) returns the last visible date. Use the FormatDate property to format a date to a specified format. Use the NextDate property to retrieve the next or previous date giving a specified time unit. Use the ScrollTo method to ensure that a specified date fits the chart's client area. Use the AddBar property to add new bars to an item. The DateFromPoint property gets the date from the cursor. Use the FirstWeekDay property to specify the first day in the week. Use the Zoom method to scale the chart to a specified interval of dates. The StartPrintDate property indicates the starting date to print the chart.

The following VB sample displays the first visible date when the user changes the first visible date:

```
Private Sub G2antt1_DateChange()
  With G2antt1.Chart
    Debug.Print FormatDateTime(.FirstVisibleDate)
  End With
End Sub
```

or you can use the FormatDate method like follows:

```
Private Sub G2antt1_DateChange()
  With G2antt1.Chart
    Debug.Print .FormatDate(.FirstVisibleDate, "<%yyyy%>-<%m%>-<%d%>")
  End With
End Sub
```

The following VB sample determines the last visible date:

```
Private Function LastVisibleDate(ByVal g As EXG2ANTTLibCtl.G2antt) As Date
  With G2antt1
    With .Chart
      Dim d As Date
```

```
        d = .FirstVisibleDate
        Do While .IsDateVisible(d)
            d = .NextDate(d, exDay, 1)
        Loop
    End With
  End With
  LastVisibleDate = d - 1
End Function
```

The following C++ sample displays the first visible date when the user changes the first visible date:

```
#include "G2antt.h"
#include "Chart.h"

static DATE V2D( VARIANT* pvtDate )
{
  COleVariant vtDate;
  vtDate.ChangeType( VT_DATE, pvtDate );
  return V_DATE( &vtDate );
}

void OnDateChangeG2antt1()
{
  if ( m_g2antt.GetControlUnknown() )
  {
    CChart chart = m_g2antt.GetChart();
    TCHAR szDate[1024] = _T("");
    SYSTEMTIME stDate = {0};
    VariantTimeToSystemTime( V2D( &chart.GetFirstVisibleDate() ), &stDate );
    GetDateFormat( LOCALE_SYSTEM_DEFAULT, LOCALE_USE_CP_ACP, &stDate, NULL,
szDate, 1024 );
    OutputDebugString( szDate );
  }
}
```

The following VB.NET sample displays the first visible date when the user changes the first visible date:

```
Private Sub AxG2antt1_DateChange(ByVal sender As Object, ByVal e As System.EventArgs)
Handles AxG2antt1.DateChange
    Debug.Write(AxG2antt1.Chart.FirstVisibleDate.ToString())
End Sub
```

The following C# sample displays the first visible date when the user changes the first visible date:

```
private void axG2antt1_DateChange(object sender, EventArgs e)
{
    System.Diagnostics.Debug.Write(axG2antt1.Chart.FirstVisibleDate.ToString());
}
```

The following VFP sample displays the first visible date when the user changes the first visible date:

```
*** ActiveX Control Event ***

with thisform.G2antt1.Chart
    wait window nowait .FormatDate(.FirstVisibleDate, "<%yyyy%>-<%m%>-<%d%>")
endwith
```

property Chart.FirstWeekDay as WeekDayEnum

Specifies the first day of the week.

Type	Description
WeekDayEnum	A WeekDayEnum expression that indicates the first day in the week.

By default, the FirstWeekDay property is exSunday. Use the FirstWeekDay property to specify the first day in the week. Use WeekDays property to specify the name of the days in the week. Use the MonthNames property to specify the name of the months in the year. Use the AMPM property to specify the name of the AM and PM indicators. The FormatDate property formats a date. The NextDate property computes the next date based on the time unit. Use the FirstVisibleDate property to specify the first visible date in the chart. Use the MarkTodayColor property to specify the color to mark the today date area. The WeekNumberAs property specifies the way the control displays the week number.

property Chart.ForeColor as Color

Retrieves or sets a value that indicates the chart's foreground color.

Type	Description
Color	A Color expression that indicates the chart's foreground color.

Use the ForeColor property to specify the chart's foreground color. Use the BackColor property to specify the chart's background color. Use the BackColorLevelHeader property to specify the background color of the chart's header. Use the ForeColorLevelHeader property to specify the foreground color of the chart's header. Use the BackColor property to specify the background color for a specified level. Use the ForeColor property to specify the foreground color for a specified level. Use the ItemBackColor property to change the item's background color. Use the NonworkingDaysColor property the color of the brush to fill the nonworking days area. Use the Picture property to specify the picture being displayed on the chart's area. Use the SelForeColor property to change the foreground color of selected items being displayed in the chart area.

The following VB sample changes the chart's foreground color:

```
With G2antt1.Chart
   .ForeColor = RGB(&H80, &H80, &H80)
End With
```

The following C++ sample changes the chart's foreground color:

```
m_g2antt.GetChart().SetForeColor( RGB(0x80,0x80,0x80) );
```

The following VB.NET sample changes the chart's foreground color:

```
With AxG2antt1.Chart
   .ForeColor = ToUInt32(Color.FromArgb(&H80, &H80, &H80))
End With
```

where the ToUInt32 function converts a Color expression to an OLE_COLOR type:

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
   Dim i As Long
   i = c.R
   i = i + 256 * c.G
   i = i + 256 * 256 * c.B
```

```
    ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample changes the chart's foreground color:

```
axG2antt1.Chart.ForeColor = ToUInt32(Color.FromArgb(0x80, 0x80, 0x80));
```

where the ToUInt32 function converts a Color expression to an OLE_COLOR type:

```
private UInt32 ToUInt32(Color c)
{
    long i;
    i = c.R;
    i = i + 256 * c.G;
    i = i + 256 * 256 * c.B;
    return Convert.ToUInt32(i);
}
```

The following VFP sample changes the chart's foreground color:

```
With thisform.G2antt1.Chart
    .ForeColor = RGB(128, 128, 128)
EndWith
```

property Chart.ForeColorLevelHeader as Color

Specifies the foreground color for the chart's levels.

Type	Description
Color	A Color expression that indicates the background color for the chart's header.

Use the ForeColorLevelHeader property to specify the foreground color of the chart's header. Use the BackColorLevelHeader property to specify the background color of the chart's header. Use the LevelCount property to specify the number of levels in the chart's header. Use the Level property to access a level. Use the BackColor property to specify the background color for a specified level. Use the ForeColor property to specify the foreground color for a specified level. Use the BackColor property to specify the chart's background color. Use the ForeColor property to specify the chart's foreground color. Use the ItemBackColor property to change the item's background color. Use the NonworkingDaysColor property the color of the brush to fill the nonworking days area. Use the Picture property to specify the picture being displayed on the chart's area.

The following VB sample changes the chart's header foreground color:

```
With G2antt1.Chart
   .ForeColorLevelHeader = RGB(&H80, &H80, &H80)
End With
```

The following C++ sample changes the chart's header foreground color:

```
m_g2antt.GetChart().SetForeColorLevelHeader( RGB(0x80,0x80,0x80) );
```

The following VB.NET sample changes the chart's header foreground color:

```
With AxG2antt1.Chart
   .ForeColorLevelHeader = ToUInt32(Color.FromArgb(&H80, &H80, &H80))
End With
```

where the ToUInt32 function converts a Color expression to an OLE_COLOR type:

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
   Dim i As Long
   i = c.R
   i = i + 256 * c.G
   i = i + 256 * 256 * c.B
```

```
    ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample changes the chart's header foreground color:

```
axG2antt1.Chart.ForeColorLevelHeader = ToUInt32(Color.FromArgb(0x80, 0x80, 0x80));
```

where the ToUInt32 function converts a Color expression to an OLE_COLOR type:

```
private UInt32 ToUInt32(Color c)
{
    long i;
    i = c.R;
    i = i + 256 * c.G;
    i = i + 256 * 256 * c.B;
    return Convert.ToUInt32(i);
}
```

The following VFP sample changes the chart's header foreground color:

```
With thisform.G2antt1.Chart
    .ForeColorLevelHeader = RGB(128, 128, 128)
EndWith
```

property Chart.FormatDate (Date as Date, Format as String) as String

Formats the date.

Type	Description
Date as Date	A Date expression being formatted
Format as String	A String expression that indicates the format of date.
String	A String expression that indicates the formatted date.

Use the FormatDate property to format a date. Use the NextDate property to increase or decrease a date based on a time unit. Use the FirstVisibleDate property to retrieve the first visible date. The DateFromPoint property gets the date from the cursor. Use the WeekDays property to specify the name of the days in the week. Use the MonthNames property to specify the name of the months in the year. Use the AMPM property to specify the name of the AM and PM indicators.

The Format parameter may include the following built-in tags:

- <%**d**%> - Day of the month in one or two numeric digits, as needed (1 to 31).
- <%**dd**%> - Day of the month in two numeric digits (01 to 31).
- <%**d1**%> - First letter of the weekday (S to S). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**d2**%> - First two letters of the weekday (Su to Sa). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**d3**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**ddd**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays property to specify the name of the days in the week). You can use the <%**loc_ddd**%> that indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%**loc_ddd**%> - Indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%**dddd**%> - Full name of the weekday (Sunday to Saturday). (Use the WeekDays property to specify the name of the days in the week). You can use the <%**loc_dddd**%> that indicates day of week as its full name using the current user regional and language settings.
- <%**loc_dddd**%> - Indicates day of week as its full name using the current user

regional and language settings.

- <%**i**%> - Displays the number instead the date. For instance, you can display numbers as 1000, 1001, 1002, 1003, instead dates. (the valid range is from -647,434 to 2,958,465)
- <%**w**%> - Day of the week (1 to 7).
- <%**ww**%> - Week of the year (1 to 53).
- <%**m**%> - Month of the year in one or two numeric digits, as needed (1 to 12).
- <%**mr**%> - Month of the year in Roman numerals, as needed (I to XII).
- <%**mm**%> - Month of the year in two numeric digits (01 to 12).
- <%**m1**%> - First letter of the month (J to D). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**m2**%> - First two letters of the month (Ja to De). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**m3**%> - First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**mmm**%> - First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmm**%> that indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%**loc_mmm**%> - Indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%**mmmm**%> - Full name of the month (January to December). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmmm**%> that indicates month as its full name using the current user regional and language settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user regional and language settings.
- <%**q**%> - Date displayed as the quarter of the year (1 to 4).
- <%**y**%> - Number of the day of the year (1 to 366).
- <%**yy**%> - Last two digits of the year (01 to 99).
- <%**yyyy**%> - Full year (0100 to 9999).
- <%**hy**%> - Date displayed as the half of the year (1 to 2).
- <%**loc_gg**%> - Indicates period/era using the current user regional and language settings.
- <%**loc_sdate**%> - Indicates the date in the short format using the current user regional and language settings.
- <%**loc_ldate**%> - Indicates the date in the long format using the current user regional

and language settings.

- <%**loc_dsep**%> - Indicates the date separator using the current user regional and language settings (/).
- <%**h**%> - Hour in one or two digits, as needed (0 to 23).
- <%**hh**%> - Hour in two digits (00 to 23).
- <%**h12**%> - Hour in 12-hour time format, in one or two digits - [0(12),11]
- <%**hh12**%> - hour in 12-hour time format, in two digits - [00(12),11]
- <%**n**%> - Minute in one or two digits, as needed (0 to 59).
- <%**nn**%> - Minute in two digits (00 to 59).
- <%**s**%> - Second in one or two digits, as needed (0 to 59).
- <%**ss**%> - Second in two digits (00 to 59).
- <%**AM/PM**%> - Twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate. (Use the AMPM property to specify the name of the AM and PM indicators). You can use the <%**loc_AM/PM**%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%**loc_A/P**%> that indicates the one character time marker such as A or P using the current user regional and language settings
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user regional and language settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user regional and language settings.
- <%**loc_time**%> - Indicates the time using the current user regional and language settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%**loc_tsep**%> - indicates the time separator using the current user regional and language settings (:).
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

The following tags are displayed based on the user's Regional and Language Options:

- <%**loc_sdate**%> - Indicates the date in the short format using the current user

settings.

- <%**loc_ldate**%> - Indicates the date in the long format using the current user settings.
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**loc_ddd**%> - Indicates day of week as a three-letters abbreviation using the current user settings.
- <%**loc_dddd**%> - Indicates day of week as its full name using the current user settings.
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**loc_mmm**%> - Indicates month as a three-letters abbreviation using the current user settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user settings.
- <%**loc_gg**%> - Indicates period/era using the current user settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user settings.
- <%**loc_time**%> - Indicates the time using the current user settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user settings.
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user settings.
- <%**loc_tsep**%> - Indicates the time separator using the current user settings.
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

The following VB sample displays the next day as "Tue, May 31, 2005":

```
With G2antt1.Chart
   Debug.Print .FormatDate(.NextDate(.FirstVisibleDate, exDay, 2), "<%ddd%>,
<%mmmm%> <%d%>, <%yyyy%>")
End With
```

The following C++ sample displays the next day as "Tue, May 31, 2005":

```
CChart chart = m_g2antt.GetChart();
DATE d = chart.GetNextDate( V2D( &chart.GetFirstVisibleDate() ), 4096, COleVariant(
(long)1 ) );
CString strFormat = chart.GetFormatDate( d, "<%ddd%>, <%mmmm%> <%d%>,
<%yyyy%>" );
OutputDebugString( strFormat );
```

where the V2D function converts a Variant expression to a DATE expression:

```
static DATE V2D( VARIANT* pvtDate )
{
   COleVariant vtDate;
   vtDate.ChangeType( VT_DATE, pvtDate );
   return V_DATE( &vtDate );
}
```

The following VB.NET sample displays the next day as "Tue, May 31, 2005":

```
With AxG2antt1.Chart
   Debug.Write(.FormatDate(.NextDate(.FirstVisibleDate, EXG2ANTTLib.UnitEnum.exDay,
2), "<%ddd%>, <%mmmm%> <%d%>, <%yyyy%>"))
End With
```

The following C# sample displays the next day as "Tue, May 31, 2005":

```
DateTime d = Convert.ToDateTime(
axG2antt1.Chart.get_NextDate(Convert.ToDateTime(axG2antt1.Chart.FirstVisibleDate),
EXG2ANTTLib.UnitEnum.exDay, 1) );
String strFormat = axG2antt1.Chart.get_FormatDate(d, "<%ddd%>, <%mmmm%>
<%d%>, <%yyyy%>");
```

```
System.Diagnostics.Debug.Write(strFormat);
```

The following VFP sample displays the next day as "Tue, May 31, 2005":

```
With thisform.G2antt1.Chart
    wait window nowait .FormatDate(.NextDate(.FirstVisibleDate, 4096, 2), "<%ddd%>,
<%mmmm%> <%d%>, <%yyyy%>")
EndWith
```

property Chart.GridLineStyle as GridLinesStyleEnum

Retrieves or sets a value that indicates style for the gridlines being shown in the chart area.

Type	Description
GridLinesStyleEnum	A GridLinesStyleEnum expression that indicates the style to show the grid lines in the chart view part of the control.

By default, the GridLineStyle property is exGridLinesDot. The GridLineStyle property has effect only if the chart's DrawGridLines property is not zero. Use the DrawGridLines property of the Level object to show the vertical grid lines for the specified level. Use the GridLineColor property of the Level object to specify the color for vertical grid lines in the chart area. Use the GridLineStyle property of the Level object to specify the style for vertical grid lines in the chart area. Use the GridLineColor property to specify the color for grid lines. Use the DrawGridLines property to specify whether the control draws the grid lines in the items area. Use the DrawLevelSeperator property to draw lines between levels inside the chart's header. Use the DrawTickLines property to specify whether the grid lines between time units in the level are visible or hidden. Use the MarkTodayColor property to specify the color to mark the today date.

method Chart.GroupUndoRedoActions (Count as Long)

Groups the next to current Undo/Redo Actions in a single block.

Type	Description
Count as Long	A Long expression that specifies the number of entries being grouped in a single block of actions, in the Undo/Redo queue.

A block may hold multiple Undo/Redo actions. Use the GroupUndoRedoActions method to group two or more entries in the Undo/Redo queue in a single block, so when a next Undo/Redo operation is performed, multiple actions may occur. For instance, moving several bars in the same time (multiple bars selection) is already recorded as a single block. Use the UndoRedoQueueLength property to specify the number of entries that Undo/Redo queue may store.

A block starts with StartBlock and ends with EndBlock when listed by UndoListAction/RedoListAction property as in the following sample:

```
StartBlock
MoveBar;1;E
MoveBar;2;E
MoveBar;3;
MoveBar;4;
EndBlock
```

property Chart.HistogramBackColor as Color

Specifies the background color of the chart's histogram.

Type	Description
Color	A Color expression that defines the histogram's background color. The last 7 bits in the high significant byte of the color indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

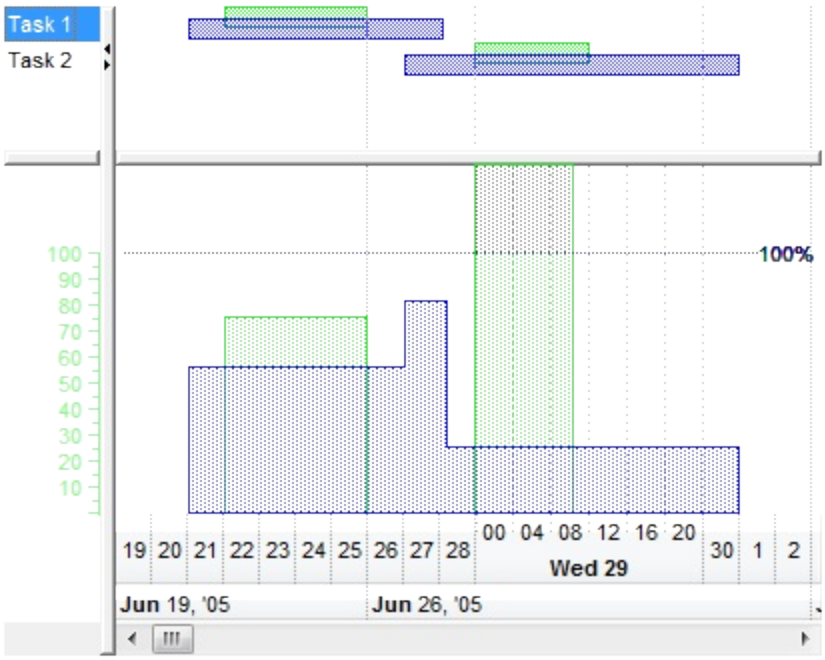
By default, the HistogramBackColor is changed when the BackColor property is changed. Use the HistogramVisible property to show or hide the histogram. Use the HistogramHeight property to specify at runtime the height of the histogram. Use the HistogramPattern and HistogramColor property to define the shape, the pattern, the color or the skinning object to be used in defining your histogram for specifies type of bar. The control allows to show workload and capacity by a histogram that automatically adapts to your current planning situation. The histogram may show the overloads and subloads for visible bars, or for selected bars. Over-loads and Sub-loads can be shown in several ways using curves, patterns or colors in the same histogram. They are updated interactively, so as user moves or resizes a bar, the histogram is updated automatically.

property Chart.HistogramHeaderVisible as Boolean

Specifies whether a copy of chart's header is displayed in the bottom side of the histogram.

Type	Description
Boolean	A Boolean expression that specifies whether a mirror copy of the chart's header is being shown in the bottom side of the histogram.

By default, the HistogramHeaderVisible property is False. Use the HistogramHeight property to specify the height in pixels of the histogram. Use the HeaderVisible property to show or hide the control's header. The HistogramBackColor property specifies the histogram's background color. Use the OnResizeControl property on exDisableHistogram property to specify whether the user can resizes the histogram at runtime.



property Chart.HistogramHeight as Long

Specifies whether the height of the chart's histogram.

Type	Description
Long	A Long expression that specifies the height in pixels, of the histogram.

By default, the HistogramHeight property is 0. Use the HistogramVisible property to show or hide the histogram. The control allows to show workload and capacity by a histogram that automatically adapts to your current planning situation. The histogram may show the overloads and subloads for visible bars, or for selected bars. Over-loads and Sub-loads can be shown in several ways using curves, patterns or colors in the same histogram. They are updated interactively, so as user moves or resizes a bar, the histogram is updated automatically. Use the HistogramBackColor property to specify the histogram's background color. Use the HistogramPattern and HistogramColor property to define the shape, the pattern, the color or the skinning object to be used in defining your histogram for specifies type of bar. Use the OnResizeControl property on exDisableHistogram to prevent resizing the histogram at runtime. The HistogramBoundsChanged event notifies your application when the location and the size of the chart's histogram is changed, so you can use it to add your legend for the histogram in a panel component.

property Chart.HistogramUnitCount as Long

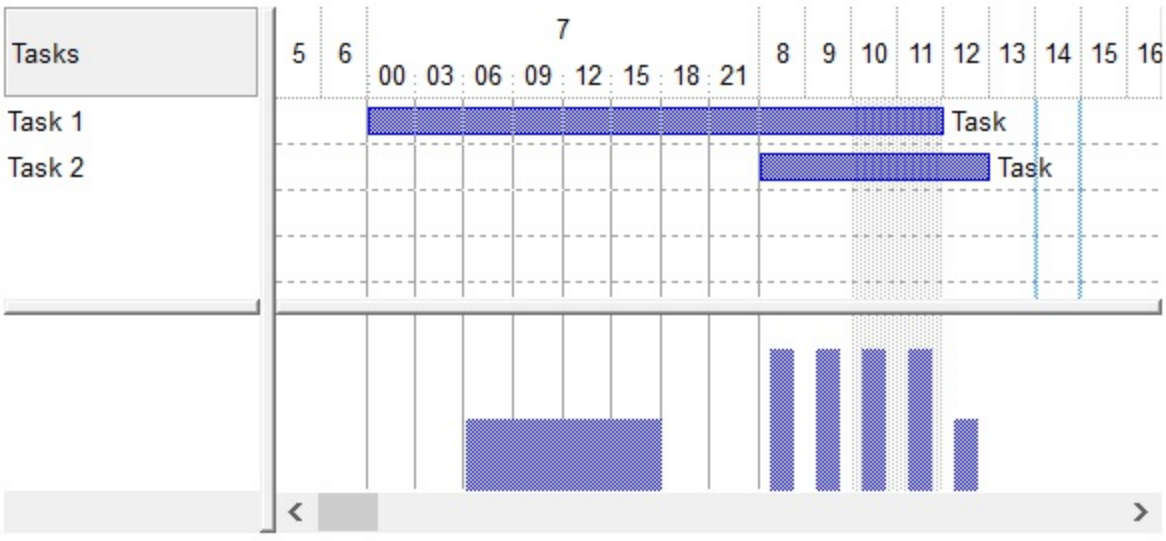
Specifies the time-scale count to determine the effort of bars with variable-effort (effort of expression/string type)

Type	Description
Long	A long expression that specifies the count to determine the effort of bars with variable-effort (effort of expression/string type)

By default, the HistogramUnitCount property is equivalent with ResizeUnitCount property. The HistogramUnitScale property defines the count to determine the effort of bars with variable-effort (effort of expression/string type). The HistogramUnitScale / HistogramUnitCount property **have effect only for** item-bars with ItemBar(exBarEffort) property to refer an expression (defines an variable- effort). The **value** keyword indicates the date-time being queried, the **start** and **end** keywords specify the starting and ending points of the bar as indicated by exBarStart and exBarEnd fields in the ItemBar property. For instance, the exBarEffort on "weekday(value) in (0,6) ? 0 : 2", means that that effort to do the job is 2 for any day in the task, excepts the Sundays(0) and Saturdays(6) (weekend.

For instance, the "(hour(value) > 5 and hour(value) < 18) ? 2 : 0" indicates that the bar's effort is 2 for any hour between 6AM and 18PM, and 0 for rest.

The following screen show shows the correct histogram (HistogramUnitScale property on **exHour**):



property Chart.HistogramUnitScale as UnitEnum

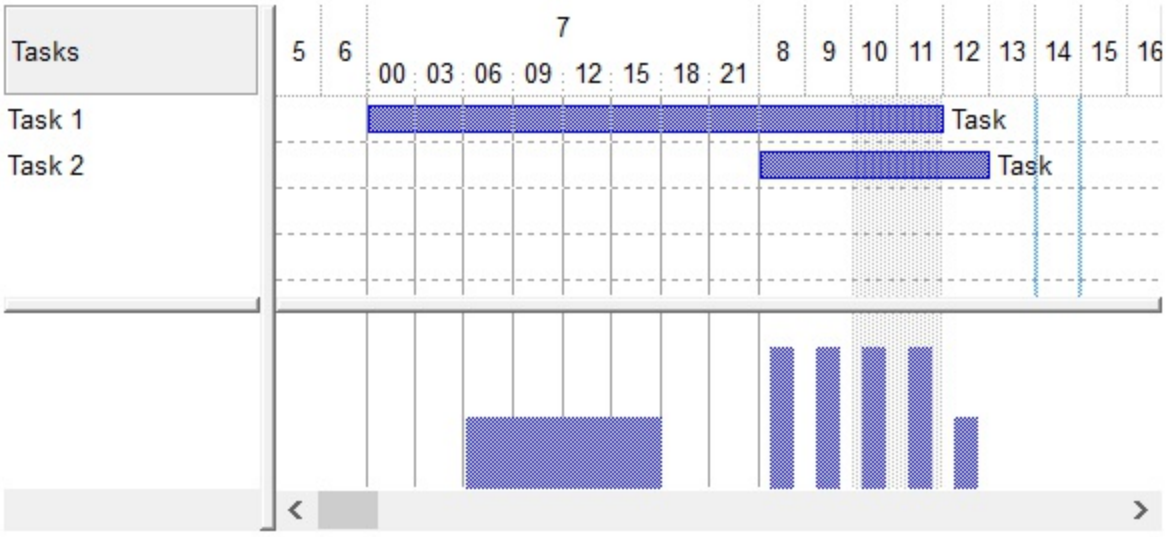
Specifies the time-scale unit to determine the effort of bars with variable-effort (effort of expression/string type)

Type	Description
UnitEnum	An UnitEnum expression that defines the effort of bars with variable-effort (effort of expression/string type)

By default, the HistogramUnitScale property is equivalent with ResizeUnitScale property. The HistogramUnitScale property defines the time-scale unit to determine the effort of bars with variable-effort (effort of expression/string type). The HistogramUnitScale / HistogramUnitCount property **have effect only for** item-bars with ItemBar(exBarEffort) property to refer an expression (defines an variable-effort). The **value** keyword indicates the date-time being queried, the **start** and **end** keywords specify the starting and ending points of the bar as indicated by exBarStart and exBarEnd fields in the ItemBar property. For instance, the exBarEffort on "weekday(value) in (0,6) ? 0 : 2", means that that effort to do the job is 2 for any day in the task, excepts the Sundays(0) and Saturdays(6) (weekend.

For instance, the "(hour(value) > 5 and hour(value) < 18) ? 2 : 0" indicates that the bar's effort is 2 for any hour between 6AM and 18PM, and 0 for rest.

The following screen show shows the correct histogram (HistogramUnitScale property on **exHour**):



property Chart.HistogramValue (Date as Variant, [Name as Variant], [Group as Variant]) as Double

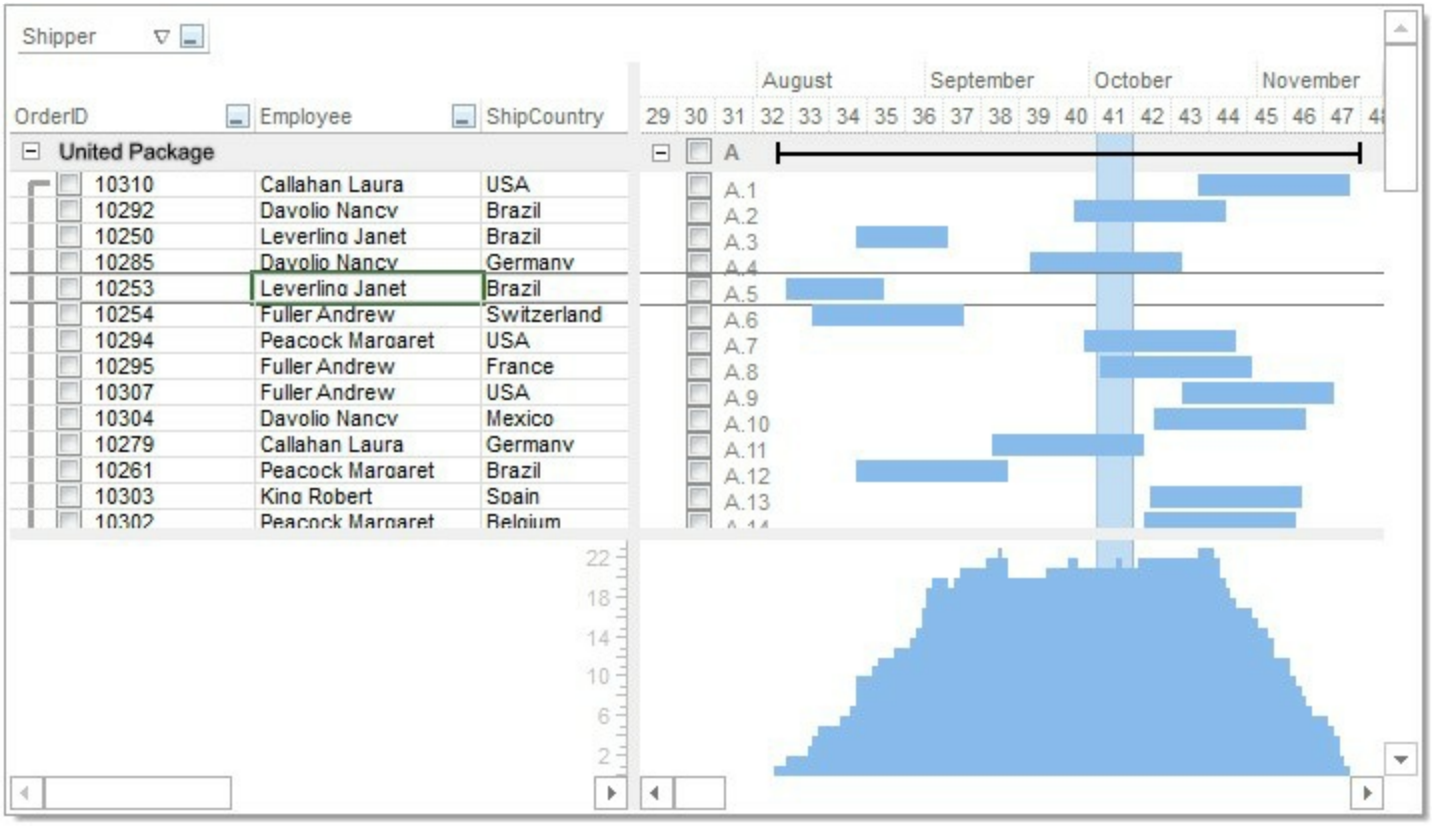
Gets the value in the histogram at specified date-time, for giving type of bars or/and groups.

Type	Description
Date as Variant	A DATE expression that specifies the date-time being queried for its value in the control's histogram, or a s STRING expression such as "**min**" to get the minimum value excepts 0 in the histogram, and "**max**" to get the maximum value in the histogram. The min and max flags determines the minimum and maximum values from the displayed values not from the entire chart. The Date parameter should be a value between first and last visible date in the chart, else the HistogramValue property gets -1 result.
Name as Variant	A String expression that specifies the name (or a list of names separated by comma(,) character) of the bar to be queried for its value in the histogram. The Name parameter has no effect, if the chart's histogram displays a single type of the bar in the histogram. The BarName parameter of the AddBar method or ItemBar(exBarName) value specifies the name of displayed bar. If Name parameter is missing or empty, all bars are queried. For instance, if the histogram displays the curves for multiple type of bars, the Name parameter indicates the name of the bars to be queried separated by comma(,) character. For instance, "**Task,Summary**" queries the values for Task or Summary bars.
Group as Variant	A Long expression that specifies the index of the group of bars in the histogram to be queried, or a String expression that indicates the list of group(index) to be queried. The Group parameter has no effect, if the chart's histogram displays no groups in the histogram. Groups are usually displayed when your data is shown as a tree. If missing or empty, all groups are queried. For instance, "**0,1**" queries the values for first and second group of bars.
Double	A DOUBLE expression that specifies the value of the specified type/bar, group and date-time in the histogram. The HistogramValue property returns -1 if any error

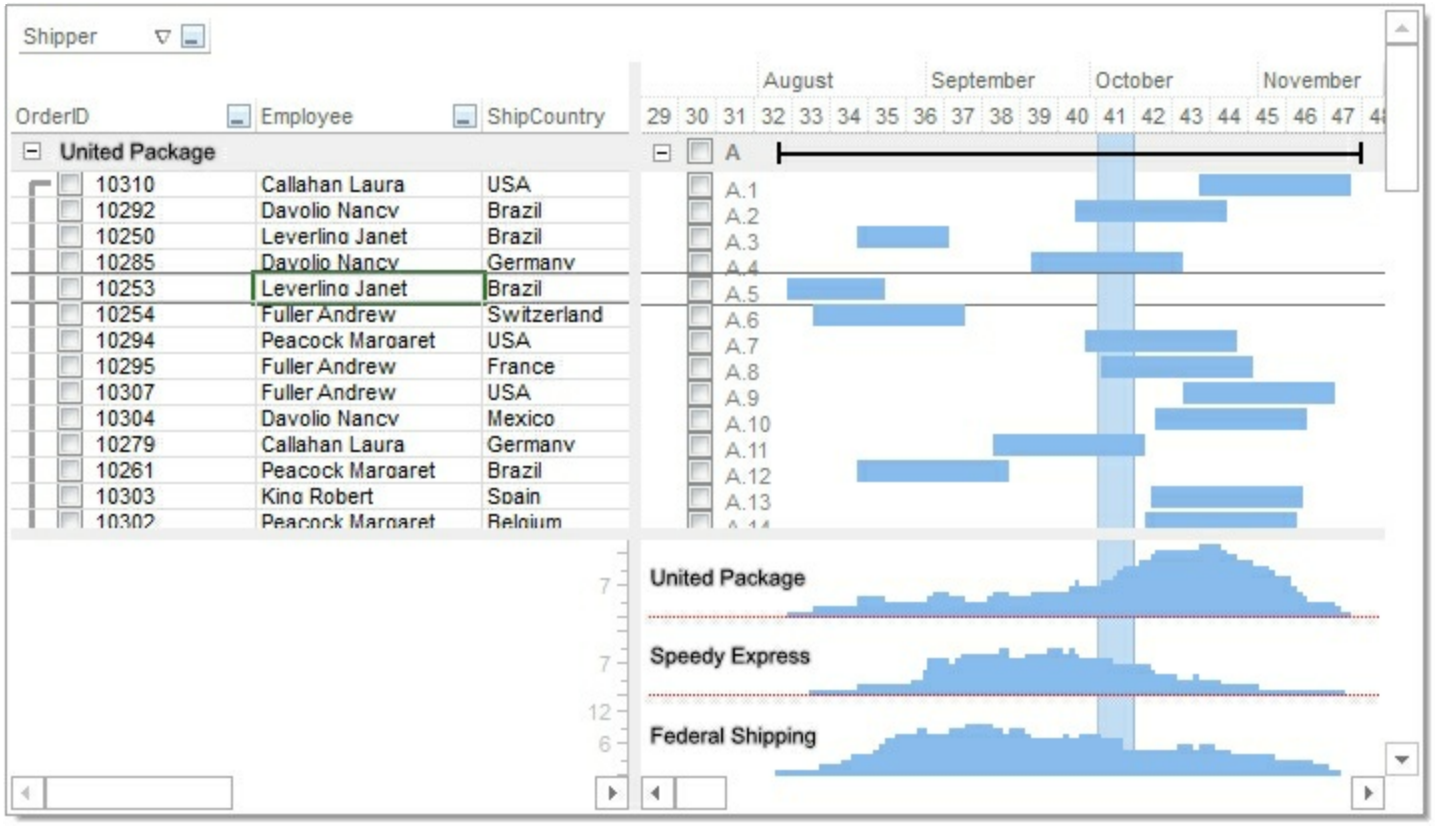
occurs. For instance, the histogram is not visible, a date-time that's not visible in the control's client area, and so on.

The HistogramValue property returns valid values while the control displays the control's histogram. Currently, the HistogramValue property returns -1, if the chart displays no histogram (HistogramVisible property is False, or HistogramHeight property is 0).

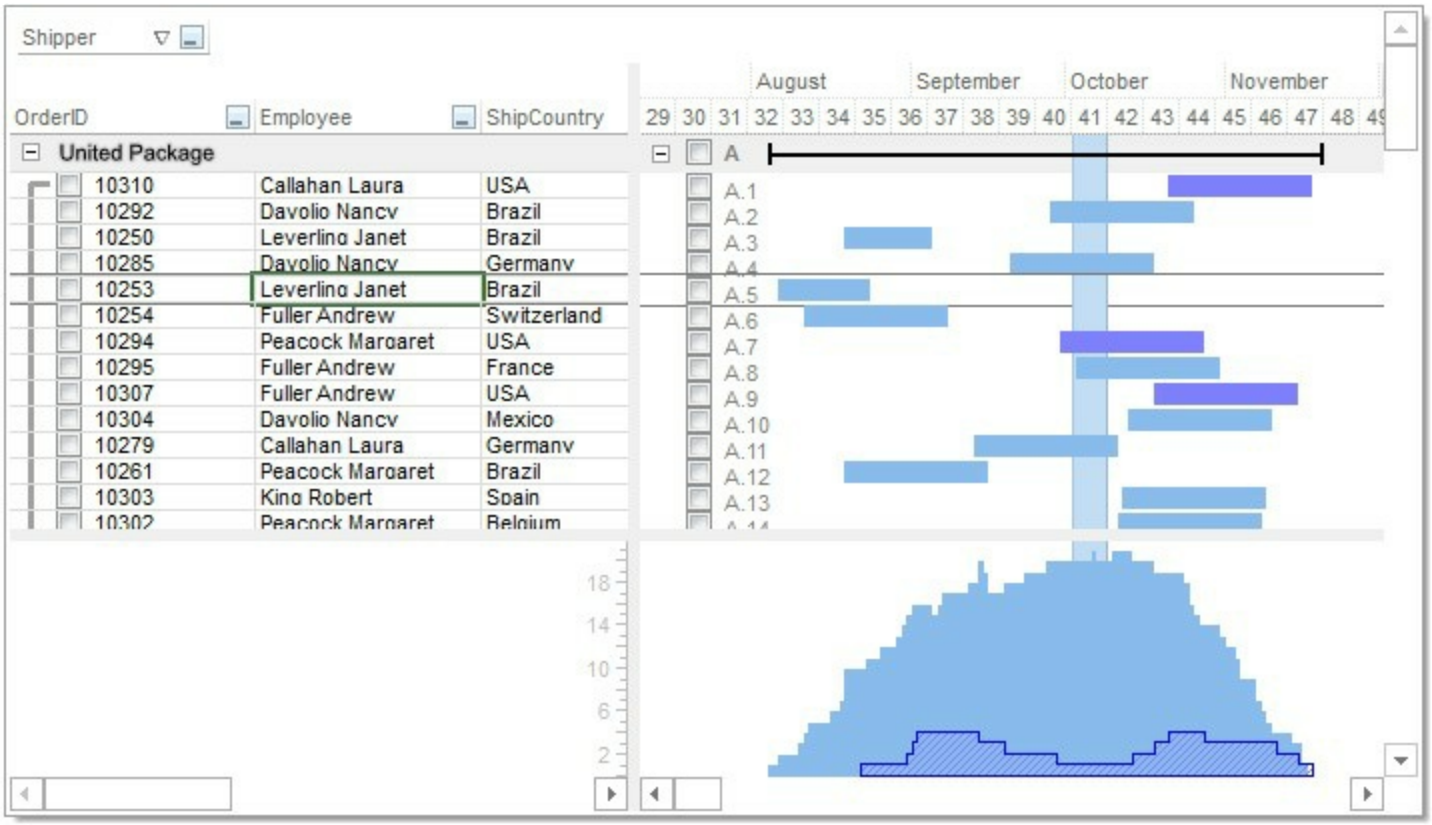
The following screen shot shows the histogram with a single task and with no grouping, where Name or Group parameters has no effect:



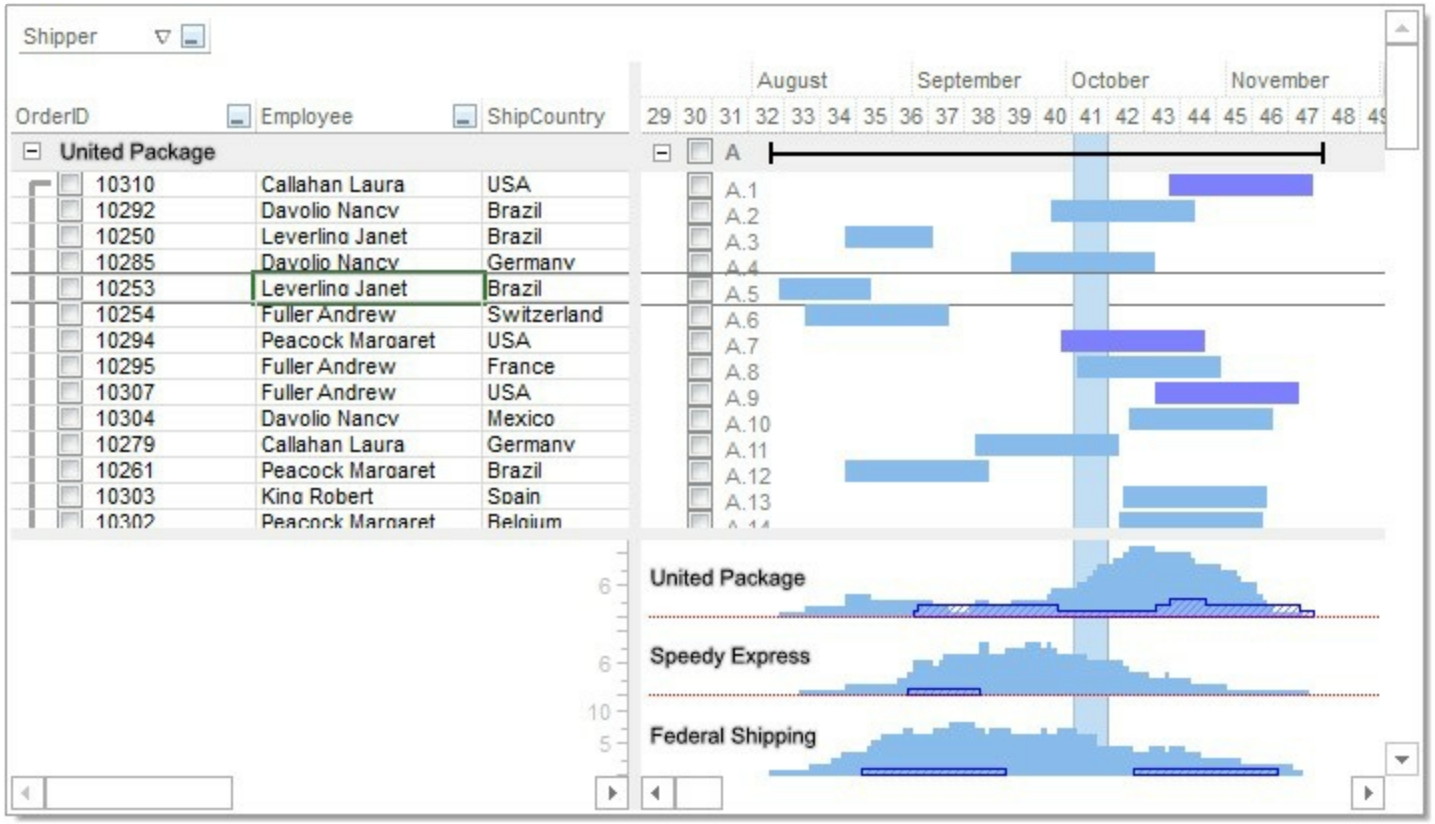
The following screen shot shows the histogram with a single task but with grouping (3 groups, United Package, Speedy Express, Federal Shipping), where Name has no effect, instead Group parameter could be 0, 1, 2 (empty or any combination of them)



The following screen shot shows the histogram for two tasks but with no grouping , where Name could be specified , instead Group parameter has no effect:



The following screen shot shows the histogram for two tasks but with grouping , where Name could be specified, and Group parameter could be 0, 1, 2 (empty or any combination of them)


Shipper
OrderID
Employee
ShipCountry
August
September
October
November
29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
United Package
10310 Callahan Laura USA
10292 Davolio Nancy Brazil
10250 Leverling Janet Brazil
10285 Davolio Nancy Germany
10253 Leverling Janet Brazil
10254 Fuller Andrew Switzerland
10294 Peacock Margaret USA
10295 Fuller Andrew France
10307 Fuller Andrew USA
10304 Davolio Nancy Mexico
10279 Callahan Laura Germany
10261 Peacock Margaret Brazil
10303 King Robert Spain
10302 Peacock Margaret Belgium
A
A.1
A.2
A.3
A.4
A.5
A.6
A.7
A.8
A.9
A.10
A.11
A.12
A.13
6
6
10
5
United Package
Speedy Express
Federal Shipping

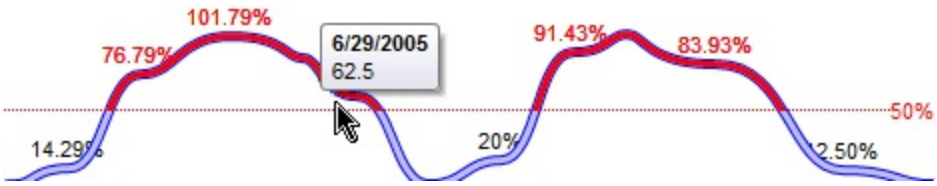
property Chart.HistogramValueFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Double

Retrieves the value from the histogram at the cursor position.

Type	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in client coordinates.
Double	A double expression that specifies the value in the histogram from the specified position.

Use the HistogramValueFromPoint property to determine the value from the specified position in the histogram. The HistogramValueFromPoint property works only if the ShowHistogramValues property is a not-empty and a valid expression. If the returned value is different than 0 you can use the ShowToolTip property to display your customized tooltip. The DateFromPoint property determines the date expression from the point. The HistogramVisible property specifies whether the chart shows the histogram for selected bars. The ShowHistogramValues property specifies the formula that returns the color to display the selected values in the histogram for specified type of bar.

The following screen shows shows a tooltip as the user moves the cursor over the chart's histogram:



property Chart.HistogramView as HistogramViewEnum

Specifies the list of items being included in the histogram.

Type	Description
HistogramViewEnum	A HistogramView type that specifies the list of items being displayed in the histogram.

By default, the HistogramView property is exHistogramVisibleItems, which makes the control to display the histogram for visible items only. The HistogramView property specify the items being represented in the histogram view. Use the HistogramVisible property to show the histogram bar/view. Use the HistogramHeight property to specify at runtime the height of the histogram bar being displayed on the bottom side of the control.

If the HistogramView property is:

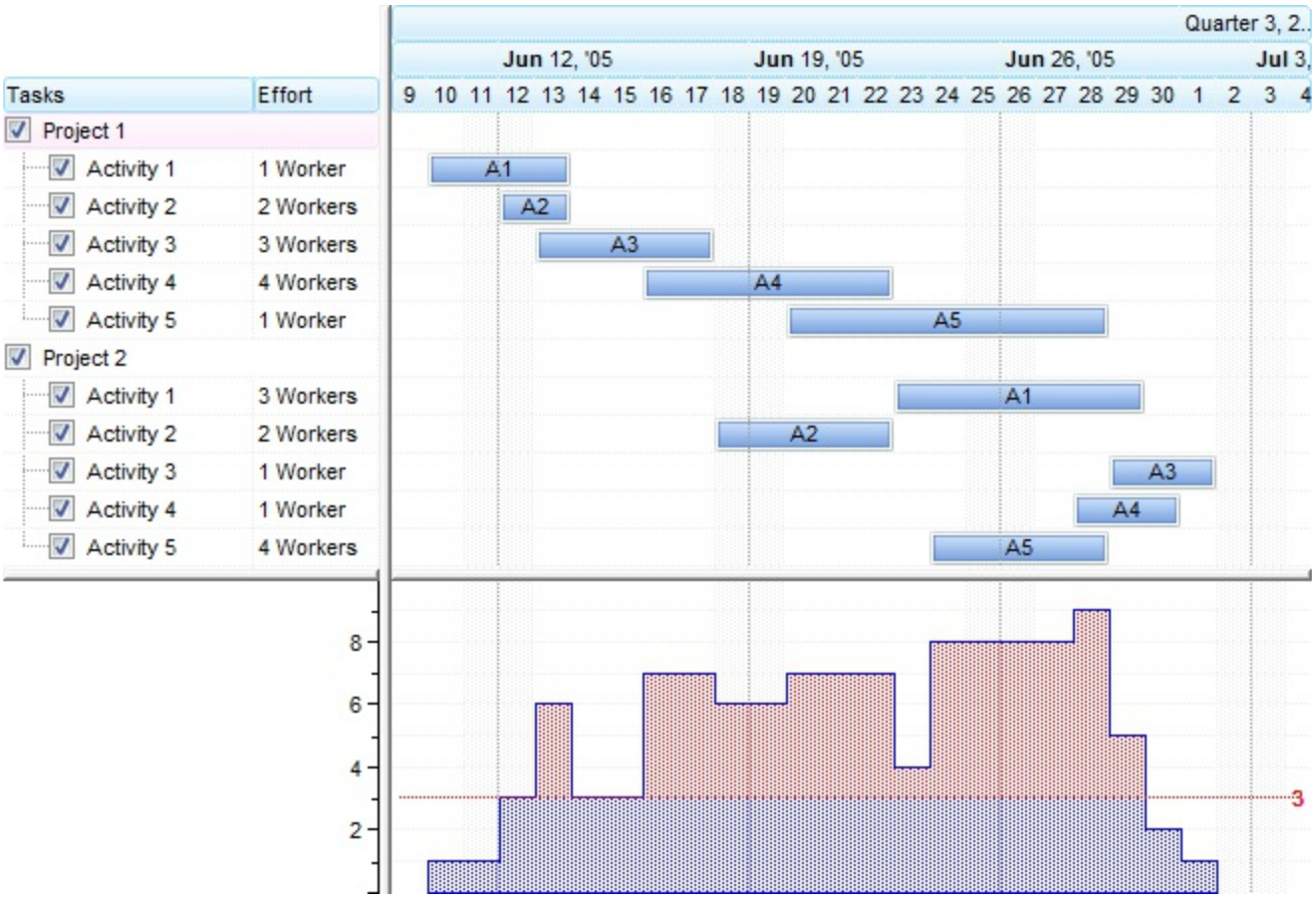
- **exHistogramVisibleItems,** the histogram includes only visible items. This can be combined with exHistogramLeafItems, exHistogramRecLeafItems or exHistogramNoGrouping. The Histogram is updated as soon as the control changes its first visible item, in other words the control gets vertically scrolled.
- **exHistogramSelectedItems,** The histogram is shown for the selected items only. Use the SingleSel property to specify whether the control can select multiple items. Use the SelectOnClick property to disable selecting new items when the user clicks the chart area. This can be combined with exHistogramLeafItems, exHistogramRecLeafItems or exHistogramNoGrouping. The Histogram is updated as soon as the selection is changed.
- **exHistogramSelectedBars,** The histogram is shown for the selected bars only. The ItemBar(exBarSelected) property specifies whether a bar is selected or unselected. The ChartSelectionChanged event notifies the application once a new bar is selected or unselected. This can be combined with exHistogramLeafItems, exHistogramRecLeafItems or exHistogramNoGrouping. The Histogram is updated as soon as the selection in the chart is changed.
- **exHistogramCheckedItems**, The histogram is shown for the checked items only. You *must* combine this with exHistogramUnlockedItems, exHistogramLockedTopItems or exHistogramLockedBottomItems. Also, this can be combined with exHistogramLeafItems, exHistogramRecLeafItems or exHistogramNoGrouping. Use the CellState property to specify the state of the cell. The histogram includes only items that have the CellState property on 1 (locked and unlocked items). By default, the check box should be on the first column (the column with the index 0). Use the high word of the HistogramView property to specify a different column. For instance, if you need to display the histogram based on the check boxes of the column index 5, the HistogramView property should be 0x5**0000 + exHistogramCheckedItems + exHistogramUnlockedItems**. *Another sample, if the HistogramView property is*

exHistogramCheckedItems + exHistogramLockedBottomItems the histogram shows only the checked items in the bottom locked area. The Histogram is updated as soon as the user changes the state of the cell's check box

- **exHistogramUnlockedItems**, The histogram is shown only for unlocked items. Use the AddItem/InsertItem methods to add unlocked items. This option can be combined with exHistogramCheckedItems, exHistogramLockedTopItems or exHistogramLockedBottomItems. *For instance, if the HistogramView property is exHistogramUnlockedItems + exHistogramLockedTopItems the histogram shows all the items in the unlocked plus the items in the top locked area.*
- **exHistogramLockedTopItems**, The histogram is shown only for locked items in the top side of the control. Use the LockedItemCount property to specify how many items are in the locked area. This option can be combined with exHistogramCheckedItems, exHistogramUnlockedItems or exHistogramLockedBottomItems. *For instance, if the HistogramView property is exHistogramUnlockedItems + exHistogramLockedTopItems the histogram shows all the items in the unlocked plus the items in the top locked area.*
- **exHistogramLockedBottomItems**, The histogram is shown only for locked items in the bottom side of the control. Use the LockedItemCount property to specify how many items are in the locked area. This option can be combined with exHistogramCheckedItems, exHistogramUnlockedItems or exHistogramLockedTopItems. *For instance, if the HistogramView property is exHistogramUnlockedItems + exHistogramLockedBottomItems the histogram shows all the items in the unlocked plus the items in the bottom locked area.*
- **exHistogramAllItems**, The histogram is shown for all items, locked and unlocked items too. The exHistogramAllItems is a shortcut for the exHistogramUnlockedItems + exHistogramLockedTopItems + exHistogramLockedBottomItems. This can be combined with exHistogramLeafItems, exHistogramRecLeafItems or exHistogramNoGrouping.
- **exHistogramLeafItems**, The histogram shows the bars for leaf items, in other words, the item itself if contains no child items, or all child items that contains no other child items. Use this option to include in the histogram the bars in the child items too.
- **exHistogramRecLeafItems**, The histogram shows all bars for all recursive leaf items, so all child leaf items are displayed. Use this option to include in the histogram the bars in all child items (recursively) too.
- **exHistogramNoGrouping**, If present, the histogram shows all bars without grouping based on the item's parent, and so all bars shares the same space for the histogram. If missing, the bars included in the histogram are grouped based on their parents, and each group has allocated a space in the histogram, so each group is shown separately.

The following screen shot shows the items grouped by their parents (the exHistogramNoGrouping option **is not set**)

The following screen shot shows the items grouped by their parents (the exHistogramNoGrouping option **is set**)



Only the exHistogramVisibleItems and exHistogramSelectedItems shows the histogram by grouping the items based on their common parent. The rest of options shows the histogram in the same space, without grouping.

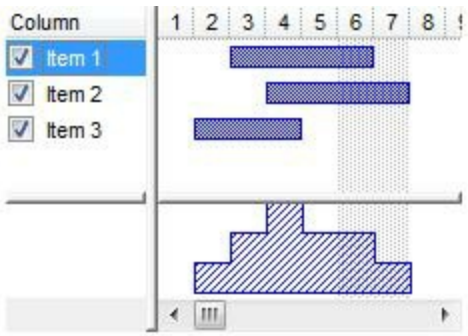
Please follow the steps in order to view your bars in the histogram.

1. Changes the **HistogramVisible** property on True (by default, it is False). After setting the HistogramVisible property on True, the control shows a horizontal splitter in the bottom side of the control.
2. Adjusts the height of the histogram view using the **HistogramHeight** property (by default it is 0). After setting the HistogramHeight property on a value greater than 0, the control shows a the histogram view in the bottom side of the control.
3. Changes the **HistogramPattern** **or/and** **HistogramColor** property, else no bars will be shown in the histogram. The HistogramPattern/HistogramColor properties belong to a Bar object. For instance the Chart.Bars("Task").HistogramPattern = exPatternDot, specifies that the Task bars will be represented in the histogram using the exPatternDot pattern ( )

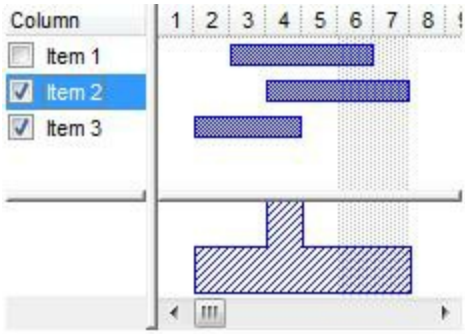
The followings are optional properties that you can set in order to customize your histogram:

- The **HistogramType** property indicates the type of the histogram being displayed for a specified bar.
- Use the **HistogramView** property to specify the items being represented in the histogram view. By default, only visible items are displayed in the histogram. For instance, using the HistogramView property you can select the items being represented in the histogram
- Use the HistogramBackColor property to specify the histogram's background color.

The following screen shot shows the histogram for all items (as they are all checked in the first column):



The following screen shot shows the histogram for the last 2 items (Item 2, Item 3 as they are checked):



The following VB sample shows the histogram for "Task" bars in the checked items:

```
With G2antt1
   .BeginUpdate
   With .Chart
      .FirstVisibleDate = #1/1/2001#
      .HistogramVisible = True
      .HistogramView = exHistogramCheckedItems
      .HistogramHeight = 32
      .Bars.Item("Task").HistogramPattern = exPatternBDiagonal
   End With
   .Columns.Add("Column").Def(exCellHasCheckBox) = True
   With .Items
      .AddBar .AddItem("Item 1"),"Task",#1/3/2001#,#1/5/2001#
      h = .AddItem("Item 2")
      .AddBar h,"Task",#1/4/2001#,#1/7/2001#
      .CellState(h,0) = 1
      h = .AddItem("Item 3")
      .AddBar h,"Task",#1/2/2001#,#1/5/2001#
      .CellState(h,0) = 1
   End With
   .EndUpdate
End With
```

The following VB.NET sample shows the histogram for "Task" bars in the checked items:

```
Dim h
With AxG2antt1
   .BeginUpdate
   With .Chart
      .FirstVisibleDate = #1/1/2001#
```

```
    .HistogramVisible = True
    .HistogramView = EXG2ANTTLib.HistogramViewEnum.exHistogramCheckedItems
    .HistogramHeight = 32
    .Bars.Item("Task").HistogramPattern = EXG2ANTTLib.PatternEnum.exPatternBDiagonal
  End With
  .Columns.Add("Column").Def(EXG2ANTTLib.DefColumnEnum.exCellHasCheckBox) =
True
  With .Items
    .AddBar .AddItem("Item 1"),"Task",#1/3/2001#,#1/5/2001#
    h = .AddItem("Item 2")
    .AddBar h,"Task",#1/4/2001#,#1/7/2001#
    .CellState(h,0) = 1
    h = .AddItem("Item 3")
    .AddBar h,"Task",#1/2/2001#,#1/5/2001#
    .CellState(h,0) = 1
  End With
  .EndUpdate
End With
```

The following C# sample shows the histogram for "Task" bars in the checked items:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
  var_Chart.FirstVisibleDate = "1/1/2001";
  var_Chart.HistogramVisible = true;
  var_Chart.HistogramView =
EXG2ANTTLib.HistogramViewEnum.exHistogramCheckedItems;
  var_Chart.HistogramHeight = 32;
  var_Chart.Bars["Task"].HistogramPattern =
EXG2ANTTLib.PatternEnum.exPatternBDiagonal;
(axG2antt1.Columns.Add("Column") as
EXG2ANTTLib.Column).set_Def(EXG2ANTTLib.DefColumnEnum.exCellHasCheckBox,true);
EXG2ANTTLib.Items var_Items = axG2antt1.Items;
  var_Items.AddBar(var_Items.AddItem("Item 1"),"Task","1/3/2001","1/5/2001",null,null);
  int h = var_Items.AddItem("Item 2");
  var_Items.AddBar(h,"Task","1/4/2001","1/7/2001",null,null);
  var_Items.set_CellState(h,0,1);
```

```
    h = var_Items.AddItem("Item 3");
    var_Items.AddBar(h,"Task","1/2/2001","1/5/2001",null,null);
    var_Items.set_CellState(h,0,1);
axG2antt1.EndUpdate();
```

The following C++ sample shows the histogram for "Task" bars in the checked items:

```
/*
    Copy and paste the following directives to your header file as
    it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

    #import "D:\\Windows\\System32\\ExG2antt.dll"
    using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
    var_Chart->PutFirstVisibleDate("1/1/2001");
    var_Chart->PutHistogramVisible(VARIANT_TRUE);
    var_Chart->PutHistogramView(EXG2ANTTLib::exHistogramCheckedItems);
    var_Chart->PutHistogramHeight(32);
    var_Chart->GetBars()->GetItem("Task")-
>PutHistogramPattern(EXG2ANTTLib::exPatternBDiagonal);
((EXG2ANTTLib::IColumnPtr)(spG2antt1->GetColumns()->Add(L"Column")))-
>PutDef(EXG2ANTTLib::exCellHasCheckBox,VARIANT_TRUE);
EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems();
    var_Items->AddBar(var_Items->AddItem("Item
1"),"Task","1/3/2001","1/5/2001",vtMissing,vtMissing);
    long h = var_Items->AddItem("Item 2");
    var_Items->AddBar(h,"Task","1/4/2001","1/7/2001",vtMissing,vtMissing);
    var_Items->PutCellState(h,long(0),1);
    h = var_Items->AddItem("Item 3");
    var_Items->AddBar(h,"Task","1/2/2001","1/5/2001",vtMissing,vtMissing);
    var_Items->PutCellState(h,long(0),1);
spG2antt1->EndUpdate();
```

The following VFP sample shows the histogram for "Task" bars in the checked items:

```
with thisform.G2antt1
    .BeginUpdate
    with .Chart
        .FirstVisibleDate = {^2001-1-1}
        .HistogramVisible = .T.
        .HistogramView = 4
        .HistogramHeight = 32
        .Bars.Item("Task").HistogramPattern = 6
    endwith
    .Columns.Add("Column").Def(0) = .T.
    with .Items
        .AddBar(.AddItem("Item 1"),"Task",{^2001-1-3},{^2001-1-5})
        h = .AddItem("Item 2")
        .AddBar(h,"Task",{^2001-1-4},{^2001-1-7})
        .CellState(h,0) = 1
        h = .AddItem("Item 3")
        .AddBar(h,"Task",{^2001-1-2},{^2001-1-5})
        .CellState(h,0) = 1
    endwith
    .EndUpdate
endwith
```

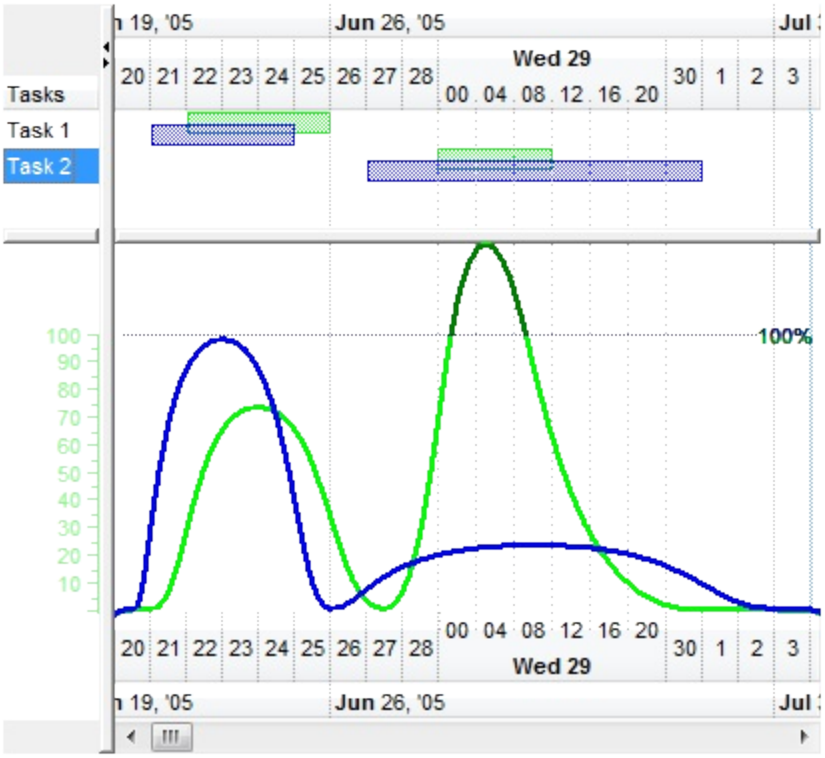
property Chart.HistogramVisible as Boolean

Specifies whether the chart's histogram layout is visible or hidden.

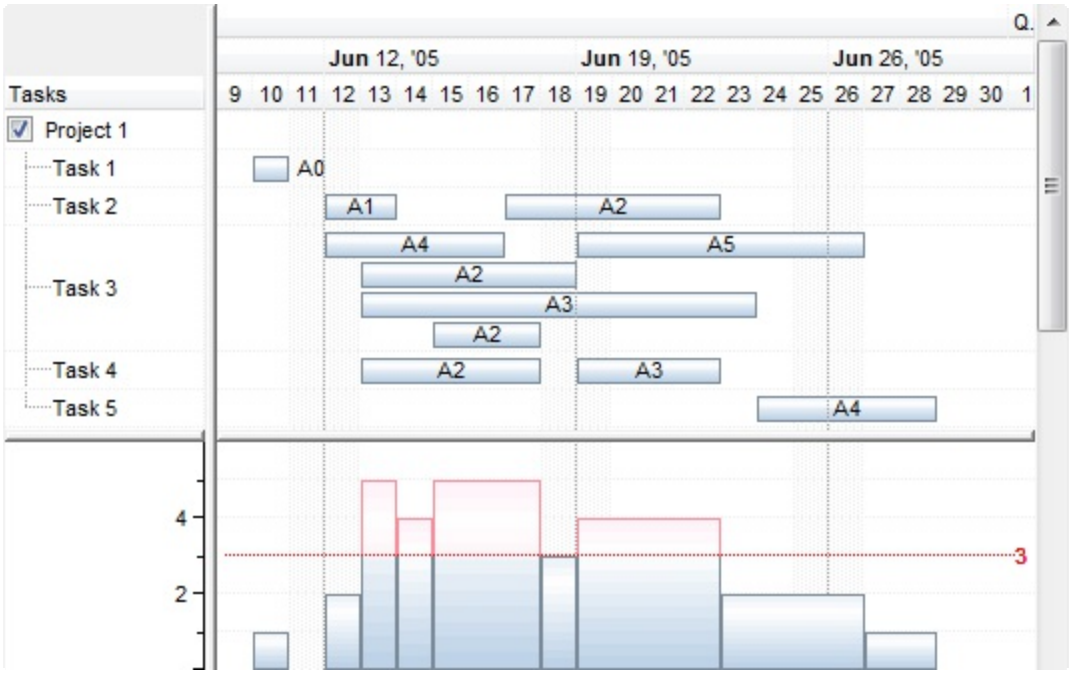
Type	Description
Boolean	A Boolean expression that specifies whether the histogram is visible or hidden. The True value indicates whether the histogram is visible, The False value indicates whether the histogram is hidden.

By default, the HistogramVisible property is False (hidden). The control allows showing workload and capacity by a histogram that automatically adapts to your current planning situation. The histogram may show the overloads, overallocations and subloads for visible bars, or for selected bars. Over-loads and Sub-loads can be shown in several ways using curves, patterns or colors in the same histogram. They are updated interactively, so as user moves or resizes a bar, the histogram is updated automatically. Use the OnResizeControl property on exDisableHistogram to prevent resizing the histogram at runtime. The HistogramBoundsChanged event notifies your application when the location and the size of the chart's histogram is changed, so you can use it to add your legend for the histogram in a panel component. The HistogramHeaderVisible property to show the chart's header in bottom part of the histogram. The ShowHistogramValues property specifies the formula that returns the color to display the selected values in the histogram for specified type of bar. The HistogramValueFromPoint property gets the value in the histogram from the specified location.

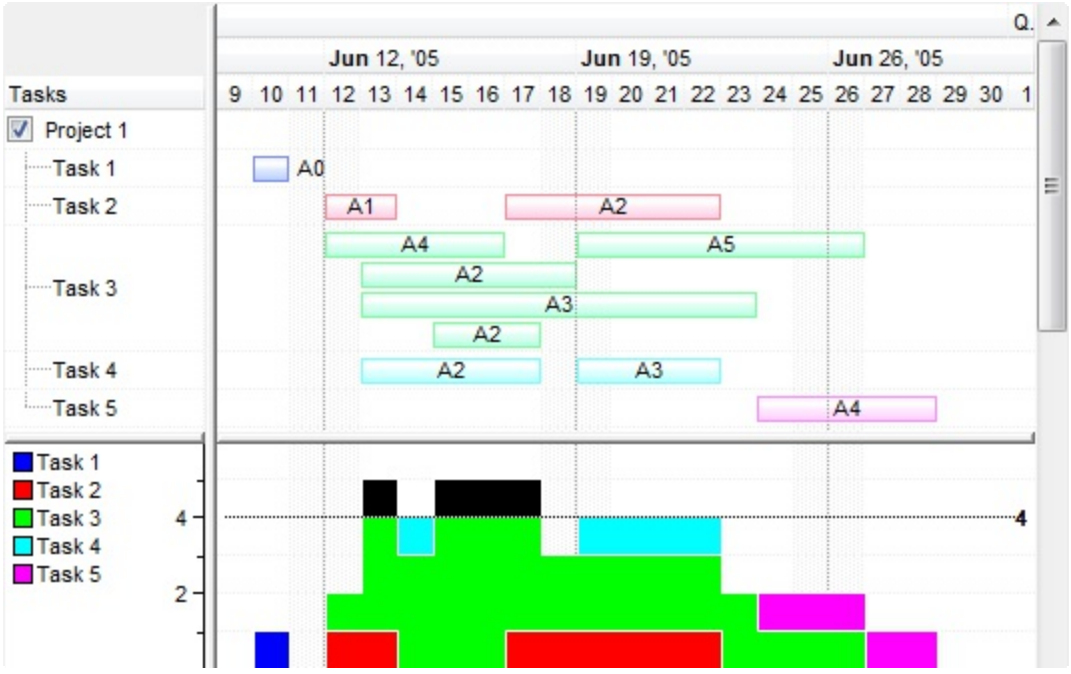
The following screen shot shows the *over-allocation* histogram



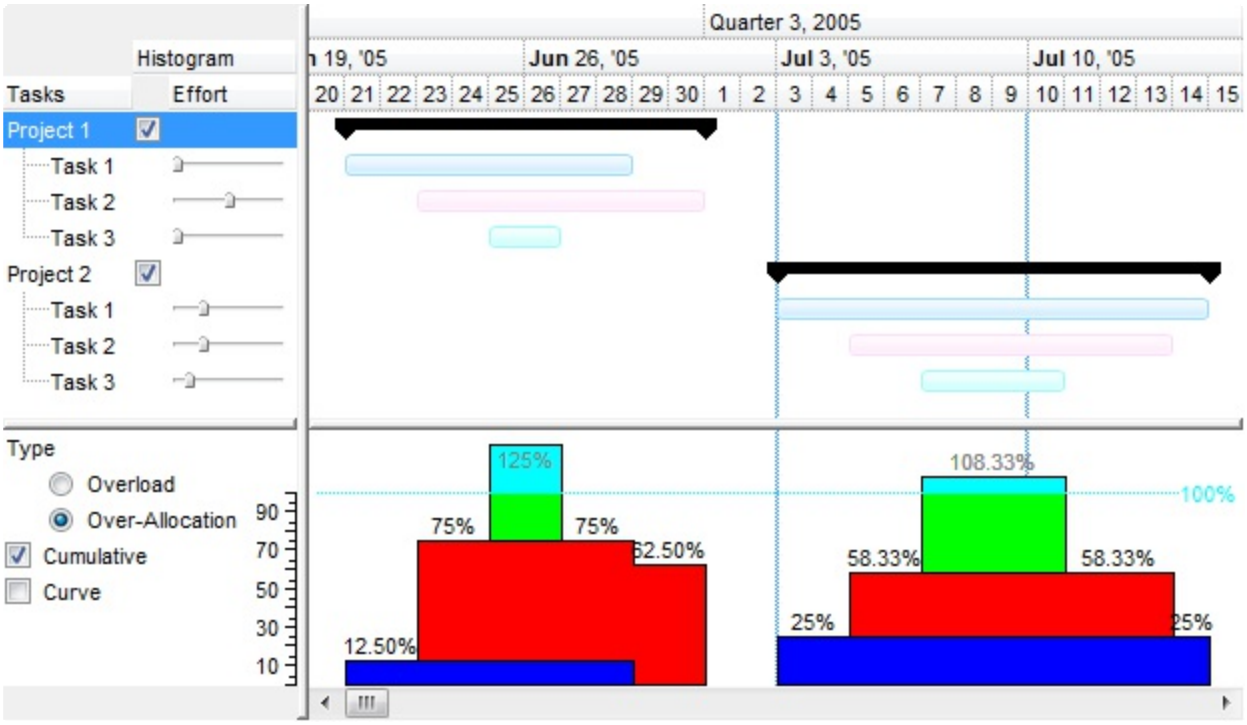
The following screen shot shows the *over-load* histogram:



The following screen shot shows the *cumulative* histogram:



The following screen shot shows the *cumulative* histogram, including the values, and the legend in the left side of the histogram:



Please follow the steps in order to view your bars in the histogram.

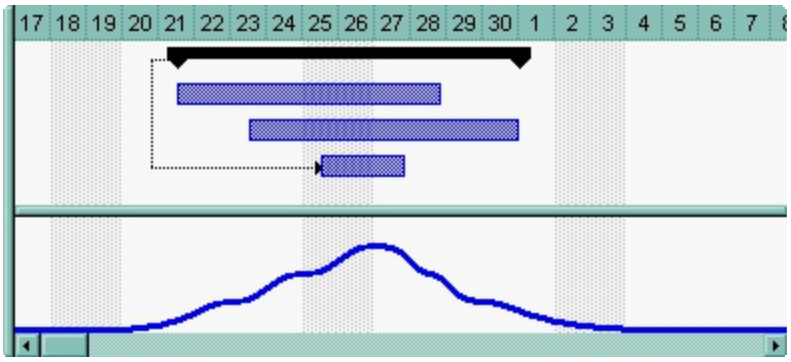
1. Changes the **HistogramVisible** property on True (by default, it is False). After setting the HistogramVisible property on True, the control shows a horizontal splitter in the bottom side of the control.
2. Adjusts the height of the histogram view using the **HistogramHeight** property (by default it is 0). After setting the HistogramHeight property on a value greater than 0, the control shows a the histogram view in the bottom side of the control.
3. Changes the **HistogramPattern** **or/and** **HistogramColor** property, else no bars will be shown in the histogram. The HistogramPattern/HistogramColor properties belong to a Bar object. For instance the Chart.Bars("Task").HistogramPattern = exPatternDot, specifies that the Task bars will be represented in the histogram using the exPatternDot pattern ( )

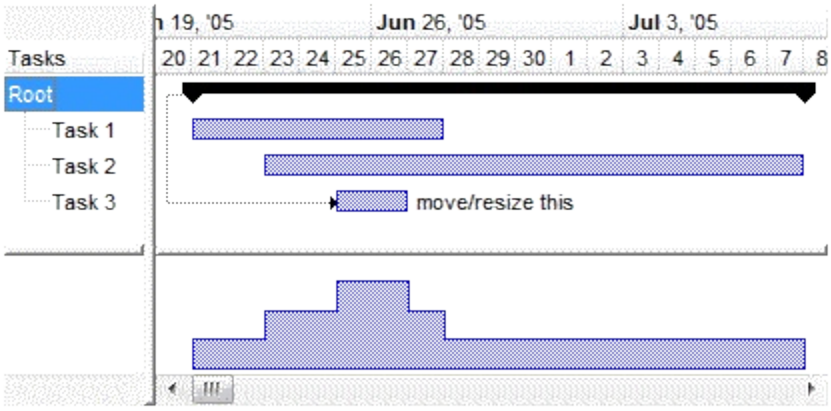
The followings are optional properties that you can set in order to customize your histogram:

- The **HistogramType** property indicates the type of the histogram being displayed for a specified bar.
- Use the **HistogramView** property to specify the items being represented in the histogram view. By default, only visible items are displayed in the histogram. For instance, using the HistogramView property you can select the items being represented in the histogram
- Use the HistogramBackColor property to specify the histogram's background color.
- Use the HistogramBoundsChanged event to resize the inside controls being displayed in the histogram
- Use the BeforeDrawPart/AfterDrawPart event to perform custom drawing over the chart's histogram.

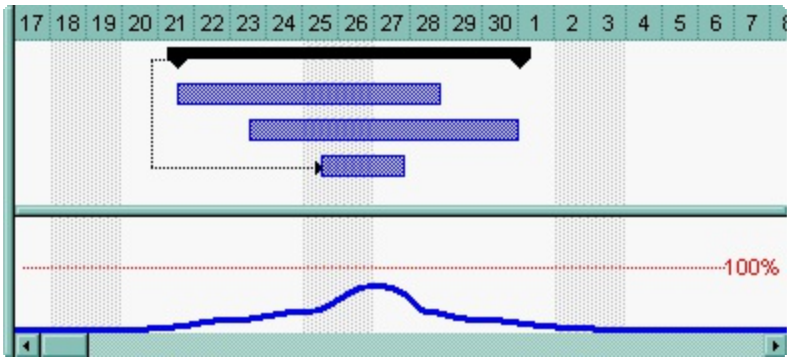
A bar is represented in the histogram only if HistogramPattern or HistogramColor property is set. If any of these are not set, the bar will not be represented in the histogram.

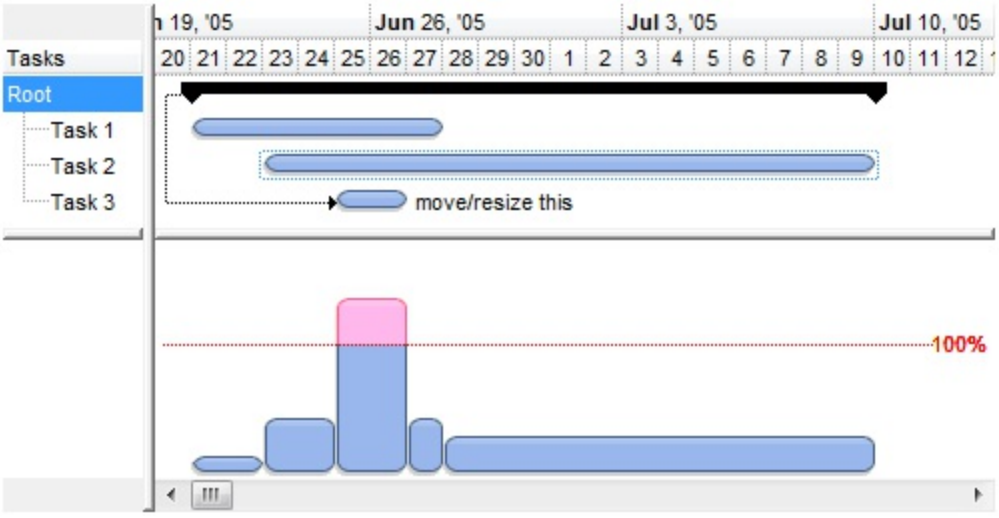
The following screen shot shows the **exHistOverload** histogram and how it is updated as soon the bars are moved or resized:

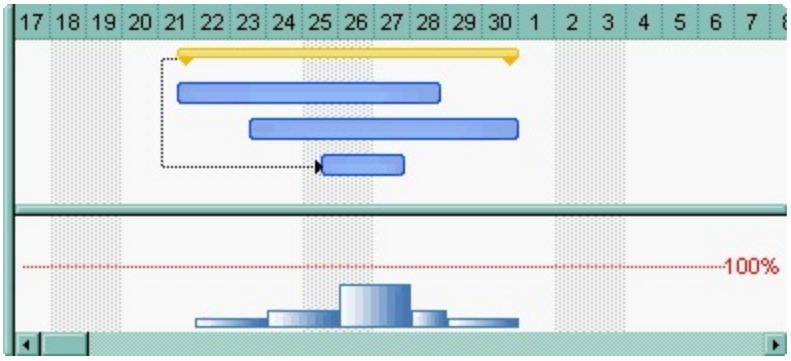


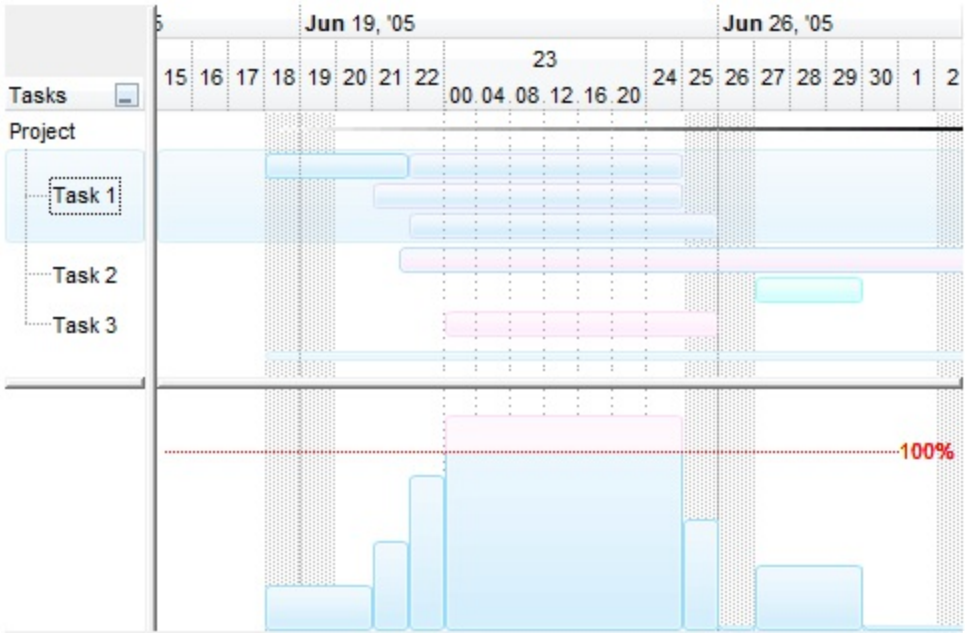


The following screen shot shows the **exHistOverAllocation** histogram and how it is updated as soon the bars are moved or resized:

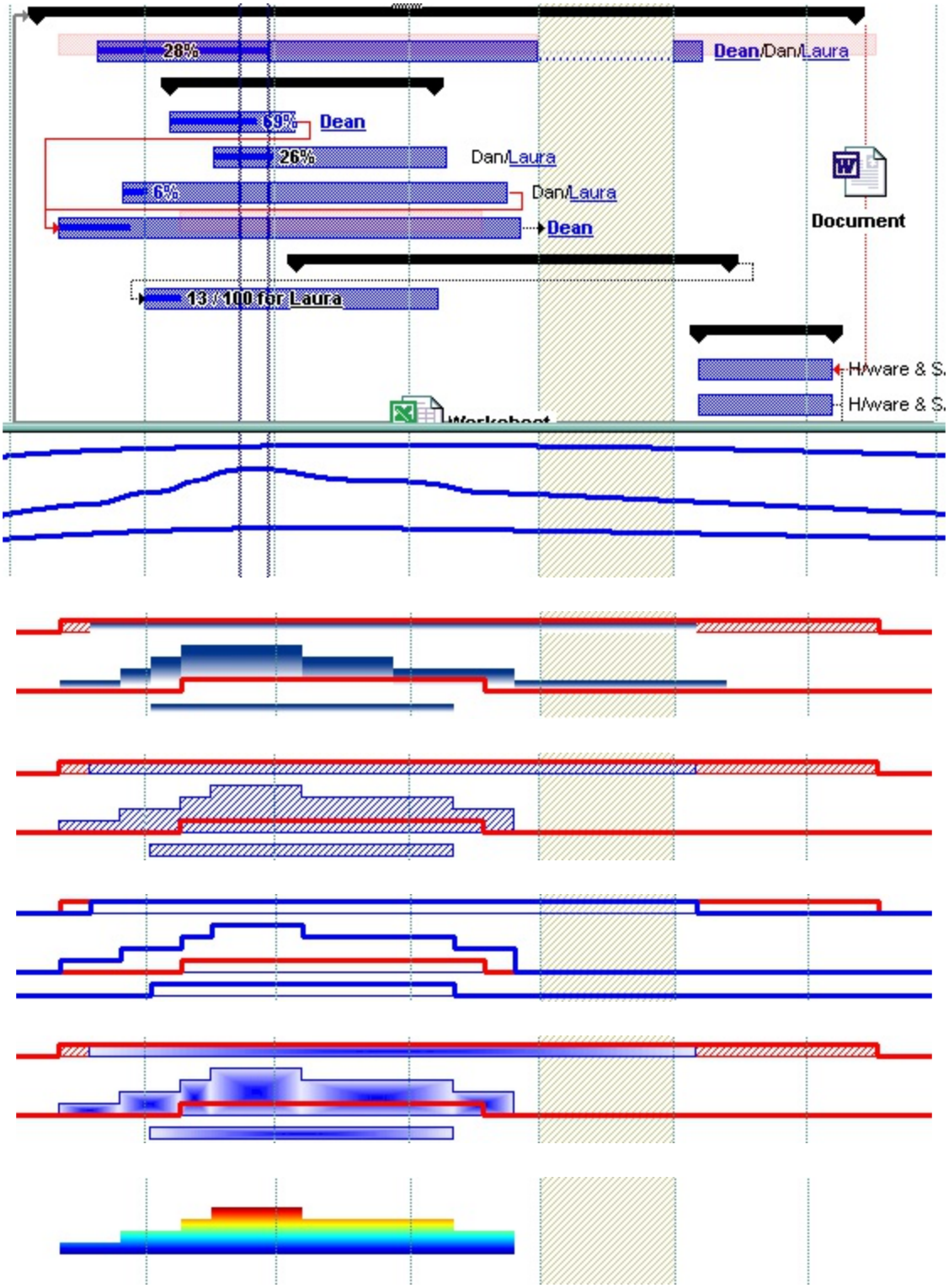








Bellow you can view few screen shots of histograms being displayed in different ways, using different color, patterns, curves or EBN skin files:



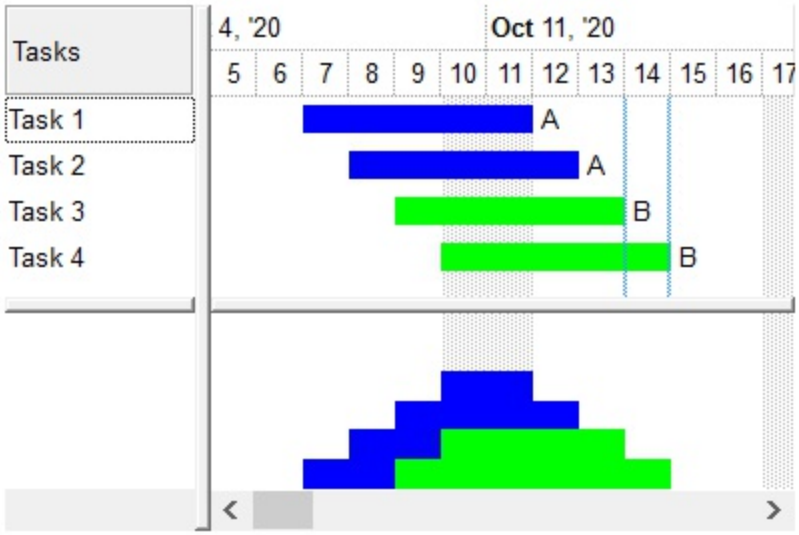
Use the Background(exSplitBar) property to define the shape and the color for horizontal split bar, that may be used to resize the histogram at runtime.

property Chart.HistogramZOrder as String

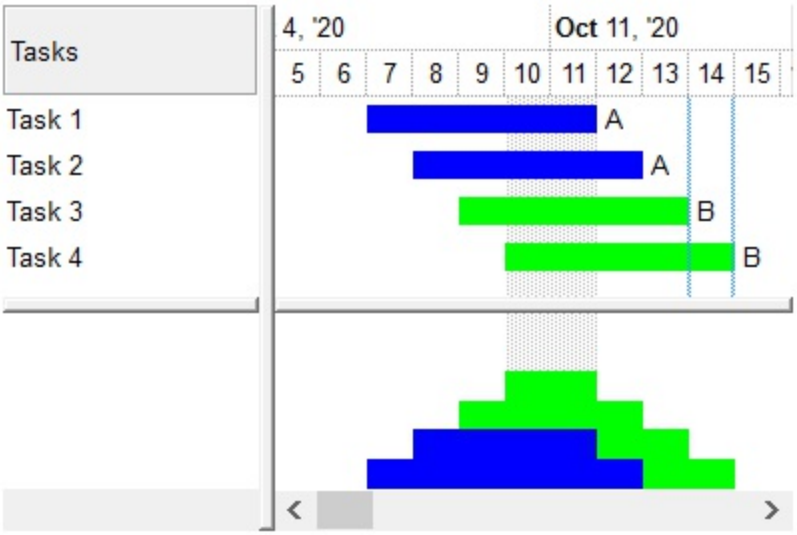
Specifies the z-order of the bars to be shown within the chart's histogram.

Type	Description
String	A string expression that defines the z-order of the bars to be shown within the chart's histogram. The list can include one or more names separated by the comma character

By default, the HistogramZOrder property is empty, which indicates that it has no effect. The HistogramZOrder property defines the z-order of the bars to be shown within the chart's histogram. The Name property defines the bar's name. For instance, let's say we have defined the new type of bars A, and B so the HistogramZOrder: "A,B" shows as:



while HistogramZOrder on "B,A" shows as:



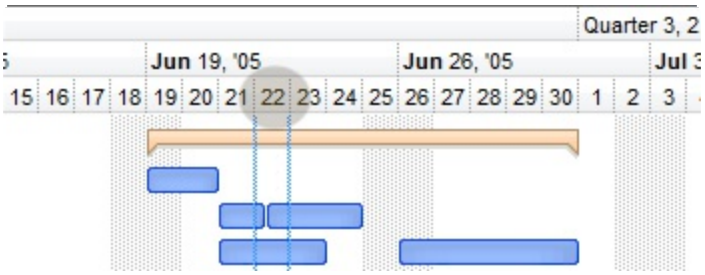
property Chart.InsideZoomOnDblClick as Boolean

Gets or sets a value that indicates whether a portion of the chart is magnified or zoomed when the user double click a date.

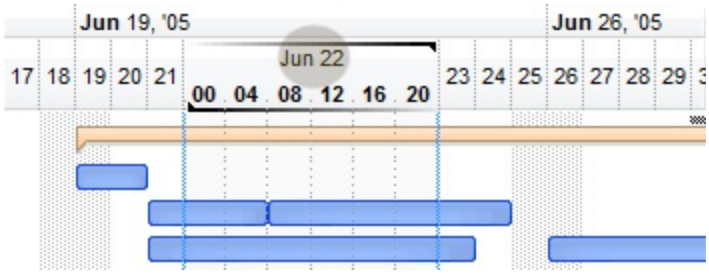
Type	Description
Boolean	A boolean expression that specifies whether the user can magnify a time unit, by double clicking it in the chart's base level.

By default, the InsideZoomOnDblClick property is True. The inside zoom units are shown ONLY if the AllowInsideZoom property is True. The inside zoom feature allows displaying portions of the chart with different time scale units. For instance, you can display the bars on hours, while the chart still displays days. Use the CondInsideZoom property to specify the dates that can be magnified by double clicking the chart's base level. Use the DefaultInsideZoomFormat property to specify the format of the dates being magnified. The AllowResizeInsideZoom property specifies whether the user can magnify dates by resizing them in the chart's base level. The InsideZooms property retrieves the collection of inside zoom units.

The following chart displays days:



The Jun 22, gets magnified (once it is double clicked) to hours so it looks like follows (the first line displays the day, while the second displays the hours, the rest of the chart displays days):



property Chart.InsideZooms as InsideZooms

Retrieves the collection of inside zoom dates.

Type	Description
InsideZooms	An InsideZooms object that holds a collection of the InsideZoom objects.

The InsideZooms property retrieves the control's InsideZooms collection. Use the Add method to add programmatically new inside zoom units. *The Add method returns nothing, if the AllowInsideZoom property is False (by default)*. The control fires the InsideZoom event once a new inside zoom unit is added. The DefaultInsideZoomFormat retrieves an InsideZoomFormat object that customizes the dates being magnified.

property Chart.IsDateVisible (Date as Variant) as Boolean

Specifies whether the date fits the control's chart area.

Type	Description
Date as Variant	A Date expression being queried
Boolean	A Boolean expression that indicates whether the date fits the chart's area.

The IsDateVisible property specifies whether a date is visible or hidden. Use the FirstVisibleDate property to specify the first visible date in the chart's area. The DateChange event notifies your application whether the chart changes it's first visible date, or whether the user browses a new area in the chart.

The following VB sample enumerates all visible dates:

```
With G2antt1
  .BeginUpdate
  With .Chart
    Dim d As Date
    d = .FirstVisibleDate
    Do While .IsDateVisible(d)
      If Day(d) = 11 Then
        If Not (.IsNonworkingDate(d)) Then
          .AddNonworkingDate d
        End If
      End If
      d = .NextDate(d, exDay, 1)
    Loop
  End With
  .EndUpdate
End With
```

The following VB.NET sample enumerates all visible dates:

```
With AxG2antt1
  .BeginUpdate()
  With .Chart
    Dim d As Date
    d = .FirstVisibleDate
```

```
    Do While .IsDateVisible(d)
      If d.Day = 11 Then
        If Not (.IsNonworkingDate(d)) Then
          .AddNonworkingDate(d)
        End If
      End If
      d = .NextDate(d, EXG2ANTTLib.UnitEnum.exDay, 1)
    Loop
  End With
  .EndUpdate()
End With
```

The following C# sample enumerates all visible dates:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Chart chart = axG2antt1.Chart;
DateTime d = Convert.ToDateTime(chart.FirstVisibleDate);
while ( chart.get_IsDateVisible(d) )
{
  if ( d.Day == 11 )
    if ( !chart.get_IsNonworkingDate( d ) )
      chart.AddNonworkingDate(d);
  d = chart.get_NextDate(d, EXG2ANTTLib.UnitEnum.exDay, 1);
}
axG2antt1.EndUpdate();
```

The following VFP sample enumerates all visible dates:

```
With thisform.G2antt1
  .BeginUpdate
  With .Chart
    local d
    d = .FirstVisibleDate
    Do While .IsDateVisible(d)
      If Day(d) = 11 Then
        If Not (.IsNonworkingDate(d)) Then
          .AddNonworkingDate(d)
        EndIf
```

```
        EndIf
        d = .NextDate(d, 4096, 1)
      enddo
   EndWith
   .EndUpdate
EndWith
```

property Chart.IsNonworkingDate (Date as Variant, [Item as Variant]) as Boolean

Specifies whether the date is a nonworking day.

Type	Description
Date as Variant	A Date expression that indicates the date being queried.
Item as Variant	A Long expression that indicates the handle of the item where the date-time is queried. For instance, the ItemNonWorkingUnits may specify a different non-working part for a particular item. If Item parameter is missing or 0, the default non-working part is used to determine whether specified date-time is a non-working or working unit. If the Item parameter is valid and the item displays a different non-working part using the ItemNonWorkingUnits property, the IsNonworkingDate property queries the Date using the item's non-working expression to determine whether it is a non-working unit.
Boolean	A boolean expression that specifies whether the date is nonworking day.

The IsNonworkingDate property specifies whether the giving date-time is a non-working or working unit in the chart. Use the ShowNonworkingUnits property to display or hide the non-working units as hours or days in your chart. Use the UnitWidthNonworking property to specify a different width for non-working units in the base level. Use the ShowNonworkingDates property to specify whether the the days are shown or hidden while the ShowNonworkingUnits property is False.

You can use the following functions to specify non-working parts in the chart:

- The NonworkingDays property specifies the days being marked as nonworking in a week. Use the AddNonworkingDate method to add custom dates as being nonworking days. Use the NonworkingDaysPattern property to specify the pattern being used to fill non-working days. The NonworkingDaysColor property specifies the color being used to fill the non-working days. Use the ClearNonworkingDates method to remove all nonworking dates. Use the IsDateVisible property to specify whether a date fits the chart's area.
- The NonworkingHours property indicates the non-working hours within a day. The non-working hours are shown using the NonworkingHoursPattern and the NonworkingHoursColor which defines the pattern and the color, when the base level of the chart displays hours, if the ShowNonworkingUnits property is True (by default).
- The ItemNonworkingUnits property specifies different non-working zones for different items. If the Item parameter indicates a valid handle, the IsNonworkingDate property

queries the non-working expression for the item if the giving Date parameter is being non-working or working unit.

You can use the following attributes for a bar (ItemBar property) to work with non-working part of the bars:

- exBarWorkingCount attribute specifies the working count in days for the giving bar. For instance, if the exBarWorkingCount is 1 indicates a full day, or 24 working hours, while if it is 1/24 it indicates one working hour.
- exBarNonWorkingCount attributes specifies the working count in days for the giving bar. For instance, if the exBarNonWorkingCount is 1 indicates a full day, or 24 non-working hours, while if it is 1/24 it indicates one non-working hour.
- exBarWorkingUnits attribute retrieves a safe array of pair (start-end) that indicates the working parts of the bar. You can use the exBarWorkingUnitsAsString attribute to display the working parts of the bar as a string. The /NET assembly provides the *public virtual DateTime[] get_BarWorkingUnits(int Item, object Key)* method that returns the array of 2 DateTime objects that specifies the working parts of the bar.
- exBarWorkingUnitsAsString attribute retrieves the working part of the bar as a string, in other words it is similar with the exBarWorkingUnits excepts that it returns a string that shows the working parts of the bar.
- exBarNonWorkingUnits attribute retrieves a safe array of pair (start-end) that indicates the non-working parts of the bar. You can use the exBarNonWorkingUnitsAsString attribute to display the non-working parts of the bar as a string. The /NET assembly provides the *public virtual DateTime[] get_BarNonWorkingUnits(int Item, object Key)* method that returns the array of 2 DateTime objects that specifies the working parts of the bar.
- exBarNonWorkingUnitsAsString attribute retrieves the non-working part of the bar as a string, in other words it is similar with the exBarNonWorkingUnits excepts that it returns a string that shows the non-working parts of the bar.

The following VB sample displays True if the cursor hovers a nonworking part, and False if the cursor hovers a working part in the chart area of the control:

```
Private Sub G2antt1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
  With G2antt1
    Dim c As Long, hit As HitTestInfoEnum
    Debug.Print .Chart.IsNonworkingDate(.Chart.DateFromPoint(-1, -1), .ItemFromPoint(-1, -1, c, hit))
  End With
End Sub
```

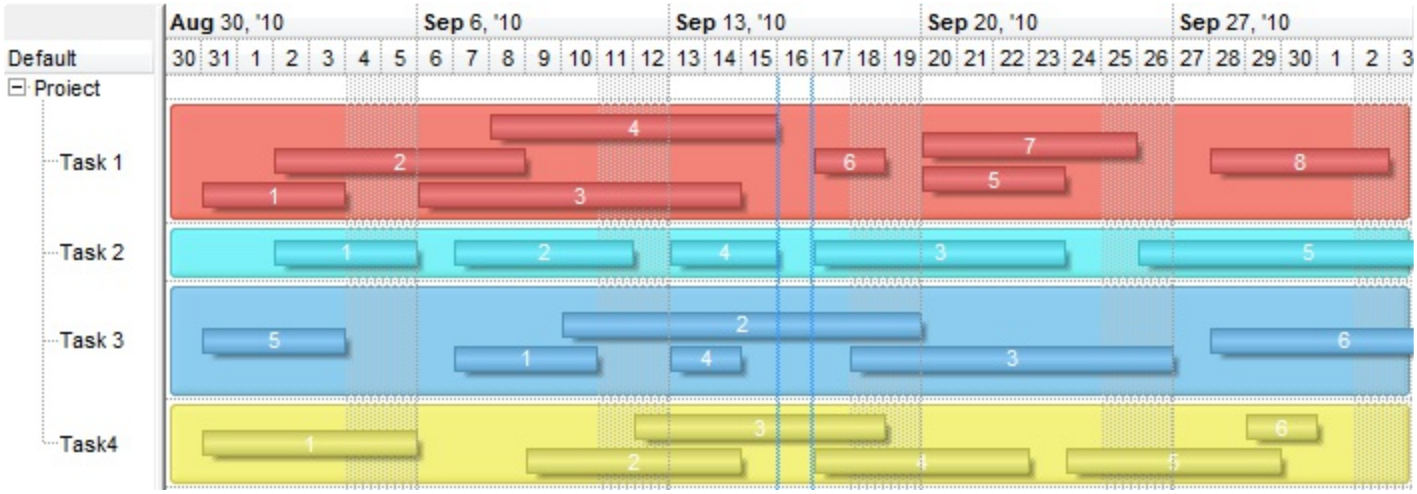
property Chart.ItemBackColor(Item as HITEM) as Color

Retrieves or sets a background color for a specific item, in the chart area.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item.
Color	A color expression that indicates the item's background color. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the

By default, the ItemBackColor property is the same as Chart's BackColor property. The ItemBackColor property specifies the background or the visual appearance for the item's background on the chart area. The ItemBackColor property specifies the item's background color for the list area (columns part of the control). The ClearItemBackColor method clears the item's background on the chart part of the control.

The following screen shot shows the chart part when using the ItemBackColor property of the Chart object:



The following samples changes the background color for the item in the chart part only.

VBA (MS Access, Excell...)

```
With G2antt1
   .Columns.Add "Default"
   With .Items
      h = .AddItem("Root")
      hC = .InsertItem(h,0,"Child 1")
```

```
      G2antt1.Chart.ItemBackColor(hC) = RGB(255,0,0)
      .InsertItem h,0,"Child 2"
      .ExpandItem(h) = True
   End With
End With
```

VB6

```
With G2antt1
   .Columns.Add "Default"
   With .Items
      h = .AddItem("Root")
      hC = .InsertItem(h,0,"Child 1")
      G2antt1.Chart.ItemBackColor(hC) = RGB(255,0,0)
      .InsertItem h,0,"Child 2"
      .ExpandItem(h) = True
   End With
End With
```

VB.NET

```
Dim h,hC
With Exg2antt1
   .Columns.Add("Default")
   With .Items
      h = .AddItem("Root")
      hC = .InsertItem(h,0,"Child 1")
      Exg2antt1.Chart.set_ItemBackColor(hC,Color.FromArgb(255,0,0))
      .InsertItem(h,0,"Child 2")
      .set_ExpandItem(h,True)
   End With
End With
```

VB.NET for /COM

```
Dim h,hC
With AxG2antt1
   .Columns.Add("Default")
   With .Items
```

```
    h = .AddItem("Root")
    hC = .InsertItem(h,0,"Child 1")
    AxG2antt1.Chart.ItemBackColor(hC) = RGB(255,0,0)
    .InsertItem(h,0,"Child 2")
    .ExpandItem(h) = True
  End With
End With
```

C++

```
/*
  Copy and paste the following directives to your header file as
  it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

  #import <ExG2antt.dll>
  using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->GetColumns()->Add(L"Default");
EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems();
  long h = var_Items->AddItem("Root");
  long hC = var_Items->InsertItem(h,long(0),"Child 1");
  spG2antt1->GetChart()->PutItemBackColor(hC,RGB(255,0,0));
  var_Items->InsertItem(h,long(0),"Child 2");
  var_Items->PutExpandItem(h,VARIANT_TRUE);
```

C#

```
exg2antt1.Columns.Add("Default");
exontrol.EXG2ANTTLib.Items var_Items = exg2antt1.Items;
  int h = var_Items.AddItem("Root");
  int hC = var_Items.InsertItem(h,0,"Child 1");
  exg2antt1.Chart.set_ItemBackColor(hC,Color.FromArgb(255,0,0));
  var_Items.InsertItem(h,0,"Child 2");
  var_Items.set_ExpandItem(h,true);
```

C# for /COM

```
axG2antt1.Columns.Add("Default");
EXG2ANTTLib.Items var_Items = axG2antt1.Items;
   int h = var_Items.AddItem("Root");
   int hC = var_Items.InsertItem(h,0,"Child 1");
   axG2antt1.Chart.set_ItemBackColor(hC,
(uint)ColorTranslator.ToWin32(Color.FromArgb(255,0,0)));
   var_Items.InsertItem(h,0,"Child 2");
   var_Items.set_ExpandItem(h,true);
```

Delphi 8 (.NET only)

```
with AxG2antt1 do
begin
   Columns.Add('Default');
   with Items do
   begin
      h := AddItem('Root');
      hC := InsertItem(h,TObject(0),'Child 1');
      AxG2antt1.Chart.ItemBackColor[hC] := $ff;
      InsertItem(h,TObject(0),'Child 2');
      ExpandItem[h] := True;
   end;
end
```

Delphi (standard)

```
with G2antt1 do
begin
   Columns.Add('Default');
   with Items do
   begin
      h := AddItem('Root');
      hC := InsertItem(h,OleVariant(0),'Child 1');
      G2antt1.Chart.ItemBackColor[hC] := $ff;
      InsertItem(h,OleVariant(0),'Child 2');
      ExpandItem[h] := True;
   end;
end
```

VFP

```
with thisform.G2antt1
   .Columns.Add("Default")
   with .Items
      h = .AddItem("Root")
      hC = .InsertItem(h,0,"Child 1")
      thisform.G2antt1.Chart.ItemBackColor(hC) = RGB(255,0,0)
      .InsertItem(h,0,"Child 2")
      .ExpandItem(h) = .T.
   endwith
endwith
```

property Chart.Label(Unit as UnitEnum) as String

Retrieves or sets a value that indicates the predefined format of the level's label for a specified unit.

Type	Description
Unit as UnitEnum	An UnitEnum expression that indicates the time unit
String	A String expression that includes the format of the label.

The Label property specifies a predefined label for a specified unit. Use the UnitScale property to change the scale unit. The UnitScale property changes the Label, Unit and the ToolTip for a level with predefined values defined by the Label and LabelToolTip properties. Use the UnitWidth property to specify the width of the time unit. Use the Zoom method to zoom the chart to a specified interval of dates. Use the Label property to assign a different label for a specified level. Use the LabelToolTip property to specify the predefined type of tooltip being displayed when the chart is zoomed. Use the ToolTip property to specify the tooltip that shows up when the cursor hovers the level. Use the FormatDate property to format a date. Use the MonthNames property to specify the name of the months in the year. The WeekDays property retrieves or sets a value that indicates the list of names for each week day, separated by space. If the Label property is empty, the unit is not displayed in the zooming scale, if the AllowOverviewZoom property is not exDisableZoom.

The Label property supports alternative HTML labels being separated by "<|>" and values for Count and Unit being separated by "<||>". By alternate HTML label we mean that you can define a list of HTML labels that may be displayed in the chart's header based on the space allocated for the time-unit. In other words, the control chooses automatically the alternate HTML label to be displayed for best fitting in the portion of the chart where the time-unit should be shown.

The Label property format is "**ALT1[<|>ALT2<|>...[<||>COUNT[<||>UNIT]]]**" where

- ALT defines a HTML label
- COUNT specifies the value for the Count property
- UNIT field indicates the value for the Unit property
- and the parts delimited by [] brackets may miss.

The Label property may change the Unit and the Count property. You can always use a different Unit or Count by setting the property after setting the Label property.

The Label property supports the following built-in tags:

- <%**d**%> - Day of the month in one or two numeric digits, as needed (1 to 31).
- <%**dd**%> - Day of the month in two numeric digits (01 to 31).
- <%**d1**%> - First letter of the weekday (S to S). (Use the WeekDays property to

specify the name of the days in the week)

- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**d2**%> - First two letters of the weekday (Su to Sa). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**d3**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**ddd**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays property to specify the name of the days in the week). You can use the <%**loc_ddd**%> that indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%**loc_ddd**%> - Indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%**dddd**%> - Full name of the weekday (Sunday to Saturday). (Use the WeekDays property to specify the name of the days in the week). You can use the <%**loc_dddd**%> that indicates day of week as its full name using the current user regional and language settings.
- <%**loc_dddd**%> - Indicates day of week as its full name using the current user regional and language settings.
- <%**i**%> - Displays the number instead the date. For instance, you can display numbers as 1000, 1001, 1002, 1003, instead dates. (the valid range is from -647,434 to 2,958,465)
- <%**w**%> - Day of the week (1 to 7).
- <%**ww**%> - Week of the year (1 to 53).
- <%**m**%> - Month of the year in one or two numeric digits, as needed (1 to 12).
- <%**mr**%> - Month of the year in Roman numerals, as needed (I to XII).
- <%**mm**%> - Month of the year in two numeric digits (01 to 12).
- <%**m1**%> - First letter of the month (J to D). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**m2**%> - First two letters of the month (Ja to De). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**m3**%> - First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**mmm**%> - First three letters of the month (Jan to Dec). (Use the MonthNames

property to specify the name of the months in the year). You can use the <%**loc_mmm**%> that indicates month as a three-letter abbreviation using the current user regional and language settings.

- <%**loc_mmm**%> - Indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%**mmmm**%> - Full name of the month (January to December). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmmm**%> that indicates month as its full name using the current user regional and language settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user regional and language settings.
- <%**q**%> - Date displayed as the quarter of the year (1 to 4).
- <%**y**%> - Number of the day of the year (1 to 366).
- <%**yy**%> - Last two digits of the year (01 to 99).
- <%**yyyy**%> - Full year (0100 to 9999).
- <%**hy**%> - Date displayed as the half of the year (1 to 2).
- <%**loc_gg**%> - Indicates period/era using the current user regional and language settings.
- <%**loc_sdate**%> - Indicates the date in the short format using the current user regional and language settings.
- <%**loc_ldate**%> - Indicates the date in the long format using the current user regional and language settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user regional and language settings (/).
- <%**h**%> - Hour in one or two digits, as needed (0 to 23).
- <%**hh**%> - Hour in two digits (00 to 23).
- <%**h12**%> - Hour in 12-hour time format, in one or two digits - [0(12),11]
- <%**hh12**%> - hour in 12-hour time format, in two digits - [00(12),11]
- <%**n**%> - Minute in one or two digits, as needed (0 to 59).
- <%**nn**%> - Minute in two digits (00 to 59).
- <%**s**%> - Second in one or two digits, as needed (0 to 59).
- <%**ss**%> - Second in two digits (00 to 59).
- <%**AM/PM**%> - Twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate. (Use the AMPM property to specify the name of the AM and PM indicators). You can use the <%**loc_AM/PM**%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%**loc_A/P**%> that indicates the one character time marker such as A or P using the current user regional and language settings
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user regional and language settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user regional and language settings.

- <%**loc_time**%> - Indicates the time using the current user regional and language settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%**loc_tsep**%> - indicates the time separator using the current user regional and language settings (:)
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

The following tags are displayed based on the user's Regional and Language Options:

- <%**loc_sdate**%> - Indicates the date in the short format using the current user settings.
- <%**loc_ldate**%> - Indicates the date in the long format using the current user settings.
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**loc_ddd**%> - Indicates day of week as a three-letters abbreviation using the current user settings.
- <%**loc_dddd**%> - Indicates day of week as its full name using the current user settings.
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**loc_mmm**%> - Indicates month as a three-letters abbreviation using the current user settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user settings.
- <%**loc_gg**%> - Indicates period/era using the current user settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user settings.

- <%**loc_time**%> - Indicates the time using the current user settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user settings.
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user settings.
- <%**loc_tsep**%> - Indicates the time separator using the current user settings.
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

The Label property supports the following built-in HTML tags:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.

  The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using <a ;exp=> or <a ;e64=> anchor tags. The exp/e64 field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption.

  - exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
  - e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu

" that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being

inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.

- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with $^{\text{subscript}}$
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

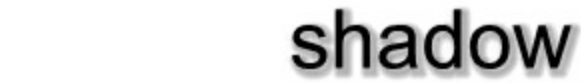
gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the

following picture:



or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor></**sha**>*" gets:

The Label property may be a combination of any of these tags. For instance, the "<%mmm%> <%d%>, '<%yy%>" displays a date like: "**May** 29,'05".

By default, the Label property is:

- exYear: "<%yy%><|>'<%yy%><|><%yyyy%>"
- exHalfYear: ""
- exQuarterYear: ""
- exMonth: "<|><%m1%><|><%m2%><|><%m3%><|><%mmmm%><|><%m3%> '<%yy%><|><%mmmm%> <%yyyy%>"
- exThirdMonth: ""
- exWeek: "<|><%ww%><|><%m3%> <%d%>, '<%yy%><r><%ww%><|> <%mmmm%> <%d%>, <%yyyy%><r><%ww%><||><||>256"
- exDay: "<|><%d1%><|><%d2%><|><%d3%><|><%dddd%><|><%d3%>, <%m3%> <%d%>, '<%yy%><|><%dddd%>, <%mmmm%> <%d%>, <%yyyy%><||><||>4096"
- exHour: "<|><%hh%><|><%h%> <%AM/PM%><|><%d3%>, <%m3%> <%d%>, '<%yy%> <%h%> <%AM/PM%><|><%dddd%>, <%mmmm%> <%d%>, <%yyyy%> <%h%> <%AM/PM%><||><||>65536"
- exMinute: "<|><%nn%><|><%h%>:<%nn%> <%AM/PM%><|><%d3%>, <%m3%> <%d%>, '<%yy%> <%h%>:<%nn%> <%AM/PM%><|><%dddd%>, <%mmmm%> <%d%>, <%yyyy%> <%h%>:<%nn%> <%AM/PM%>"
- exSecond: "<|><%ss%><|><%nn%>:<%ss%><|><%h%>:<%nn%>:<%ss%> <%AM/PM%><|><%d3%>, <%m3%> <%d%>, '<%yy%> <%h%>:<%nn%>:<%ss%> <%AM/PM%><|><%dddd%>, <%mmmm%> <%d%>, <%yyyy%> <%h%>:<%nn%>:<%ss%> <%AM/PM%>"

For instance the Label(exWeek) is "<|><%ww%><|><%m3%> <%d%>, '<%yy%><r><%ww%><|><%mmmm%> <%d%>, <%yyyy%><r><%ww%><||><||>256" which means that if a level's unit is set on exWeek it may display one of the following alternate labels:

- nothing, if the space is less than 6 pixels
- <%ww%> - week number
- <%m3%> <%d%>, '<%yy%><r><%ww%> - month, day, year in short format where the week begins, including the week number on the right

- <%mmmm%> <%d%>, <%yyyy%><r><%ww%> - month, day, year in long format where the week begins, including the week number on the right

So actually, the control will choose any of these formats based on the UnitWidth, Font and the layout of the levels.

property Chart.LabelToolTip(Unit as UnitEnum) as String

Retrieves or sets a value that indicates the predefined format of the level's tooltip for a specified unit.

Type	Description
Unit as UnitEnum	An UnitEnum expression that indicates the time unit
String	A String expression that includes the format of the tooltip.

The LabelToolTip property specifies a predefined tooltip for a specified unit. Use the ToolTip property to specify the tooltip that shows up when the cursor hovers the level. The ToolTip property retrieves or sets a value that indicates the format of the tooltip being shown while the user scrolls the chart. Use the FormatDate property to format a date. Use the MonthNames property to specify the name of the months in the year. The WeekDays property retrieves or sets a value that indicates the list of names for each week day, separated by space. Use the Zoom method to zoom the chart to a specified interval of dates. Use the AMPM property to specify the name of the AM and PM indicators. The Label property specifies a predefined label for a specified unit.

The LabelToolTip property supports the following built-in tags:

- <%**d**%> - Day of the month in one or two numeric digits, as needed (1 to 31).
- <%**dd**%> - Day of the month in two numeric digits (01 to 31).
- <%**d1**%> - First letter of the weekday (S to S). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**d2**%> - First two letters of the weekday (Su to Sa). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**d3**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**ddd**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays property to specify the name of the days in the week). You can use the <%**loc_ddd**%> that indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%**loc_ddd**%> - Indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%**dddd**%> - Full name of the weekday (Sunday to Saturday). (Use the WeekDays property to specify the name of the days in the week). You can use the <%**loc_dddd**%> that indicates day of week as its full name using the current user

regional and language settings.

- <%**loc_dddd**%> - Indicates day of week as its full name using the current user regional and language settings.
- <%**i**%> - Displays the number instead the date. For instance, you can display numbers as 1000, 1001, 1002, 1003, instead dates. (the valid range is from -647,434 to 2,958,465)
- <%**w**%> - Day of the week (1 to 7).
- <%**ww**%> - Week of the year (1 to 53).
- <%**m**%> - Month of the year in one or two numeric digits, as needed (1 to 12).
- <%**mr**%> - Month of the year in Roman numerals, as needed (I to XII).
- <%**mm**%> - Month of the year in two numeric digits (01 to 12).
- <%**m1**%> - First letter of the month (J to D). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**m2**%> - First two letters of the month (Ja to De). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**m3**%> - First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**mmm**%> - First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmm**%> that indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%**loc_mmm**%> - Indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%**mmmm**%> - Full name of the month (January to December). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmmm**%> that indicates month as its full name using the current user regional and language settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user regional and language settings.
- <%**q**%> - Date displayed as the quarter of the year (1 to 4).
- <%**y**%> - Number of the day of the year (1 to 366).
- <%**yy**%> - Last two digits of the year (01 to 99).
- <%**yyyy**%> - Full year (0100 to 9999).
- <%**hy**%> - Date displayed as the half of the year (1 to 2).
- <%**loc_gg**%> - Indicates period/era using the current user regional and language settings.
- <%**loc_sdate**%> - Indicates the date in the short format using the current user regional

and language settings.

- <%**loc_ldate**%> - Indicates the date in the long format using the current user regional and language settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user regional and language settings (/).
- <%**h**%> - Hour in one or two digits, as needed (0 to 23).
- <%**hh**%> - Hour in two digits (00 to 23).
- <%**h12**%> - Hour in 12-hour time format, in one or two digits - [0(12),11]
- <%**hh12**%> - hour in 12-hour time format, in two digits - [00(12),11]
- <%**n**%> - Minute in one or two digits, as needed (0 to 59).
- <%**nn**%> - Minute in two digits (00 to 59).
- <%**s**%> - Second in one or two digits, as needed (0 to 59).
- <%**ss**%> - Second in two digits (00 to 59).
- <%**AM/PM**%> - Twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate. (Use the AMPM property to specify the name of the AM and PM indicators). You can use the <%**loc_AM/PM**%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%**loc_A/P**%> that indicates the one character time marker such as A or P using the current user regional and language settings
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user regional and language settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user regional and language settings.
- <%**loc_time**%> - Indicates the time using the current user regional and language settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%**loc_tsep**%> - indicates the time separator using the current user regional and language settings (:)
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

The following tags are displayed based on the user's Regional and Language Options:

- <%**loc_sdate**%> - Indicates the date in the short format using the current user settings.
- <%**loc_ldate**%> - Indicates the date in the long format using the current user settings.
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**loc_ddd**%> - Indicates day of week as a three-letters abbreviation using the current user settings.
- <%**loc_dddd**%> - Indicates day of week as its full name using the current user settings.
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**loc_mmm**%> - Indicates month as a three-letters abbreviation using the current user settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user settings.
- <%**loc_gg**%> - Indicates period/era using the current user settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user settings.
- <%**loc_time**%> - Indicates the time using the current user settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user settings.
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user settings.
- <%**loc_tsep**%> - Indicates the time separator using the current user settings
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

The LabelToolTip property supports the following built-in HTML tags:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.

  The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using <a ;exp=> or <a ;e64=> anchor tags. The exp/e64 field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption.

  - exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
  - e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

  Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.

- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{subscript}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or

<fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

By default, the LabelToolTip property is:

- exYear: "<%yyyy%>"
- exHalfYear: ""
- exQuarterYear: ""
- exMonth: "<%mmmm%>/ <%yyyy%>"
- exThirdMonth: ""
- exWeek: "<%mmmm%> <%d%>, <%yyyy%> <%ww%>"
- exDay: "<%dddd%>, <%mmmm%> <%d%>, <%yyyy%>"
- exHour: "<%dddd%>, <%mmmm%> <%d%>, <%yyyy%> <%h%> <%AM/PM%>"

- exMinute: "<%dddd%>, <%mmmm%> <%d%>, <%yyyy%> <%h%>:<%nn%> <%AM/PM%>"
- exSecond: "<%dddd%>, <%mmmm%> <%d%>, <%yyyy%> <%h%>:<%nn%>:<%ss%> <%AM/PM%>"

property Chart.Level (Index as Long) as Level

Retrieves the level based on its index.

Type	Description
Index as Long	A long expression that indicates the index of the level being accessed.
Level	A Level object being accessed.

The Level property retrieves the Level based on its index. Use the LevelCount property to specify the number of levels being displayed in the chart's header. Use the HeaderVisible property to hide the control's header bar. The control's header bar displays the levels in the chart area too. If the control displays the header bar using multiple levels the HeaderHeight property gets the height in pixels of a single level in the header bar. Use the LevelFromPoint property to get the index of the level from the cursor.

The following VB sample enumerates the levels in the chart:

```
With G2antt1.Chart
   Dim i As Long
   For i = 0 To .LevelCount - 1
      With .Level(i)
         Debug.Print .Label
      End With
   Next
End With
```

The following C++ sample enumerates the levels in the chart:

```
CChart chart = m_g2antt.GetChart();
for ( long i = 0; i < chart.GetLevelCount(); i++ )
{
   CLevel level = chart.GetLevel( i );
   OutputDebugString( V2S( &level.GetLabel() ) );
}
```

where the V2S function converts a Variant expression to a string expression:

```
static CString V2S( VARIANT* pvtDate )
{
   COleVariant vtDate;
```

```
    vtDate.ChangeType( VT_BSTR, pvtDate );
    return V_BSTR( &vtDate );
}
```

The following VB.NET sample enumerates the levels in the chart:

```
With AxG2antt1.Chart
    Dim i As Long
    For i = 0 To .LevelCount - 1
        With .Level(i)
            Debug.Write(.Label())
        End With
    Next
End With
```

The following C# sample enumerates the levels in the chart:

```
for (int i = 0; i < axG2antt1.Chart.LevelCount; i++)
{
    EXG2ANTTLib.Level level = axG2antt1.Chart.get_Level(i);
    System.Diagnostics.Debug.Write(level.Label);
}
```

The following VFP sample enumerates the levels in the chart:

```
With thisform.G2antt1.Chart
    For i = 0 To .LevelCount - 1
        With .Level(i)
            wait window nowait .Label
        EndWith
    Next
EndWith
```

property Chart.LevelCount as Long

Specifies the number of levels in the control's header.

Type	Description
Long	A Long expression that indicates the number of levels being displayed in the control's header.

By default, the control displays a single level. Use the LevelCount property to specify the number of levels being displayed in the chart's header. Use the Level property to access the level in the chart area. Use the Label property to specify the level's HTML label. Use the Unit property to specify the time-scale unit for the chart's level. Use the HeaderVisible property to show/extent/hide the control's header bar. The control's header bar displays the levels in the chart area too. Use the Caption property to specify the column's caption being displayed in the control's header bar. Use the BackColorLevelHeader property to specify the background color of the chart's header. Use the ForeColorLevelHeader property to specify the foreground color of the chart's header. If the control displays the header bar using multiple levels the HeaderHeight property gets the height in pixels of a single level in the header bar. Use the LevelKey property to specify the key of the column.

Newer versions support **Regional and Language Options** for tags such as:

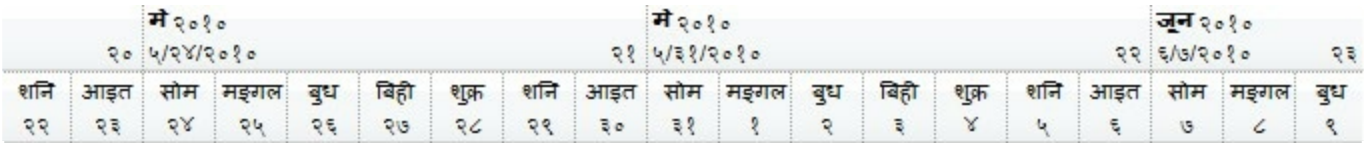
- <%**loc_sdate**%> - Indicates the date in the short format using the current user settings.
- <%**loc_ldate**%> - Indicates the date in the long format using the current user settings.
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**loc_ddd**%> - Indicates day of week as a three-letters abbreviation using the current user settings.
- <%**loc_dddd**%> - Indicates day of week as its full name using the current user settings.
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**loc_mmm**%> - Indicates month as a three-letters abbreviation using the current user settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user settings.
- <%**loc_gg**%> - Indicates period/era using the current user settings.

- <%**loc_dsep**%> - Indicates the date separator using the current user settings.
- <%**loc_time**%> - Indicates the time using the current user settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user settings.
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user settings.
- <%**loc_tsep**%> - Indicates the time separator using the current user settings.

You can use these in methods as: Level.Label, Level.ToolTip, Chart.Label, Chart.LabelToolTip, Chart.FormatDate, Chart.OverviewToolTip, Chart.ToolTip, InsideZoomFormat.InsideLabel, InsideZoomFormat.OwnerLabel, Note.PartText and Note.Text (where supported).

The following screen shot shows the chart's header for **English** (United States) format:

The following screen shot shows the chart's header for **Nepali** (Nepal) format:



The following screen shot shows the chart's header for **German** (Germany) format:

The following **VBA** sample shows how you can specify the levels using the user's Regional and Language Options?

```
With G2antt1
   .BeginUpdate
   .Font.Name = "Arial Unicode MS"
   .HeaderHeight = 36
   With .Chart
      .FirstVisibleDate = #5/30/2010#
      .PaneWidth(False) = 0
```

```
    .FirstWeekDay = 1
    .UnitWidth = 36
    .LevelCount = 2
    With .Level(0)
      .Label = "<b><%loc_mmmm%></b> <%yyyy%><br><%loc_sdate%><r>
<%ww%> "
      .ToolTip = .Label
      .Unit = 256
    End With
    With .Level(1)
      .Label = "<%loc_ddd%><br><%d%>"
      .ToolTip = .Label
    End With
    .ToolTip = "<%loc_ldate%>"
  End With
  .EndUpdate
End With
```

The following **VB6** sample shows how you can specify the levels using the user's Regional and Language Options?

```
With G2antt1
  .BeginUpdate
  .Font.Name = "Arial Unicode MS"
  .HeaderHeight = 36
  With .Chart
    .FirstVisibleDate = #5/30/2010#
    .PaneWidth(False) = 0
    .FirstWeekDay = exMonday
    .UnitWidth = 36
    .LevelCount = 2
    With .Level(0)
      .Label = "<b><%loc_mmmm%></b> <%yyyy%><br><%loc_sdate%><r>
<%ww%> "
      .ToolTip = .Label
      .Unit = exWeek
    End With
```

```
      With .Level(1)
         .Label = "<%loc_ddd%><br><%d%>"
         .ToolTip = .Label
      End With
      .ToolTip = "<%loc_ldate%>"
   End With
   .EndUpdate
End With
```

The following **VB.NET** sample shows how you can specify the levels using the user's Regional and Language Options?

```
With Exg2antt1
   .BeginUpdate()
   .Font.Name = "Arial Unicode MS"
   .HeaderHeight = 36
   With .Chart
      .FirstVisibleDate = #5/30/2010#
      .set_PaneWidth(False,0)
      .FirstWeekDay = exontrol.EXG2ANTTLib.WeekDayEnum.exMonday
      .UnitWidth = 36
      .LevelCount = 2
      With .get_Level(0)
         .Label = "<b><%loc_mmmm%></b> <%yyyy%><br><%loc_sdate%><r>
<%ww%> "
         .ToolTip = .Label
         .Unit = exontrol.EXG2ANTTLib.UnitEnum.exWeek
      End With
      With .get_Level(1)
         .Label = "<%loc_ddd%><br><%d%>"
         .ToolTip = .Label
      End With
      .ToolTip = "<%loc_ldate%>"
   End With
   .EndUpdate()
End With
```

The following **VB.NET for /COM** sample shows how you can specify the levels using the

user's Regional and Language Options?

```
With AxG2antt1
   .BeginUpdate()
   .Font.Name = "Arial Unicode MS"
   .HeaderHeight = 36
   With .Chart
      .FirstVisibleDate = #5/30/2010#
      .PaneWidth(False) = 0
      .FirstWeekDay = EXG2ANTTLib.WeekDayEnum.exMonday
      .UnitWidth = 36
      .LevelCount = 2
      With .Level(0)
         .Label = "<b><%loc_mmmm%></b> <%yyyy%><br><%loc_sdate%><r>
<%ww%> "
         .ToolTip = .Label
         .Unit = EXG2ANTTLib.UnitEnum.exWeek
      End With
      With .Level(1)
         .Label = "<%loc_ddd%><br><%d%>"
         .ToolTip = .Label
      End With
      .ToolTip = "<%loc_ldate%>"
   End With
   .EndUpdate()
End With
```

The following **C++** sample shows how you can specify the levels using the user's Regional and Language Options?

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
```

```
>GetControlUnknown();
spG2antt1->BeginUpdate();
spG2antt1->GetFont()->PutName(L"Arial Unicode MS");
spG2antt1->PutHeaderHeight(36);
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutFirstVisibleDate("5/30/2010");
   var_Chart->PutPaneWidth(VARIANT_FALSE,0);
   var_Chart->PutFirstWeekDay(EXG2ANTTLib::exMonday);
   var_Chart->PutUnitWidth(36);
   var_Chart->PutLevelCount(2);
   EXG2ANTTLib::ILevelPtr var_Level = var_Chart->GetLevel(0);
      var_Level->PutLabel("<b><%loc_mmmm%></b> <%yyyy%><br><%loc_sdate%>
<r> <%ww%> ");
      var_Level->PutToolTip(var_Level->GetLabel());
      var_Level->PutUnit(EXG2ANTTLib::exWeek);
   EXG2ANTTLib::ILevelPtr var_Level1 = var_Chart->GetLevel(1);
      var_Level1->PutLabel("<%loc_ddd%><br><%d%>");
      var_Level1->PutToolTip(var_Level1->GetLabel());
   var_Chart->PutToolTip(L"<%loc_ldate%>");
spG2antt1->EndUpdate();
```

The following **C#** sample shows how you can specify the levels using the user's Regional and Language Options?

```
exg2antt1.BeginUpdate();
exg2antt1.Font.Name = "Arial Unicode MS";
exg2antt1.HeaderHeight = 36;
exontrol.EXG2ANTTLib.Chart var_Chart = exg2antt1.Chart;
   var_Chart.FirstVisibleDate = Convert.ToDateTime("5/30/2010");
   var_Chart.set_PaneWidth(false,0);
   var_Chart.FirstWeekDay = exontrol.EXG2ANTTLib.WeekDayEnum.exMonday;
   var_Chart.UnitWidth = 36;
   var_Chart.LevelCount = 2;
   exontrol.EXG2ANTTLib.Level var_Level = var_Chart.get_Level(0);
      var_Level.Label = "<b><%loc_mmmm%></b> <%yyyy%><br><%loc_sdate%><r>
<%ww%> ";
      var_Level.ToolTip = var_Level.Label;
```

```
    var_Level.Unit = exontrol.EXG2ANTTLib.UnitEnum.exWeek;
  exontrol.EXG2ANTTLib.Level var_Level1 = var_Chart.get_Level(1);
    var_Level1.Label = "<%loc_ddd%><br><%d%>";
    var_Level1.ToolTip = var_Level1.Label;
  var_Chart.ToolTip = "<%loc_ldate%>";
exg2antt1.EndUpdate();
```

The following **C# for /COM** sample shows how you can specify the levels using the user's Regional and Language Options?

```
axG2antt1.BeginUpdate();
axG2antt1.Font.Name = "Arial Unicode MS";
axG2antt1.HeaderHeight = 36;
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
  var_Chart.FirstVisibleDate = Convert.ToDateTime("5/30/2010");
  var_Chart.set_PaneWidth(false,0);
  var_Chart.FirstWeekDay = EXG2ANTTLib.WeekDayEnum.exMonday;
  var_Chart.UnitWidth = 36;
  var_Chart.LevelCount = 2;
  EXG2ANTTLib.Level var_Level = var_Chart.get_Level(0);
    var_Level.Label = "<b><%loc_mmmm%></b> <%yyyy%><br><%loc_sdate%><r>
<%ww%> ";
    var_Level.ToolTip = var_Level.Label;
    var_Level.Unit = EXG2ANTTLib.UnitEnum.exWeek;
  EXG2ANTTLib.Level var_Level1 = var_Chart.get_Level(1);
    var_Level1.Label = "<%loc_ddd%><br><%d%>";
    var_Level1.ToolTip = var_Level1.Label;
  var_Chart.ToolTip = "<%loc_ldate%>";
axG2antt1.EndUpdate();
```

The following **VFP** sample shows how you can specify the levels using the user's Regional and Language Options?

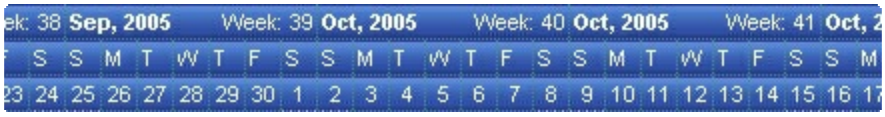
```
with thisform.G2antt1
  .BeginUpdate
  .Font.Name = "Arial Unicode MS"
  .HeaderHeight = 36
  with .Chart
```

```
    .FirstVisibleDate = {^2010-5-30}
    .PaneWidth(.F.) = 0
    .FirstWeekDay = 1
    .UnitWidth = 36
    .LevelCount = 2
    with .Level(0)
      .Label = "<b><%loc_mmmm%></b> <%yyyy%><br><%loc_sdate%><r>
<%ww%> "
      .ToolTip = .Label
      .Unit = 256
    endwith
    with .Level(1)
      .Label = "<%loc_ddd%><br><%d%>"
      .ToolTip = .Label
    endwith
    .ToolTip = "<%loc_ldate%>"
  endwith
  .EndUpdate
endwith
```

The following **Delphi** sample shows how you can specify the levels using the user's Regional and Language Options?

```
with AxG2antt1 do
begin
  BeginUpdate();
  Font.Name := 'Arial Unicode MS';
  HeaderHeight := 36;
  with Chart do
  begin
    FirstVisibleDate := '5/30/2010';
    PaneWidth[False] := 0;
    FirstWeekDay := EXG2ANTTLib.WeekDayEnum.exMonday;
    UnitWidth := 36;
    LevelCount := 2;
    with Level[0] do
    begin
```

```
        Label := '<b><%loc_mmmm%></b> <%yyyy%><br><%loc_sdate%><r>
<%ww%> ';
        ToolTip := Label;
        Unit := EXG2ANTTLib.UnitEnum.exWeek;
      end;
      with Level[1] do
      begin
        Label := '<%loc_ddd%><br><%d%>';
        ToolTip := Label;
      end;
      ToolTip := '<%loc_ldate%>';
   end;
   EndUpdate();
end
```



The first level displays the month, the year and the number of the week in the year , the second level displays the name of the week day, and the third level displays the day of the month. The LevelCount property specifies the number of levels being displayed, in our case 3.

The following Template shows how to display your header using three levels as arranged in the picture above (just copy and paste the following script to Template page):

```
BeginUpdate()
Chart
{
   LevelCount = 3
   Level(0)
   {
      Label = "<b><%mmm%>, <%yyyy%></b> <r>Week: <%ww%>"
      Unit = 256 'exWeek
   }
   Level(1).Label = "<%d1%>"
   Level(2).Label = "<%d%>"
}
EndUpdate()
```

The following VB sample displays your header using 3 levels as shown above:

```
With G2antt1
   .BeginUpdate
   With .Chart
      .LevelCount = 3
      With .Level(0)
         .Label = "<b><%mmm%>, <%yyyy%></b> <r>Week: <%ww%>"
         .Unit = EXG2ANTTLibCtl.UnitEnum.exWeek
      End With
      .Level(1).Label = "<%d1%>"
      .Level(2).Label = "<%d%>"
   End With
   .EndUpdate
End With
```

The following VFP sample displays your header using 3 levels:

```
with thisform.g2antt1
.BeginUpdate()
with .Chart
   .LevelCount = 3
   with .Level(0)
      .Label = "<b><%mmm%>, <%yyyy%></b> <r>Week: <%ww%>"
      .Unit = 256
   endwith
   .Level(1).Label = "<%d1%>"
   .Level(2).Label = "<%d%>"
endwith
.EndUpdate()
endwith
```

The following VB.NET sample displays your header using 3 levels:

```
With AxG2antt1
   .BeginUpdate()
   With .Chart
      .LevelCount = 3
```

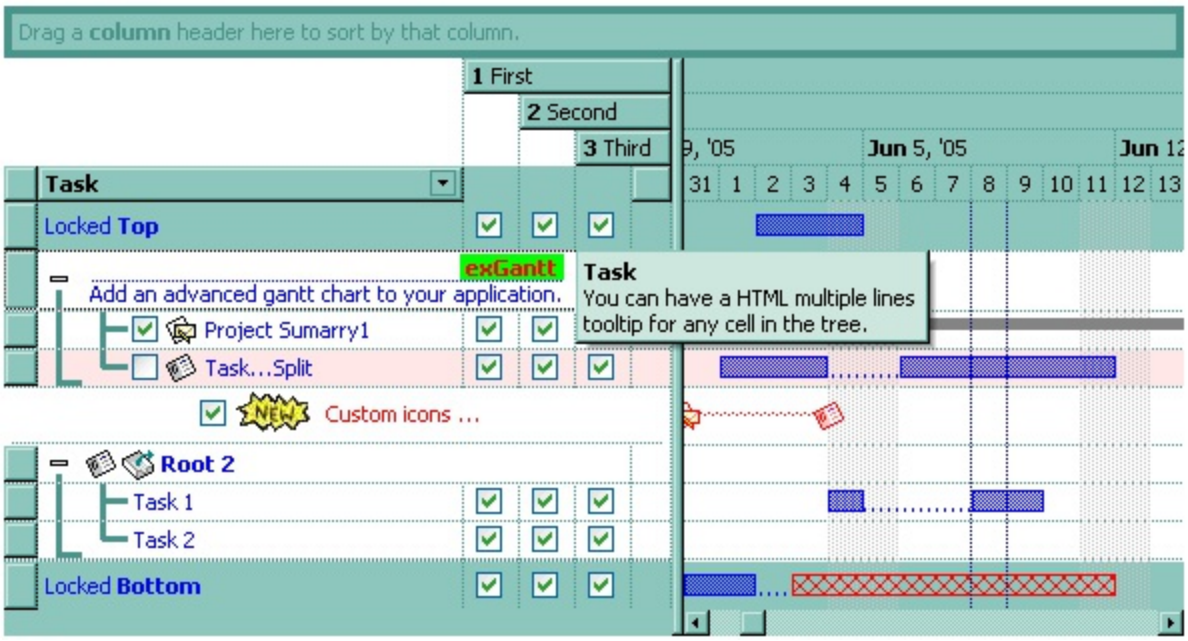
```
    With .Level(0)
      .Label = "<b><%mmm%>, <%yyyy%></b> <r>Week: <%ww%>"
      .Unit = EXG2ANTTLib.UnitEnum.exWeek
    End With
    .Level(1).Label = "<%d1%>"
    .Level(2).Label = "<%d%>"
  End With
  .EndUpdate()
End With
```

The following C# sample displays your header using 3 levels:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Chart chart = axG2antt1.Chart;
chart.LevelCount = 3;
chart.get_Level(0).Label = "<b><%mmm%>, <%yyyy%></b> <r>Week: <%ww%>";
chart.get_Level(0).Unit = EXG2ANTTLib.UnitEnum.exWeek;
chart.get_Level(1).Label = "<%d1%>";
chart.get_Level(2).Label = "<%d%>";
axG2antt1.EndUpdate();
```

The following C++ sample displays your header using 3 levels:

```
m_g2antt.BeginUpdate();
CChart chart = m_g2antt.GetChart();
chart.SetLevelCount( 3 );
chart.GetLevel(0).SetLabel(COleVariant( "<b><%mmm%>, <%yyyy%></b> <r>Week:
<%ww%>" ));
chart.GetLevel(0).SetUnit(256);
chart.GetLevel(1).SetLabel(COleVariant( "<%d1%>" ));
chart.GetLevel(2).SetLabel(COleVariant( "<%d%>" ));
m_g2antt.EndUpdate();
```



The following VB sample enumerates the levels in the chart:

```
With G2antt1.Chart
   Dim i As Long
   For i = 0 To .LevelCount - 1
      With .Level(i)
         Debug.Print .Label
      End With
   Next
End With
```

The following C++ sample enumerates the levels in the chart:

```
CChart chart = m_g2antt.GetChart();
for ( long i = 0; i < chart.GetLevelCount(); i++ )
{
   CLevel level = chart.GetLevel( i );
   OutputDebugString( V2S( &level.GetLabel() ) );
}
```

where the V2S function converts a Variant expression to a string expression:

```
static CString V2S( VARIANT* pvtDate )
{
   COleVariant vtDate;
   vtDate.ChangeType( VT_BSTR, pvtDate );
```

```
    return V_BSTR( &vtDate );
}
```

The following VB.NET sample enumerates the levels in the chart:

```
With AxG2antt1.Chart
    Dim i As Long
    For i = 0 To .LevelCount - 1
        With .Level(i)
            Debug.Write(.Label())
        End With
    Next
End With
```

The following C# sample enumerates the levels in the chart:

```
for (int i = 0; i < axG2antt1.Chart.LevelCount; i++)
{
    EXG2ANTTLib.Level level = axG2antt1.Chart.get_Level(i);
    System.Diagnostics.Debug.Write(level.Label);
}
```

The following VFP sample enumerates the levels in the chart:

```
With thisform.G2antt1.Chart
    For i = 0 To .LevelCount - 1
        With .Level(i)
            wait window nowait .Label
        EndWith
    Next
EndWith
```

property Chart.LevelFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Long

Retrieves the index of the level from the point.

Type	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in client coordinates.
Long	A long expression that indicates the index of the level from the point, or -1 if the cursor is not in the chart's header.

The LevelFromPoint property gets the level from the point. Use the Level property to access a Level object. The LevelCount property counts the number of the levels in the chart's header. Use the ItemFromPoint property to get the cell/item from the cursor. Use the ColumnFromPoint property to retrieve the column from cursor. Use the BarFromPoint property to get the bar from the point. Use the LinkFromPoint property to get the link from the point. **If the X parameter is -1 and Y parameter is -1 the ItemFromPoint property determines the handle of the item from the cursor**.

The following VB sample displays the label of the level from the cursor:

```
Private Sub G2antt1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
  With G2antt1.Chart
    Dim d As Long
    d = .LevelFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
    If d >= 0 Then
      Debug.Print .Level(d).Label
    End If
  End With
End Sub
```

The following C++ sample displays the label of the level from the point:

```
void OnMouseMoveG2antt1(short Button, short Shift, long X, long Y)
{

```

```
    CChart chart = m_g2antt.GetChart();
    long d = chart.GetLevelFromPoint( X, Y );
    if ( d >= 0 )
        OutputDebugString( V2S( &chart.GetLevel(d).GetLabel() ) );
}
```

The following VB.NET sample displays the label of the level from the point:

```
Private Sub AxG2antt1_MouseMoveEvent(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_MouseMoveEvent) Handles AxG2antt1.MouseMoveEvent
    With AxG2antt1.Chart
        Dim d As Integer = .LevelFromPoint(e.x, e.y)
        If (d >= 0) Then
            Debug.Write(.Level(d).Label)
        End If
    End With
End Sub
```

The following C# sample displays the label of the level from the point:

```
private void axG2antt1_MouseMoveEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_MouseMoveEvent e)
{
    int d = axG2antt1.Chart.get_LevelFromPoint(e.x, e.y);
    if ( d >=0 )
        System.Diagnostics.Debug.Write(axG2antt1.Chart.get_Level(d).Label );
}
```

The following VFP sample displays the label of the level from the point:

```
*** ActiveX Control Event ***
*** ActiveX Control Event ***

LPARAMETERS button, shift, x, y

with thisform.G2antt1.Chart
    d = .LevelFromPoint(x,y)
    if ( d>=0 )
        wait window nowait .Level(d).Label
```

```
    endif
endwith
```

property Chart.LinkFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Variant

Retrieves the link from the point.

Type	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in client coordinates.
Variant	A VARIANT expression that indicates the key of the link from the cursor, or empty if no link at cursor.

The LinkFromPoint property gets the link from point. **If the X parameter is -1 and Y parameter is -1 the LinkFromPoint property determines the key of the link from the cursor.** Use the Link property to access properties of the link. Use the ItemFromPoint property to get the cell/item from the cursor. Use the ColumnFromPoint property to retrieve the column from cursor. Use the FormateDate property to format a date. Use the DrawDateTicker property to draw a ticker as cursor hovers the chart's area. Use the BarFromPoint property to get the bar from the point.

The following VB sample displays the key of the link from the cursor:

```
Private Sub G2antt1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
  With G2antt1.Chart
    Debug.Print .LinkFromPoint(-1, -1)
  End With
End Sub
```

property Chart.LinksColor as Color

Specifies the color to draw the links between the bars.

Type	Description
Color	A color expression that indicates the color to draw the links between bars.

Use the LinksColor property to change the color of the links between bars. Use the AddLink method to link two bars. Use the AddBar method to add a new bar to an item. Use the AddItem method to add a new item. Use the Link(exLinkColor) property to change the color for a specific link. Use the ShowLinks property to hide all links in the chart area. Use the LinkStyle property to specify the style of the link between bars. Use the LinkWidth property to specify the width in pixels, of the pen that draws the link. The ShowLinksColor property specifies the color to display the links when link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. The ShowLinksWidth property specifies the width to show the links when the link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. The ShowLinksStyle property specifies the width to show the links when the link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars.

property Chart.LinksStyle as LinkStyleEnum

Specifies the style to draw the links between the bars.

Type	Description
LinkStyleEnum	A LinkStyleEnum expression that indicates the style of the pen to draw the links between bars.

By default, the LinksStyle property is exLinkTDot. Use the ShowLinks property to hide all links in the chart area. Use the LinksColor property to change the color of the links between bars. Use the AddLink method to link two bars. Use the AddBar method to add a new bar to an item. Use the AddItem method to add a new item. Use the Link(exLinkStyle) property to change the style for a specific link. Use the LinkWidth property to specify the width in pixels, of the pen that draws the link. The ShowLinksColor property specifies the color to display the links when link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. The ShowLinksWidth property specifies the width to show the links when the link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. The ShowLinksStyle property specifies the width to show the links when the link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars.

property Chart.LinksWidth as Long

Specifies the width in pixels of the pen to draw the links between the bars.

Type	Description
Long	A long expression that indicates the width of the pen to draw the links between bars.

By default, the LinksWidth property is 1 pixel. Use the ShowLinks property to hide all links in the chart area. Use the LinksColor property to change the color of the links between bars. Use the AddLink method to link two bars. Use the AddBar method to add a new bar to an item. Use the AddItem method to add a new item. Use the Link(exLinkWidth) property to change the width of the pen that draws a specific link. Use the LinkStyle property to specify the style of the pen that draws the link. The ShowLinksColor property specifies the color to display the links when link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. The ShowLinksWidth property specifies the width to show the links when the link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. The ShowLinksStyle property specifies the width to show the links when the link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars.

property Chart.LocAMPM as String

Retrieves the time marker such as AM or PM using the current user regional and language settings.

Type	Description
String	A String expression that indicates the time marker such as AM or PM using the current user regional and language settings.

The LocAMPM property gets the locale AM/PM indicators as indicated by current regional settings. The <%**AM/PM**%> HTML tag indicates the twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate set by the AMPM property. The <%**loc_AM/PM**%> HTML tag indicates the time marker such as AM or PM using the current user regional and language settings (LocAMPM property). The LocFirstWeekDay property indicates the first day of the week, using the current user regional and language settings. The LocMonthNames property specifies the list of name of the months, using the current user regional and language settings. The LocWeekDays property specifies the name of the days in the week, using the current user regional and language settings.

property Chart.LocFirstWeekDay as WeekDayEnum

Indicates the first day of the week, as specified in the regional settings.

Type	Description
WeekDayEnum	A WeekDayEnum expression that specifies the first day of the week, as specified in the regional settings.

The LocFirstWeekDay property indicates the first day of the week, using the current user regional and language settings. The LocMonthNames property specifies the list of name of the months, using the current user regional and language settings. The LocWeekDays property specifies the name of the days in the week, using the current user regional and language settings. The LocAMPM property gets the locale AM/PM indicators as indicated by current regional settings.

property Chart.LocMonthNames as String

Retrieves the list of month names, as indicated in the regional settings, separated by space.

Type	Description
String	A String expression that indicates the name of the months within the year, as indicated in the regional settings, separated by space.

Use the LocMonthNames property to get the name of the months as indicated by current regional settings. The <%**m1**%>, <%**m2**%>, <%**m3**%>, <%**mmmm**%> HTML tags indicate the name of the month, as appropriate set by the MonthNames property. The <%**loc_m1**%>, <%**loc_m2**%>, <%**loc_m3**%>, <%**loc_mmmm**%> HTML tags indicate the month using the current user regional and language settings (LocMonthNames property). The LocFirstWeekDay property indicates the first day of the week, as indicated in the regional settings. The LocAMPM property specifies specifies the AM and PM indicators, as indicated in the regional settings. The LocWeekDays property specifies the name of the days in the week, as indicated in the regional settings.

property Chart.LocWeekDays as String

Retrieves the list of names for each week day, as indicated in the regional settings, separated by space.

Type	Description
String	A String expression that indicates the list of names for each week day, as indicated in the regional settings, separated by space.

The LocWeekDays property gets the locale list of names for each week day as indicated by current regional settings. The <%**d1**%>, <%**d2**%>, <%**d3**%>, <%**ddd**%> or <%**dddd**%> HTML tags indicates the week day, as appropriate set by the WeekDays property. The <%**loc_d1**%>, <%**loc_d2**%>, <%**loc_d3**%>, <%**loc_ddd**%> or <%**loc_dddd**%> HTML tags indicates the week day, as appropriate set by the WeekDays property, using the current user regional and language settings (LocAMPM property). The LocFirstWeekDay property indicates the first day of the week, using the current user regional and language settings. The LocMonthNames property specifies the list of name of the months, using the current user regional and language settings. The LocAMPM property specifies the AM/PM time indicators, using the current user regional and language settings.

property Chart.MarkNow as Variant

Specifies the the current time to show in the chart.

Type	Description
Variant	A DATE expression to specify the date-time to be highlighted using the MarkNowColor property or empty if using the current date-time.

By default, the MarkNow property is empty, which indicates that the MarkNowColor property uses the current date-time to specify the date to be highlighted in the chart. The get function of the MarkNow property returns the date-time to be highlighted in the chart. You can use the MarkNow property to indicate a custom date to be shown instead the current date-time. For instance, the MarkNow = #03/13/2012 11:55AM# indicates that the MarkNowColor property marks the specified date instead the current date time. The MarkNowDelay property can be used to display the date time with a specified delay. The DateFromPoint property gets the date from the cursor position. The MarkNow/MarkNowDelay property can be used to specify the current date-time or your custom date time. When the MarkNow property is changed, the control fires the DateTimeChanged event.

The following samples shows how you can show a vertical line when user clicks the chart area.

VB6

```
Private Sub G2antt1_Click()
   With G2antt1
      With .Chart
         .MarkNowColor = RGB(255,0,0)
         .MarkNow = .DateFromPoint(-1,-1)
      End With
   End With
End Sub

With G2antt1
   With .Chart
      .FirstVisibleDate = #3/13/2012#
      .PaneWidth(False) = 0
      .LevelCount = 2
      .MarkNowColor = RGB(0,0,0)
```

```
      .MarkNowWidth = 3
      .UnitWidth = 32
      .ResizeUnitScale = exHour
   End With
End With
```

VB.NET

```
Private Sub Exg2antt1_Click(ByVal sender As System.Object) Handles Exg2antt1.Click
   With Exg2antt1
      With .Chart
         .MarkNowColor = Color.FromArgb(255,0,0)
         .MarkNow = .get_DateFromPoint(-1,-1)
      End With
   End With
End Sub

With Exg2antt1
   With .Chart
      .FirstVisibleDate = #3/13/2012#
      .set_PaneWidth(False,0)
      .LevelCount = 2
      .MarkNowColor = Color.FromArgb(0,0,0)
      .MarkNowWidth = 3
      .UnitWidth = 32
      .ResizeUnitScale = exontrol.EXG2ANTTLib.UnitEnum.exHour
   End With
End With
```

VB.NET for /COM

```
Private Sub AxG2antt1_ClickEvent(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles AxG2antt1.ClickEvent
   With AxG2antt1
      With .Chart
         .MarkNowColor = RGB(255,0,0)
         .MarkNow = .DateFromPoint(-1,-1)
      End With
```

```
    End With
End Sub

With AxG2antt1
    With .Chart
        .FirstVisibleDate = #3/13/2012#
        .PaneWidth(False) = 0
        .LevelCount = 2
        .MarkNowColor = RGB(0,0,0)
        .MarkNowWidth = 3
        .UnitWidth = 32
        .ResizeUnitScale = EXG2ANTTLib.UnitEnum.exHour
    End With
End With
```

C++

```
void OnClickG2antt1()
{
    EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown()
    EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart()
        var_Chart->PutMarkNowColor(RGB(255,0,0))
        var_Chart->PutMarkNow(var_Chart->GetDateFromPoint(-1,-1))
}

EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
    var_Chart->PutFirstVisibleDate("3/13/2012");
    var_Chart->PutPaneWidth(VARIANT_FALSE,0);
    var_Chart->PutLevelCount(2);
    var_Chart->PutMarkNowColor(RGB(0,0,0));
    var_Chart->PutMarkNowWidth(3);
    var_Chart->PutUnitWidth(32);
    var_Chart->PutResizeUnitScale(EXG2ANTTLib::exHour);
```

C++ Builder

```
void __fastcall TForm1::G2antt1Click(TObject *Sender)
{
   Exg2anttlib_tlb::IChartPtr var_Chart = G2antt1->Chart
      var_Chart->MarkNowColor = RGB(255,0,0)
      var_Chart->set_MarkNow(TVariant(var_Chart->get_DateFromPoint(-1,-1)))
}

Exg2anttlib_tlb::IChartPtr var_Chart = G2antt1->Chart;
   var_Chart->set_FirstVisibleDate(TVariant(TDateTime(2012,3,13).operator double()));
   var_Chart->set_PaneWidth(false,0);
   var_Chart->LevelCount = 2;
   var_Chart->MarkNowColor = RGB(0,0,0);
   var_Chart->MarkNowWidth = 3;
   var_Chart->UnitWidth = 32;
   var_Chart->ResizeUnitScale = Exg2anttlib_tlb::UnitEnum::exHour;
```

C#

```
private void exg2antt1_Click(object sender)
{
   exontrol.EXG2ANTTLib.Chart var_Chart = exg2antt1.Chart
      var_Chart.MarkNowColor = Color.FromArgb(255,0,0)
      var_Chart.MarkNow = var_Chart.get_DateFromPoint(-1,-1)
}
//this.exg2antt1.Click += new
exontrol.EXG2ANTTLib.exg2antt.ClickEventHandler(this.exg2antt1_Click);

exontrol.EXG2ANTTLib.Chart var_Chart = exg2antt1.Chart;
   var_Chart.FirstVisibleDate = Convert.ToDateTime("3/13/2012");
   var_Chart.set_PaneWidth(false,0);
   var_Chart.LevelCount = 2;
   var_Chart.MarkNowColor = Color.FromArgb(0,0,0);
   var_Chart.MarkNowWidth = 3;
   var_Chart.UnitWidth = 32;
   var_Chart.ResizeUnitScale = exontrol.EXG2ANTTLib.UnitEnum.exHour;
```

JavaScript

```
<SCRIPT FOR="G2antt1" EVENT="Click()" LANGUAGE="JScript">
   var var_Chart = G2antt1.Chart
      var_Chart.MarkNowColor = 255
      var_Chart.MarkNow = var_Chart.DateFromPoint(-1,-1)
</SCRIPT>

<OBJECT classid="clsid:CD481F4D-2D25-4759-803F-752C568F53B7" id="G2antt1">
</OBJECT>

<SCRIPT LANGUAGE="JScript">
   var var_Chart = G2antt1.Chart
      var_Chart.FirstVisibleDate = "3/13/2012"
      var_Chart.PaneWidth(false) = 0
      var_Chart.LevelCount = 2
      var_Chart.MarkNowColor = 0
      var_Chart.MarkNowWidth = 3
      var_Chart.UnitWidth = 32
      var_Chart.ResizeUnitScale = 65536
</SCRIPT>
```

C# for /COM

```
private void axG2antt1_ClickEvent(object sender, EventArgs e)
{
   EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart
      var_Chart.MarkNowColor = (uint)ColorTranslator.ToWin32(Color.FromArgb(255,0,0))
      var_Chart.MarkNow = var_Chart.get_DateFromPoint(-1,-1)
}
//this.axG2antt1.ClickEvent += new EventHandler(this.axG2antt1_ClickEvent);

EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.FirstVisibleDate = Convert.ToDateTime("3/13/2012");
   var_Chart.set_PaneWidth(false,0);
   var_Chart.LevelCount = 2;
   var_Chart.MarkNowColor = (uint)ColorTranslator.ToWin32(Color.FromArgb(0,0,0));
   var_Chart.MarkNowWidth = 3;
   var_Chart.UnitWidth = 32;
```

```
var_Chart.ResizeUnitScale = EXG2ANTTLib.UnitEnum.exHour;
```

X++ (Dynamics Ax 2009)

```
void onEvent_Click()
{
   COM com_Chart
   anytype var_Chart
   var_Chart = exg2antt1.Chart()
    com_Chart = var_Chart
      com_Chart.MarkNowColor(WinApi::RGB2int(255,0,0))
      com_Chart.MarkNow(com_Chart.DateFromPoint(-1,-1))
}

public void init()
{
   COM com_Chart

   anytype var_Chart

   super()

   var_Chart = exg2antt1.Chart()
    com_Chart = var_Chart
      com_Chart.FirstVisibleDate(COMVariant::createFromDate(str2Date("3/13/2012",213)))
   /*should be called during the form's activate method*/    com_Chart.PaneWidth(false,0);
      com_Chart.LevelCount(2)
      com_Chart.MarkNowColor(WinApi::RGB2int(0,0,0))
      com_Chart.MarkNowWidth(3)
      com_Chart.UnitWidth(32)
      com_Chart.ResizeUnitScale(65536/*exHour*/)

}

/*
public void activate(boolean _active)
{
```

```
   super(_active)


   exg2antt1.Chart().PaneWidth(false,0)

}
*/
```

VFP

```
*** Click event - Occurs when the user presses and then releases the left mouse button
over the tree control. ***
LPARAMETERS nop
   with thisform.G2antt1
      with .Chart
         .MarkNowColor = RGB(255,0,0)
         .MarkNow = .DateFromPoint(-1,-1)
      endwith
   endwith

with thisform.G2antt1
   with .Chart
      .FirstVisibleDate = {^2012-3-13}
      .PaneWidth(0) = 0
      .LevelCount = 2
      .MarkNowColor = RGB(0,0,0)
      .MarkNowWidth = 3
      .UnitWidth = 32
      .ResizeUnitScale = 65536
   endwith
endwith
```

dBASE Plus

```
/*
with (this.ACTIVEX1.nativeObject)
   Click = class::nativeObject_Click
endwith
```

```
*/
// Occurs when the user presses and then releases the left mouse button over the tree control.
function nativeObject_Click()
    local oG2antt,var_Chart
    oG2antt = form.Activex1.nativeObject
    var_Chart = oG2antt.Chart
        var_Chart.MarkNowColor = 0xff
        var_Chart.MarkNow = var_Chart.DateFromPoint(-1,-1)
return

local oG2antt,var_Chart

oG2antt = form.Activex1.nativeObject
var_Chart = oG2antt.Chart
    var_Chart.FirstVisibleDate = "03/13/2012"
    // var_Chart.PaneWidth(false) = 0
    with (oG2antt)
        TemplateDef = [Dim var_Chart]
        TemplateDef = var_Chart
        Template = [var_Chart.PaneWidth(false) = 0]
    endwith
    var_Chart.LevelCount = 2
    var_Chart.MarkNowColor = 0x0
    var_Chart.MarkNowWidth = 3
    var_Chart.UnitWidth = 32
    var_Chart.ResizeUnitScale = 65536
```

XBasic (Alpha Five)

```
function Click as v ()
    Dim oG2antt as P
    Dim var_Chart as P
    oG2antt = topparent:CONTROL_ACTIVEX1.activex
    var_Chart = oG2antt.Chart
        var_Chart.MarkNowColor = 255
        var_Chart.MarkNow = var_Chart.DateFromPoint(-1,-1)
```

```
end function

Dim oG2antt as P
Dim var_Chart as P

oG2antt = topparent:CONTROL_ACTIVEX1.activex
var_Chart = oG2antt.Chart
   var_Chart.FirstVisibleDate = {03/13/2012}
   ' var_Chart.PaneWidth(.f.) = 0
   oG2antt.TemplateDef = "Dim var_Chart"
   oG2antt.TemplateDef = var_Chart
   oG2antt.Template = "var_Chart.PaneWidth(False) = 0"

   var_Chart.LevelCount = 2
   var_Chart.MarkNowColor = 0
   var_Chart.MarkNowWidth = 3
   var_Chart.UnitWidth = 32
   var_Chart.ResizeUnitScale = 65536
```

Delphi 8 (.NET only)

```
procedure TWinForm1.AxG2antt1_ClickEvent(sender: System.Object; e: System.EventArgs);
begin
   with AxG2antt1 do
   begin
      with Chart do
      begin
         MarkNowColor := $ff
         MarkNow := TObject(DateFromPoint[-1,-1])
      end
   end
end;

with AxG2antt1 do
begin
   with Chart do
   begin
```

```
    FirstVisibleDate := '3/13/2012';
    PaneWidth[False] := 0;
    LevelCount := 2;
    MarkNowColor := $0;
    MarkNowWidth := 3;
    UnitWidth := 32;
    ResizeUnitScale := EXG2ANTTLib.UnitEnum.exHour;
  end;
end
```

Delphi (standard)

```
procedure TForm1.G2antt1Click(ASender: TObject; );
begin
  with G2antt1 do
  begin
    with Chart do
    begin
      MarkNowColor := $ff
      MarkNow := OleVariant(DateFromPoint[-1,-1])
    end
  end
end;

with G2antt1 do
begin
  with Chart do
  begin
    FirstVisibleDate := '3/13/2012';
    PaneWidth[False] := 0;
    LevelCount := 2;
    MarkNowColor := $0;
    MarkNowWidth := 3;
    UnitWidth := 32;
    ResizeUnitScale := EXG2ANTTLib_TLB.exHour;
  end;
end
```

Visual Objects

```
METHOD OCX_Exontrol1Click() CLASS MainDialog
   local var_Chart as IChart
   var_Chart := oDCOCX_Exontrol1:Chart
      var_Chart:MarkNowColor := RGB(255,0,0)
      var_Chart:MarkNow := var_Chart:[DateFromPoint,-1,-1]
RETURN NIL

local var_Chart as IChart

var_Chart := oDCOCX_Exontrol1:Chart
   var_Chart:FirstVisibleDate := SToD("20120313")
   var_Chart:[PaneWidth,false] := 0
   var_Chart:LevelCount := 2
   var_Chart:MarkNowColor := RGB(0,0,0)
   var_Chart:MarkNowWidth := 3
   var_Chart:UnitWidth := 32
   var_Chart:ResizeUnitScale := exHour
```

PowerBuilder

```
/*begin event Click() - Occurs when the user presses and then releases the left mouse
button over the tree control.*/
/*
   OleObject oG2antt,var_Chart
   oG2antt = ole_1.Object
   var_Chart = oG2antt.Chart
      var_Chart.MarkNowColor = RGB(255,0,0)
      var_Chart.MarkNow = var_Chart.DateFromPoint(-1,-1)
*/
/*end event Click*/

OleObject oG2antt,var_Chart

oG2antt = ole_1.Object
var_Chart = oG2antt.Chart
   var_Chart.FirstVisibleDate = 2012-03-13
```

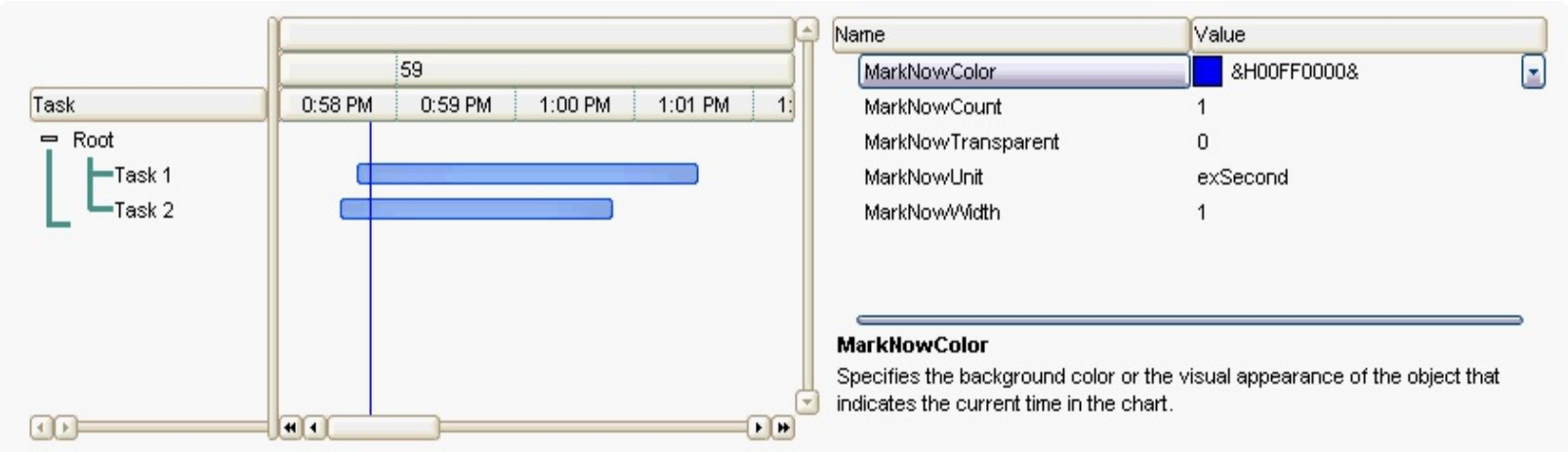
```
var_Chart.PaneWidth(false,0)
var_Chart.LevelCount = 2
var_Chart.MarkNowColor = RGB(0,0,0)
var_Chart.MarkNowWidth = 3
var_Chart.UnitWidth = 32
var_Chart.ResizeUnitScale = 65536
```

property Chart.MarkNowColor as Color

Specifies the background color or the visual appearance of the object that indicates the current time in the chart.

Type	Description
Color	A color expression that specifies the background color to show the position of the current date-time. The last 7 bits in the high significant byte of the color indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

By default, the MarkNowColor property is 0. The control's chart shows the position of the current date-time, only if the MarkNowColor property is not zero (0). Use the MarkNowColor properties to show the current date-time in the control's chart. The MarkNowUnit property specifies the unit of time to count for. For instance, you can show the current date-time from current second, to next second, from minute to next minute, and so on. Use the MarkNowCount property to specify the number of units of date-time to count from. For instance, you can show the current date-time from 5 seconds to 5 seconds, and so on. The MarkNowWidth property specifies the width in pixels of the vertical bar that shows the current date-time. The MarkNowTransparent property specifies the percent of transparency to show the vertical bar that indicates the current date-time. The MarkTodayColor property highlights the current day only. The control fires the DateTimeChanged event when the current date-time is changed. Use the SelectDate property to select a date by clicking the chart's header. Use the MarkTimeZone method to highlight different time-zones. You can use the MarkNow property to indicate a custom date to be shown instead the current date-time. The MarkNowDelay property can be used to display the date time with a specified delay.



This screen shot shows the vertical bar that indicates the current date-time. The bar is

automatically updated each second, unless the MarkNowUnit property is not changed to exMinute, when the vertical bar is updated each minute.

property Chart.MarkNowCount as Long

Specifies the number of time units to count while highlighting the current time.

Type	Description
Long	A long expression that specifies the width in pixels of the vertical bar that shows the current date-time in the control's chart.

By default, the MarkNowCount property is 1. The control's chart shows the position of the current date-time, only if the MarkNowColor property is not zero (0). Use the MarkNowCount property to specify the number of units of date-time to count from. For instance, you can show the current date-time from 5 seconds to 5 seconds, and so on. The MarkNowWidth property specifies the width in pixels of the vertical bar that shows the current date-time. The MarkNowUnit property specifies the unit of time to count for. For instance, you can show the current date-time from current second, to next second, from minute to next minute, and so on. The MarkNowTransparent property specifies the percent of transparency to show the vertical bar that indicates the current date-time. The MarkTodayColor property highlights the current day only. The control fires the DateTimeChanged event when the current date-time is changed.

property Chart.MarkNowDelay as Double

Specifies the delay to show the current time in the chart.

Type	Description
Double	A numeric expression that indicates the delay to display the date time when using the MarkNowColor property. The negative value of the MarkNowDelay property is added to the date-time to indicate the new time to be represented on the chart instead. The value could be 1 which indicates 1 day, 1/24 which indicates 1 hour, and 1/24/60 indicates 1 minute, and so on

By default the MarkNowDelay property is 0, which indicates that the current date-time or specified date-time (MarkNow property) is marked. For instance, if the MarkNowDelay property is 1, the date time to be highlighted is the current date time minus 1 day. The 1/24 (0.04167) indicates 1 hour, and 1/24/60 (00069444) indicates 1 minute, and so on. You can use the MarkNowDelay to specify a different point for your date time. You can use the MarkNow property to indicate a custom date to be shown instead the current date-time.

The following VB sample specifies a new starting point for marking the date time:

```
Private Sub G2antt1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
  With G2antt1.Chart
    Dim d As Double
    d = .DateFromPoint(-1, -1)
    If (d = 0) Then
      .MarkNowDelay = 0
    Else
      .MarkNowDelay = (.MarkNow + .MarkNowDelay) - d
    End If
  End With
End Sub
```

Once you click a date within the chart, the vertical line will be show at your clicked position, and if the MarkNow property is Empty (by default), the counting will start from your clicked position.

property Chart.MarkNowTransparent as Long

Specifies the percent of the transparency to display the object that marks the current time.

Type	Description
Long	A long expression that specifies the percent of transparency to show the vertical bar that indicates the current date-time in the control's chart. 0 means opaque, 50 means semi-transparent, and 100 means transparent.

By default, the MarkNowTransparent property is 0. The control's chart shows the position of the current date-time, only if the MarkNowColor property is not zero (0). The MarkNowTransparent property specifies the percent of transparency to show the vertical bar that indicates the current date-time. The MarkNowUnit property specifies the unit of time to count for. For instance, you can show the current date-time from current second, to next second, from minute to next minute, and so on. Use the MarkNowCount property to specify the number of units of date-time to count from. For instance, you can show the current date-time from 5 seconds to 5 seconds, and so on. The MarkNowWidth property specifies the width in pixels of the vertical bar that shows the current date-time. The MarkTodayColor property highlights the current day only. The control fires the DateTimeChanged event when the current date-time is changed.

property Chart.MarkNowUnit as UnitEnum

Retrieves or sets a value that indicates the base time unit while highlighting the current time.

Type	Description
UnitEnum	A UnitEnum expression that specifies the date-time unit to show the current date-time in the control's chart.

By default, the MarkNowUnit property is exSecond. The control's chart shows the position of the current date-time, only if the MarkNowColor property is not zero (0). Use the MarkNowColor properties to show the current date-time in the control's chart. Use the MarkNowUnit property to specify the unit of time to count for. For instance, you can show the current date-time from current second, to next second, from minute to next minute, and so on. Use the MarkNowCount property to specify the number of units of date-time to count from. For instance, you can show the current date-time from 5 seconds to 5 seconds, and so on. The MarkNowWidth property specifies the width in pixels of the vertical bar that shows the current date-time. The MarkNowTransparent property specifies the percent of transparency to show the vertical bar that indicates the current date-time. The MarkTodayColor property highlights the current day only. The control fires the DateTimeChanged event when the current date-time is changed.

property Chart.MarkNowWidth as Long

Specifies the width in pixels of the object that shows the current time.

Type	Description
Long	A long expression that specifies the width in pixels of the vertical bar that shows the current date-time in the control's chart. If the MarkNowWidth property is 0 or negative, the control computes the required width so current date-time is shown based on the MarkNowUnit and MarkNowCount properties. For instance, in this case, if your chart displays seconds, and the MarkNowCount property is 2, the width of the vertical bar that shows the current date-time is UnitWidth multiplied by 2 (the space required in the control's chart to display 2 seconds) .

By default, the MarkNowWidth property is 1. The control's chart shows the position of the current date-time, only if the MarkNowColor property is not zero (0). The MarkNowWidth property specifies the width in pixels of the vertical bar that shows the current date-time. The MarkNowUnit property specifies the unit of time to count for. For instance, you can show the current date-time from current second, to next second, from minute to next minute, and so on. Use the MarkNowCount property to specify the number of units of date-time to count from. For instance, you can show the current date-time from 5 seconds to 5 seconds, and so on. The MarkNowTransparent property specifies the percent of transparency to show the vertical bar that indicates the current date-time. The MarkTodayColor property highlights the current day only. The control fires the DateTimeChanged event when the current date-time is changed.

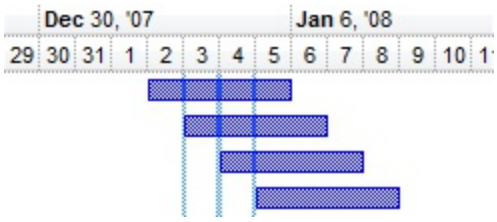
property Chart.MarkSelectDateColor as Color

Retrieves or sets a value that indicates the color to mark the selected date in the chart.

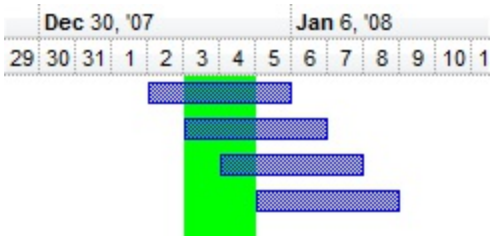
Type	Description
Color	A Color expression that indicates the color being used to highlight the selected dates. The last 7 bits in the high significant byte of the color indicates the identifier of the skin being used to display the selected dates. Use the Add method to add new skins to the control.

The MarkSelectDateColor property specifies the color being used to highlight the selected dates. The AllowSelectDate property specifies whether the user can select a date by clicking the chart's header. The user can select dates by clicking the chart's header. You can select multiple dates keeping the CTRL key and clicking a new date. Use the SelectLevel property to specify the area being highlighted when a date is selected. Use the SelectDate property to select dates programmatically. The SelectedDates property can be used to retrieve all selected dates, or to select a collection of dates. By default, the MarkSelectDateColor is blue (as your control panel indicates the color for the selected items). The selected dates are not marked if the MarkSelectDateColor property has the same value as BackColor property in the Chart object. The MarkTodayColor property specifies the color to mark the today date. Use the LevelFromPoint property to get the index of the level from the cursor. Use the DateFromPoint property to retrieve the date from the cursor. The ChartEndChanging(exSelectDate) event notifies your application when the user selects a new date by clicking the header of the chart.

The following screen shot shows the selected dates (*Dec 2 and Dec 4*) being colored with the default format:

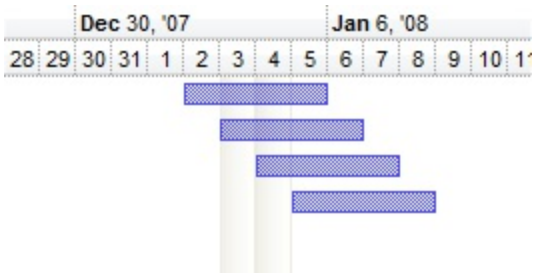


The following screen shot shows the selected dates (*Dec 2 and Dec 4*) being colored with a solid color (2130771712, 0x7F00FF00):

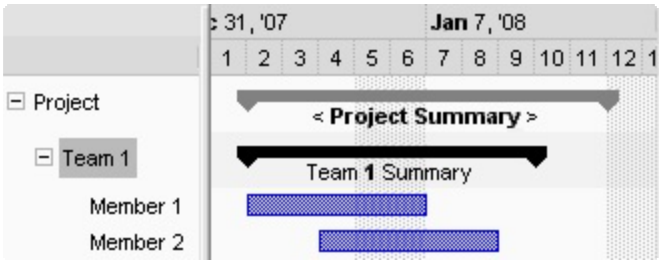


The following screen shot shows the selected dates (*Dec 2 and Dec 4*) being colored with

this EBN file (16777216, 0x1000000):



The following screen show how a new date gets selected once the user clicks a date in the chart's header:



Your application can provide some options to help user while performing moving or resizing the bars at runtime as follow:

- grid lines, that can be shown only when moving or resizing, using the ChartStartChanging and ChartEndChanging events
- select date, to specify the margins of the area you want to highlight
- ticker, that shows the cursor's position in the chart, or while resizing, it shows the size and the position of the bar
- ability to specify a resizing/moving unit, different that the displayed one ie while the chart displays days, you can specify the resizing unit on hours.
- inside zoom, that can be used to magnify the portion of the chart being selected

The following VB sample shows how you can change the color for selected dates to be solid:

```
With G2antt1
   .BeginUpdate
   With .Chart
      .FirstVisibleDate = #1/1/2008#
      .MarkTodayColor = .BackColor
      .LevelCount = 2
      .MarkSelectDateColor = &H7FFDF9F4  ' The color is actually FDF9F4 as
BBGGRR format
      .SelectLevel = 1
      .SelectDate(#1/2/2008#) = True
```

```
    .SelectDate(#1/3/2008#) = True
  End With
End With
```

The following VB.NET sample shows how you can change the color for selected dates to be solid:

```
With AxG2antt1
  .BeginUpdate
  With .Chart
    .FirstVisibleDate = #1/1/2008#
    .MarkTodayColor = .BackColor
    .LevelCount = 2
    .MarkSelectDateColor = &H7FFDF9F4  ' The color is actually FDF9F4 as
BBGGRR format
    .SelectLevel = 1
    .SelectDate(#1/2/2008#) = True
    .SelectDate(#1/3/2008#) = True
  End With
End With
```

The following C++ sample shows how you can change the color for selected dates to be solid:

```
/*
  Copy and paste the following directives to your header file as
  it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

  #import <ExG2antt.dll>
  using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
  var_Chart->PutFirstVisibleDate("1/1/2008");
  var_Chart->PutMarkTodayColor(var_Chart->GetBackColor());
  var_Chart->PutLevelCount(2);
```

```
var_Chart->PutMarkSelectDateColor(0x7FFDF9F4); // The color is actually FDF9F4 as BBGGRR format
var_Chart->PutSelectLevel(1);
var_Chart->PutSelectDate("1/2/2008",VARIANT_TRUE);
var_Chart->PutSelectDate("1/3/2008",VARIANT_TRUE);
```

The following C# sample shows how you can change the color for selected dates to be solid:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
var_Chart.FirstVisibleDate = "1/1/2008";
var_Chart.MarkTodayColor = var_Chart.BackColor;
var_Chart.LevelCount = 2;
var_Chart.MarkSelectDateColor = 0x7FFDF9F4;  // The color is actually FDF9F4 as BBGGRR format
var_Chart.SelectLevel = 1;
var_Chart.set_SelectDate("1/2/2008",true);
var_Chart.set_SelectDate("1/3/2008",true);
```

The following VFP sample shows how you can change the color for selected dates to be solid:

```
with thisform.G2antt1
  .BeginUpdate
  with .Chart
    .FirstVisibleDate = {^2008-1-1}
    .MarkTodayColor = .BackColor
    .LevelCount = 2
    .MarkSelectDateColor = 0x7FFDF9F4;  // The color is actually FDF9F4 as BBGGRR format
    .SelectLevel = 1
    .SelectDate({^2008-1-2}) = .T.
    .SelectDate({^2008-1-3}) = .T.
  endwith
endwith
```

method Chart.MarkTimeZone (Key as Variant, [Start as Variant], [End as Variant], [Color as Variant], [Options as Variant])

Highlights a specified time zone from start to end with a different background color, pattern, transparency, HTML captions and so on.

Type	Description
Key as Variant	A String expression that specifies a key to identify the zone. The String must be not empty, else the zone is not highligted. If a zone with the same key is already added, the current call changes not missing parameters.
Start as Variant	A DATE expression that specifies the starting point for the zone.
End as Variant	A DATE expression that specifies the ending point for the zone.
Color as Variant	A Color expression that specifies the date-time zone to be highlighted. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.
Options as Variant	A String expression that specifies options to mark the zone, as explained bellow.

By default, the chart displays no date-time zones, unless you are calling the MarkTimeZone method. A zone can be used to highlight a range of dates, specifying the start and end zone. Use the RemoveTimeZone method to delete the time zone being added previously using the MarkTimeZone method. The TimeZoneFromPoint property retrieves the key of the time-zone from the cursor. The TimeZoneInfo property retrieves information about the time-zone giving its key. The MarkTodayColor property specifies the color to mark the today date. Use the SelectDate property to select a date by clicking the chart's header. Use the MarkNowColor property to show a vertical bar that indicates the current date-time in the control's chart, from seconds to seconds, minutes, and so on. The OverviewShowMarkTimeZones property shows the marked time-zones on the control's overview map.

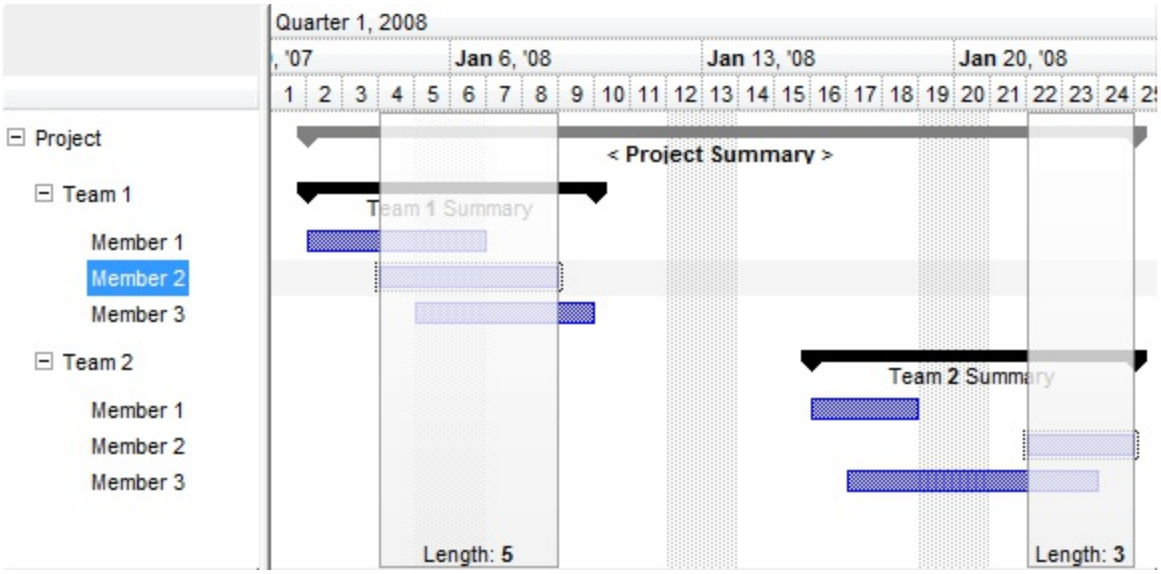
A date-time zone can display a:

- solid background color or using an EBN to display a skin object
- pattern (by default, is exPatternSolid)

- transparency (by default, it is 0 which means opaque while, 50 means semi-transparent and 100 means transparent)

Also, a zone can display multiple HTML captions aligned to any corner of the zone. By default, all HTML captions are aligned to the bottom center area.

The following screen shot shows two zones being marked :



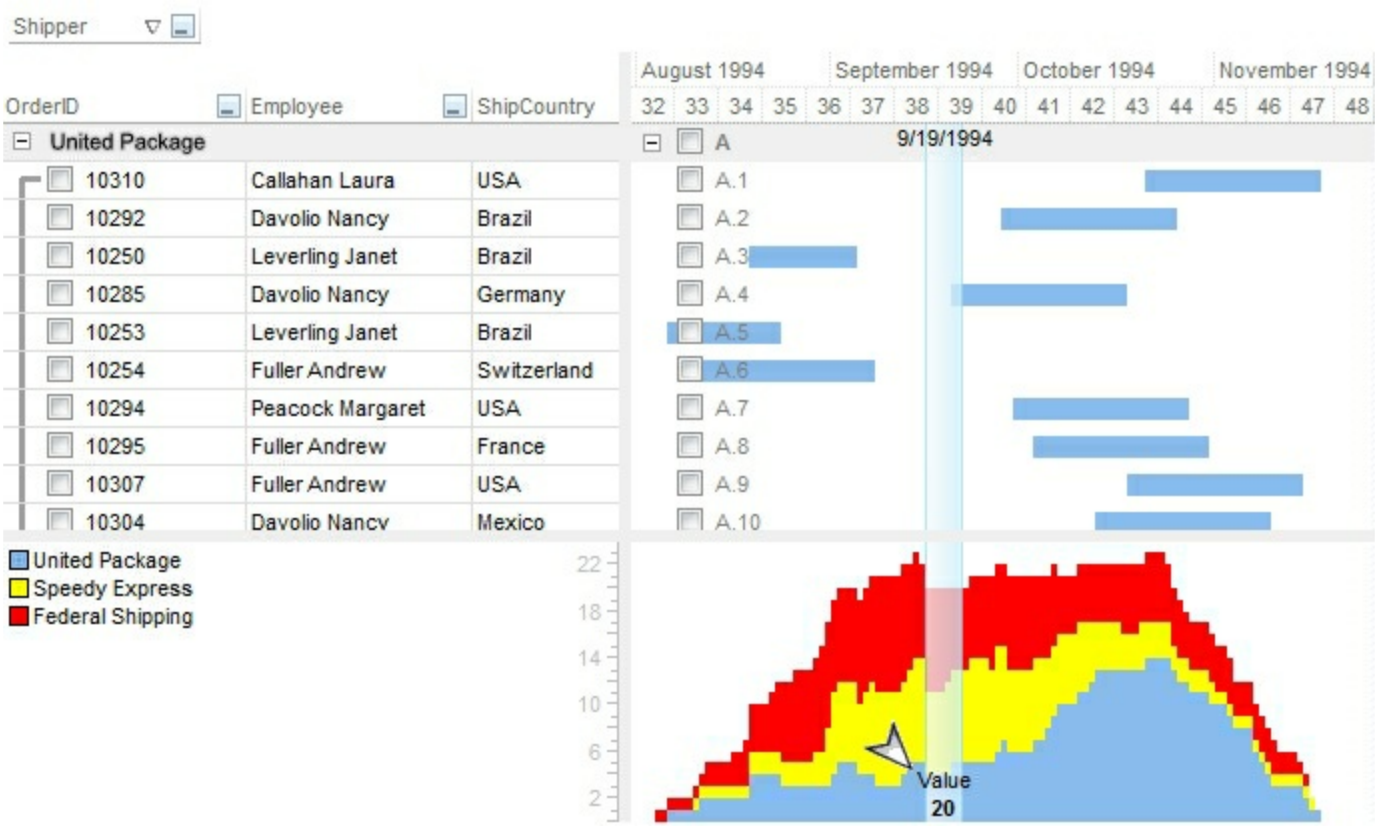
The Options parameter contains a list of fields separated by ; character as following:

- Transparency[:Width], indicating the transparency : fixed width zone. The transparency is value from 0 to 100, (0 by default or missing - opaque, 50 - semi-transparent, 100 - hidden or fully transparent). For instance,MarkTimeZone "Z1",#1/4/2010 10:00:00 AM#,#1/4/2010 10:00:00 AM#,RGB(255,0,0),"50:3;;zone" adds a red vertical line of 3-pixels wide, with a semi-transparent color, that displays "zone" at Jan 4th, 2010, at 10:00 AM. If Width field is missing, the zone is delimited by the Start and End parameters, with a visible width of minimum 1-pixel wide. If the Width field is specified, it indicates the actually width of the zone which is displayed at (Start + End) / 2.
- Pattern as described here, (1/exPatternSolid by default or missing - solid)
- HTML caption to be displayed in the zone, that may contains HTML tags as listed here on exHTML. The caption may include anchor elements that fires the AnchorClick event when it is clicked.
- Alignment of the HTML caption with the values listed here.

The list continues by pair of HTML/Alignment elements. For instance, the ";;Caption 1;1;Caption 2;17;Caption 3" specifies a three captions to be shown in the zone. For instance, the "50;;text to be shown,1" indicates that the zone is semi-transparent (5), displays a solid color (the pattern field is missing), and it displays "text to be shown" in the upper-center position of the zone.

By default, the text of the mark-time zone is not displayed in the chart's histogram area (

HistogramVisible property), but you can provide multiple HTML caption to the histogram panels, after a ";;;" sequence as in the following sample: "25;;9/19/1994;1;;;Value
<c><b>20;33" and it should look like follows:

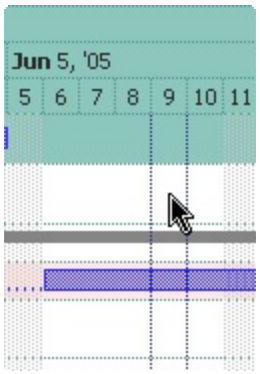


property Chart.MarkTodayColor as Color

Retrieves or sets a value that indicates the color to mark today in the chart.

Type	Description
Color	A Color expression that indicates the color being used to mark the today date.

The MarkTodayColor property specifies the color to mark the today date. If the MarkTodayColor property is the same as the BackColor property, the today date is not marked. Use the NonworkingDays property to specify the nonworking days in a week. Use the DrawTickLines property to specify whether the grid lines between time units in the level are visible or hidden. Use the DrawGridLines property to specify whether the control draws the grid lines in the chart's area. Use the GridLineColor property to specify the color for grid lines. Use the DrawGridLines property to specify whether the control draws the grid lines in the items area. Use the DrawGridLines property to draw grid lines for a specified level. Use the MarkSelectDateColor property to highlight the selected dates. Use the SelectDate property to select a date by clicking the chart's header. Use the MarkNowColor property to show a vertical bar that indicates the current date-time in the control's chart, from seconds to seconds, minutes, and so on. Use the MarkTimeZone method to highlight different time-zones.



property Chart.MaxUnitWidth as Long

Specifies the maximum value for Chart.UnitWidth property while enlarge or zoom-in/zoom-out operation is performed.

Type	Description
Long	A long expression that indicates the maximum value in pixels, for the Chart's UnitWidth property when resizing or enlarging the chart is performed.

By default, the MaxUnitWidth property is 36 pixels. The MaxUnitWidth property has effect only if the AllowResizeChart property is not-zero. The MaxUnitWidth property could be negative, in this case, the UnitWidth property has no right margin. The MaxUnitWidth property indicates the maximum value for the UnitWidth property when resizing/enlarging is performed. The MinUnitWidth property indicates the minimum value for the UnitWidth property when resizing/enlarging is performed. For instance, when zoom-in or zoom-out if the chart's UnitWidth property reaches the MaxUnitWidth property the chart's UnitScale property is changed to prev time-scale available (AllowResizeChart property includes the exAllowChangeUnitScale).

property Chart.MinUnitWidth as Long

Specifies the minimum value for Chart.UnitWidth property while enlarge or zoom-in/zoom-out operation is performed.

Type	Description
Long	A long expression that indicates the minimum value in pixels, for the Chart's UnitWidth property when resizing or enlarging the chart is performed.

By default, the MinUnitWidth property is 12 pixels. The MinUnitWidth property has effect only if the AllowResizeChart property is not-zero. The MinUnitWidth property should be greater than 0. The MinUnitWidth property indicates the minimum value for the UnitWidth property when resizing/enlarging is performed. The MaxUnitWidth property indicates the maximum value for the UnitWidth property when resizing/enlarging is performed. For instance, when zoom-in or zoom-out if the chart's UnitWidth property reaches the MinUnitWidth property the chart's UnitScale property is changed to next time-scale available (AllowResizeChart property includes the exAllowChangeUnitScale).

property Chart.MonthNames as String

Retrieves or sets a value that indicates the list of month names, separated by space.

Type	Description
String	A String expression that indicates the name of the months in the year, separated by spaces.

By default, the MonthNames property is "January February March April May June July August September October November December". The order of months is January, February, and so on. Use the MonthNames property to specify the name of the months in the year. The FormatDate property formats a date. Use the AMPM property to specify the name of the AM and PM indicators. Use the Label property to specify the label being displayed in the level. Use the Label property to specify the predefined format for a level based on the unit time. Use the ToolTip property to specify the tool tip being displayed when the cursor hovers the level. Use the FirstWeekDay property to specify the first day in the week.

The MonthNames property specifies the name of the months in the year for the following built-in tags:

- <%**m1**%> - First letter of the month (J to D).
- <%**m2**%> - First two letters of the month (Ja to De).
- <%**m3**%> - First three letters of the month (Jan to Dec).
- <%**mmm**%> - First three letters of the month (Jan to Dec).
- <%**mmmm**%> - Full name of the month (January to December).

The following VB sample assigns Romanian name for months in the year:

```
With G2antt1.Chart
   .MonthNames = "Ianuarie Februarie Martie Aprilie Mai Iunie Iulie August Septembrie
Octombrie Noiembrie Decembrie"
End With
```

The following C++ sample assigns Romanian name for months in the year:

```
m_g2antt.GetChart().SetMonthNames( "Ianuarie Februarie Martie Aprilie Mai Iunie Iulie
August Septembrie Octombrie Noiembrie Decembrie" );
```

The following VB.NET sample assigns Romanian name for months in the year:

```
With AxG2antt1.Chart
   .MonthNames = "Ianuarie Februarie Martie Aprilie Mai Iunie Iulie August Septembrie
```

```
Octombrie Noiembrie Decembrie"
End With
```

The following C# sample assigns Romanian name for months in the year:

```
axG2antt1.Chart.MonthNames = "Ianuarie Februarie Martie Aprilie Mai Iunie Iulie August Septembrie Octombrie Noiembrie Decembrie";
```

The following VFP sample assigns Romanian name for months in the year:

```
With thisform.G2antt1.Chart
    .MonthNames = "Ianuarie Februarie Martie Aprilie Mai Iunie Iulie August Septembrie Octombrie Noiembrie Decembrie"
EndWith
```

property Chart.NextDate (Date as Date, Unit as UnitEnum, [Count as Variant]) as Date

Gets the next date based on the unit.

Type	Description
Date as Date	A Date expression that indicates the start date.
Unit as UnitEnum	An UnitEnum expression that indicates the time unit to change the date.
Count as Variant	A long expression that indicates the number of time units
Date	A Date expression that indicates the result.

Use the NextDate property to retrieve the next or previous date giving a specified time unit. The FirstVisibleDate property indicates the first visible date in the chart. Use the ScrollTo method to ensure that a specified date fits the chart's client area. Use the FormatDate property to format a date to a specified format.

The following VB sample displays the next day as "Tue, May 31, 2005":

```
With G2antt1.Chart
   Debug.Print .FormatDate(.NextDate(.FirstVisibleDate, exDay, 2), "<%ddd%>,
<%mmmm%> <%d%>, <%yyyy%>")
End With
```

The following C++ sample displays the next day as "Tue, May 31, 2005":

```
CChart chart = m_g2antt.GetChart();
DATE d = chart.GetNextDate( V2D( &chart.GetFirstVisibleDate() ), 4096, COleVariant(
(long)1 ) );
CString strFormat = chart.GetFormatDate( d, "<%ddd%>, <%mmmm%> <%d%>,
<%yyyy%>" );
OutputDebugString( strFormat );
```

where the V2D function converts a Variant expression to a DATE expression:

```
static DATE V2D( VARIANT* pvtDate )
{
   COleVariant vtDate;
   vtDate.ChangeType( VT_DATE, pvtDate );
   return V_DATE( &vtDate );
```

```
}
```

The following VB.NET sample displays the next day as "Tue, May 31, 2005":

```
With AxG2antt1.Chart
    Debug.Write(.FormatDate(.NextDate(.FirstVisibleDate, EXG2ANTTLib.UnitEnum.exDay,
2), "<%ddd%>, <%mmmm%> <%d%>, <%yyyy%>"))
End With
```

The following C# sample displays the next day as "Tue, May 31, 2005":

```
DateTime d = Convert.ToDateTime(
axG2antt1.Chart.get_NextDate(Convert.ToDateTime(axG2antt1.Chart.FirstVisibleDate),
EXG2ANTTLib.UnitEnum.exDay, 1) );
String strFormat = axG2antt1.Chart.get_FormatDate(d, "<%ddd%>, <%mmmm%>
<%d%>, <%yyyy%>");
System.Diagnostics.Debug.Write(strFormat);
```

The following VFP sample displays the next day as "Tue, May 31, 2005":

```
With thisform.G2antt1.Chart
    wait window nowait .FormatDate(.NextDate(.FirstVisibleDate, 4096, 2), "<%ddd%>,
<%mmmm%> <%d%>, <%yyyy%>")
EndWith
```

property Chart.NonworkingDays as Long

Retrieves or sets a value that indicates the non-working days, for each week day a bit.

Type	Description
Long	A long expression that indicates the non-working days in a week.

By default, the NonworkingDays property is 65 (Saturday(s) and Sunday(s)). The non-working days are shown using the NonworkingDaysPattern and the NonworkingDaysColor which defines the pattern and the color, when the base level of the chart displays days, if the ShowNonworkingUnits property is True (by default). Use the ShowNonworkingUnits property to display or hide the non-working units as hours or days in your chart. Use the NonworkingHours property to indicate non-working hours in a day. Use the ItemNonworkingUnits property to specify different non-working zones for different items. Use the UnitWidthNonworking property to specify a different width for non-working units in the base level. Use the ShowNonworkingUnits property to hide the non-working units. The IsNonworkingDate property indicates whether the giving date-time is a working or non-working unit.

The control supports the following ways of specify the non-working parts for items:

- NonworkingDays and NonworkingHours properties indicate the nonworking parts of the chart being applied to all items with the exception of those that use the ItemNonworkingUnits property.
- AddNonworkingDate method adds custom dates as being nonworking date which is applied to all items with the exception of those that use the ItemNonworkingUnits property.
- ItemNonworkingUnits property defines the repetitive expression to specify the non-working parts in the item.
- ItemBar(exBarTreatAsNonworking) indicates whether the bar defines actually the non-working part of the item in addition to ItemNonworkingUnits property (which is required also)

You can select the non-working week days in the following table (In Internet Explorer, you have to allow running the script on this page).

	Saturday	Friday	Thursday	Wednesday	Tuesday	Monday	Sunday
Value	64	32	16	8	4	2	1
Bit	☑	☐	☐	☐	☐	☐	☑

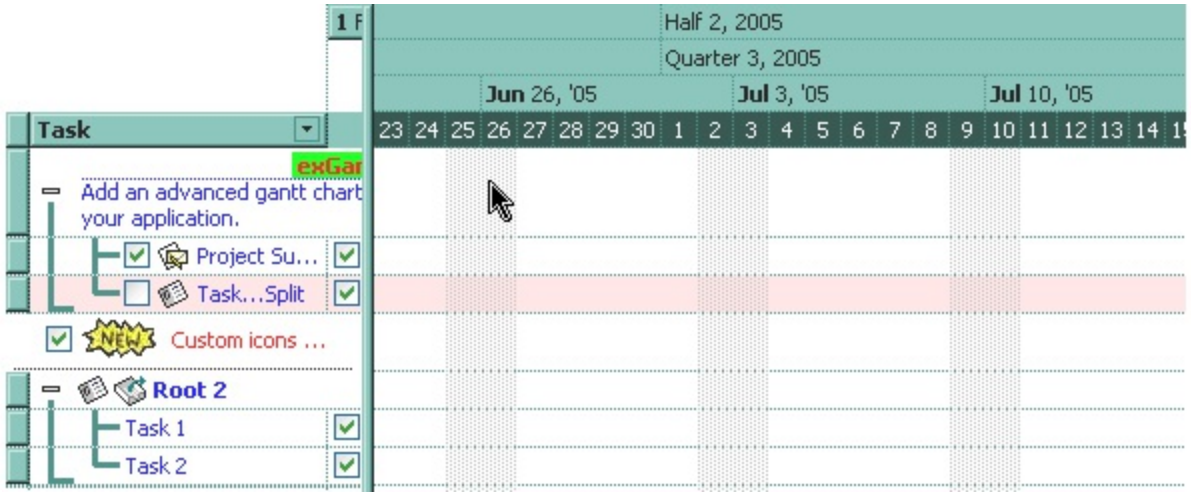
Click the Bit row for non-working days and the value for property is: [], [](hexa), [](octal), [] (binary)

The last significant byte in the NonworkingDays expression has the following meaning:

-	Sa	Fr	Th	We	Tu	Mo	Su
0	X	X	X	X	X	X	X
128	64	32	16	8	4	2	1

where **X** could be 1 (nonworking day) or 0 (working day), **Sa** means Saturday, **Fr** means Friday, and so on. For instance, the 65 value means Saturday and Sunday are non-working days. Use the AddNonworkingDate method to add custom dates as being nonworking date.

Use the ShowNonworkingDates property to show or hide the nonworking dates in the control's chart area. Use the NonworkingDaysPattern property to specify the pattern being used to fill non-working days. The NonworkingDaysColor property specifies the color being used to fill the non-working days. For instance, if the NonworkingDaysPattern is exPatternEmpty the non-working days are not highlighted. Use the MarkTodayColor property to specify the color to mark the today date. Use the DrawGridLines property to specify whether the control draws the grid lines in the chart's area. Use the GridLineColor property to specify the color for grid lines. Use the DrawGridLines property to specify whether the control draws the grid lines in the items area. Use the DrawGridLines property to draw grid lines for a specified level. Use the IsNonworkingDate property to retrieve a value that indicates whether a date is marked as nonworking day. Use the Add("A:B") to add a bar that displays the bar A in the working area, and B in non-working areas. Use the ItemBar(exBarWorkingCount) property to specify the count of working units in the bar. Use the ItemBar(exBarNonWorkingCount) property to specify the count of non-working units in the bar. Use the ItemBar(exBarKeepWorkingCount) property to specify whether the ItemBar(exBarWorkingCount) property is kept constant while user moves a bar at runtime. Use ItemBar(exBarWorkingUnits) or ItemBar(exBarWorkingUnitsAsString) property to retrieve the working parts of the bar. Use ItemBar(exBarNonWorkingUnits) or ItemBar(exBarNonWorkingUnitsAsString) property to retrieve the non-working parts of the bar.



The following VB sample retrieves the value to indicate Sunday and Monday as being non-working days:

```
With G2antt1.Chart
    .NonworkingDays = 2 ^ (EXG2ANTTLibCtl.exSunday) Or 2 ^
(EXG2ANTTLibCtl.exMonday)
End With
```

The following C++ sample retrieves the value to indicate Sunday and Monday as being non-working days:

```
m_g2antt.GetChart().SetNonworkingDays( 1 << ( EXG2ANTTLib::exSunday ) | 1 << (
EXG2ANTTLib::exMonday ) );
```

where the #import <exg2antt.dll> must be called to insert definitions for types in the control's type library.

The following VB.NET sample retrieves the value to indicate Sunday and Monday as being non-working days:

```
With AxG2antt1.Chart
    .NonworkingDays = 2 ^ (EXG2ANTTLib.WeekDayEnum.exSunday) Or 2 ^
(EXG2ANTTLib.WeekDayEnum.exMonday)
End With
```

The following C# sample retrieves the value to indicate Sunday and Monday as being non-working days:

```
axG2antt1.Chart.NonworkingDays = 1 <<
(Convert.ToInt32(EXG2ANTTLib.WeekDayEnum.exSunday) ) | 1 <<
(Convert.ToInt32(EXG2ANTTLib.WeekDayEnum.exMonday));
```

The following VFP sample retrieves the value to indicate Sunday and Monday as being non-working days:

```
with thisform.G2antt1.Chart
    .NonworkingDays = 2 ^ 0 + 2 ^ 1
endwith
```

property Chart.NonworkingDaysColor as Color

Retrieves or sets a value that indicates the color to fill the non-working days.

Type	Description
Color	A Color expression that indicates the color to fill the non-working days.

Use the NonworkingDaysColor property to specify the color being used by the NonworkingDaysPattern property. Use the NonworkingDays property to specify the nonworking days in a week. Use the AddNonworkingDate method to add custom dates as nonworking days. Use the NonworkingDaysPattern property to specify the pattern to fill the non-working days. Use the ShowNonworkingDates property to show or hide the nonworking dates in the control's chart area. For instance, if the NonworkingDaysPattern is exPatternEmpty the non-working days are not highlighted.

The following VB sample marks Sunday and Monday days on red:

```
With G2antt1.Chart
   .NonworkingDays = 2 ^ (EXG2ANTTLibCtl.exSunday) Or 2 ^
(EXG2ANTTLibCtl.exMonday)
   .NonworkingDaysColor = RGB(255, 0, 0)
End With
```

The following C++ sample sample marks Sunday and Monday days on red:

```
m_g2antt.GetChart().SetNonworkingDays( 1 << ( EXG2ANTTLib::exSunday) | 1 << (
EXG2ANTTLib::exMonday ) );
m_g2antt.GetChart().SetNonworkingDaysColor( RGB(255,0,0,) );
```

where the #import <exg2antt.dll> must be called to insert definitions for types in the control's type library.

The following VB.NET sample marks Sunday and Monday days on red:

```
With AxG2antt1.Chart
   .NonworkingDays = 2 ^ (EXG2ANTTLib.WeekDayEnum.exSunday) Or 2 ^
(EXG2ANTTLib.WeekDayEnum.exMonday)
   .NonworkingDaysColor = ToUInt32(Color.Red)
End With
```

where the ToUInt32 function converts a Color expression to a OLE_DATE expression:

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
   Dim i As Long
   i = c.R
   i = i + 256 * c.G
   i = i + 256 * 256 * c.B
   ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample marks Sunday and Monday days on red:

```
axG2antt1.Chart.NonworkingDays = 1 <<
(Convert.ToInt32(EXG2ANTTLib.WeekDayEnum.exSunday)) | 1 <<
(Convert.ToInt32(EXG2ANTTLib.WeekDayEnum.exMonday));
axG2antt1.Chart.NonworkingDaysColor = ToUInt32( Color.Red );
```

where the ToUInt32 function converts a Color expression to a OLE_DATE expression:

```
private UInt32 ToUInt32(Color c)
{
   long i;
   i = c.R;
   i = i + 256 * c.G;
   i = i + 256 * 256 * c.B;
   return Convert.ToUInt32(i);
}
```

The following VFP sample sample marks Sunday and Monday days on red:

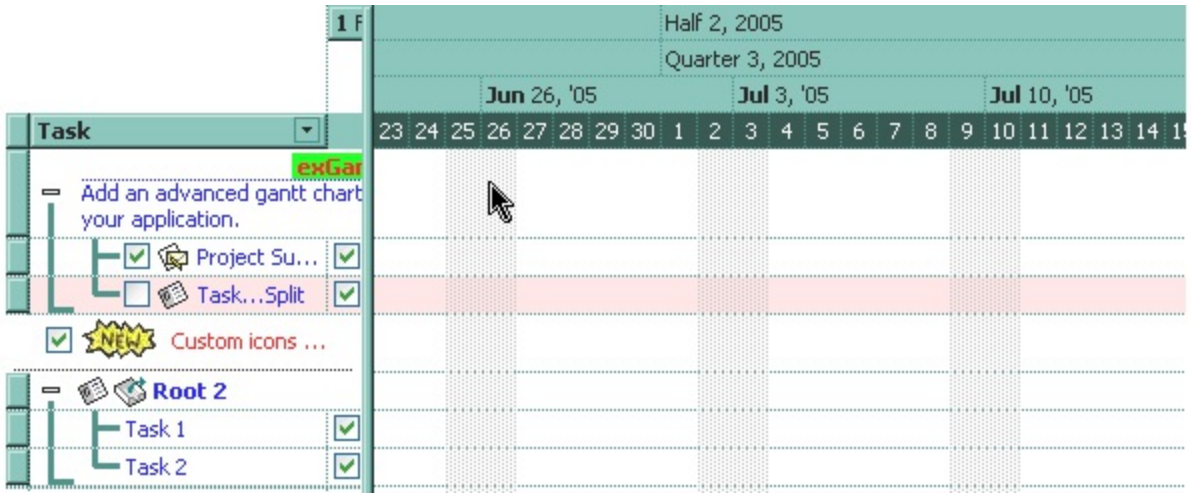
```
with thisform.G2antt1.Chart
   .NonworkingDays = 2 ^ 0 + 2 ^ 1
   .NonworkingDaysColor = RGB(255,0,0)
endwith
```

property Chart.NonworkingDaysPattern as PatternEnum

Retrieves or sets a value that indicates the pattern being used to fill non-working days.

Type	Description
PatternEnum	A PatternEnum expression that indicates the pattern to fill non working days.

Use the NonworkingDaysPattern property to specify the brush to fill the nonworking days area. Use the NonworkingDays property to specify the nonworking days. Use the AddNonworkingDate method to add custom dates as nonworking days. Use the NonworkingDaysPattern property to specify the pattern to fill non-working days. By default, the NonworkingDaysPattern property is exPatternDot. If the NonworkingDaysPattern property is exPatternEmpty, the non-working days are not highlighted. The NonworkingDaysColor property specifies the color being used to fill the non-working days. Use the MarkTodayColor property to specify the color to mark the today date. Use the DrawTickLines property to specify whether the grid lines between time units in the level are visible or hidden. Use the DrawGridLines property to specify whether the control draws the grid lines in the chart's area. Use the GridLineColor property to specify the color for grid lines. Use the DrawGridLines property to specify whether the control draws the grid lines in the items area. Use the DrawGridLines property to draw grid lines for a specified level.



property Chart.NonworkingHours as Long

Retrieves or sets a value that indicates the non-working hours, for each hour in a day a bit.

Type	Description
Long	A Long expression that indicates the non-working hours in a day.

by default, the NonworkingHours property is 0, that indicates all hours in a day are working hours. The non-working hours are shown using the NonworkingHoursPattern and the NonworkingHoursColor which defines the pattern and the color, when the base level of the chart displays hours, if the ShowNonworkingUnits property is True (by default). Use the ShowNonworkingUnits property to show or hide the non-working units as hours or days in your chart. Use the ItemNonworkingUnits property to specify different non-working zones for different items. Use ItemBar(exBarWorkingUnits) or ItemBar(exBarWorkingUnitsAsString) property to retrieve the working parts of the bar. Use ItemBar(exBarNonWorkingUnits) or ItemBar(exBarNonWorkingUnitsAsString) property to retrieve the non-working parts of the bar. Use the UnitWidthNonworking property to specify a different width for non-working units in the base level. Use the ShowNonworkingUnits property to hide the non-working units. The IsNonworkingDate property indicates whether the giving date-time is a working or non-working unit. You can use the ShowNonworkingHours property to show or hide the non-working hours while the NonworkingHours property is set, UnitScale is exDay (and Level.Count property is 1), exHour, exMinute and exSecond.

The control supports the following ways of specify the non-working parts for items:

- NonworkingDays and NonworkingHours properties indicate the nonworking parts of the chart being applied to all items with the exception of those that use the ItemNonworkingUnits property.
- AddNonworkingDate method adds custom dates as being nonworking date which is applied to all items with the exception of those that use the ItemNonworkingUnits property.
- ItemNonworkingUnits property defines the repetitive expression to specify the non-working parts in the item.
- ItemBar(exBarTreatAsNonworking) indicates whether the bar defines actually the non-working part of the item in addition to ItemNonworkingUnits property (which is required also)

You can select the non-working hours in the following table (In Internet Explorer, you have to allow running the script on this page).

24 Hour	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4
AM/PM	11PM	10PM	9PM	8PM	7PM	6PM	5PM	4PM	3PM	2PM	1PM	12AM	11AM	10AM	9AM	8AM	7AM	6AM	5AM	4AM

Value	8388608	4194304	2097152	1048576	524288	262144	131072	65536	32768	16384	8192	4096	2048	1024	512	256	128	64	32	16
Bit	☑	☑	☑	☑	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☑	☑	☑	☑

Click the Bit row for non-working hours and the value for property is: □, □(hexa), □(octal), □ (binary)

Every bit from the less significant bit, in the NonworkingHours property specifies whether the hour is a not-working or working hour. For instance, if you want to highlight that only 9AM is a not-working hour, you should set the 10th bit in the property on 1 (the hours starts from 0 to 23), and so the value for the NonworkingHours property is 512 (which binary representation is 1000000000). The hours in the property starts from 0AM for the first less significant bit, 1AM for the second bit, like in the following table.

24 Hour	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AM/PM Hour	11PM	10PM	9PM	8PM	7PM	6PM	5PM	4PM	3PM	2PM	1PM	12AM	11AM	10AM	9AM	8AM	7AM	6AM	5AM	4AM	3AM	2AM	1AM	0AM
Bit	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Value	8388608	4194304	2097152	1048576	524288	262144	131072	65536	32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

For instance, if you need the representation of non-working hours from 6PM to 8AM, you need to set on 1 each representative bit in the NonworkingHours property, or to add corresponding values in the last row in the table for each non-working hours, so in this case the NonworkingHours property is 16253183 or in binary 111110000000000011111111. For instance, if the NonworkingHours property is 0 or NonworkingHoursPattern is exPatternEmpty the not-working hours are not highlighted. Use the NonworkingDays property to specify non-working days. Use the Add("A:B") to add a bar that displays the bar A in the working area, and B in non-working areas.

The following function gets the value for the Chart.NonworkingHours property, giving start and end day shift hours:

```
Public Function getNonworkingHours(ByVal startTime As Date, ByVal endTime As
Date) As Long
   Dim nNonworkingHours As Long, d As Double, n As Long, i As Long, dHour As
Double, dSec As Double
   nNonworkingHours = 0
   dHour = 1 / 24
   dSec = dHour / 60 / 60
   d = 0
   n = 1
   For i = 1 To 24
      If (((d < startTime) And (Abs(d - startTime) > dSec)) Or ((d > endTime) And
(Abs(d - endTime) > dSec))) Then
         nNonworkingHours = nNonworkingHours + n
```

```
        End If
        n = n * 2
        d = d + dHour
    Next
    getNonworkingHours = nNonworkingHours
End Function
```

The getNonworkingHours function, takes the start / end time (values between 0 and 1), and returns the value to specify for the Chart.NonworkingHours property. For instance, the G2antt1.Chart.NonworkingHours = getNonworkingHours(#6:00:00 AM#, #6:00:00 PM#) specifies working hours between 06:00 AM and 06:00 PM (inclusive).

property Chart.NonworkingHoursColor as Color

Retrieves or sets a value that indicates the color to fill the non-working hours.

Type	Description
Color	A Color expression that indicates the color to fill the non-working hours.

Use the NonworkingHoursColor property to specify the color being used by the NonworkingHoursPattern property. Use the NonworkingHours property to specify the nonworking hours in a day. Use the NonworkingHoursPattern property to specify the pattern to fill the non-working hours. For instance, if the NonworkingHours property is 0 or NonworkingHoursPattern is exPatternEmpty the not-working hours are not highlighted.

property Chart.NonworkingHoursPattern as PatternEnum

Retrieves or sets a value that indicates the pattern being used to fill non-working hours.

Type	Description
PatternEnum	A PatternEnum expression that indicates the pattern to fill non working hours in a day.

Use the NonworkingHoursPattern property to specify the brush to fill the nonworking hours area. Use the NonworkingHoursColor property to specify the color being used by the NonworkingHoursPattern property. Use the NonworkingHours property to specify the nonworking hours in a day. For instance, if the NonworkingHours property is 0 or NonworkingHoursPattern is exPatternEmpty the not-working hours are not highlighted.

property Chart.NoteFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Note

Retrieves the note from the point.

Type	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in client coordinates.
Note	A Note object from the cursor. Nothing or Null object of no note from the point.

The NoteFromPoint property retrieves the note from the cursor or specified point. **If the X parameter is -1 and Y parameter is -1 the NoteFromPoint property determines the Note object from the cursor.** Use the ItemFromPoint property to get the cell/item from the cursor. Use the ColumnFromPoint property to retrieve the column from cursor. Use the LevelFromPoint property to retrieve the index of the level from the cursor. The DateFromPoint property determines the DATE from the cursor. Use the LinkFromPoint property to get the link from the point. Use the BarFromPoint property to get the bar from the point.

The following VB sample displays the ID of the Note from the cursor:

```
Private Sub G2antt1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
  With G2antt1.Chart
    Dim n As EXG2ANTTLibCtl.Note
    Set n = .NoteFromPoint(-1, -1)
    If (Not n Is Nothing) Then
      Debug.Print n.ID
    End If
  End With
End Sub
```

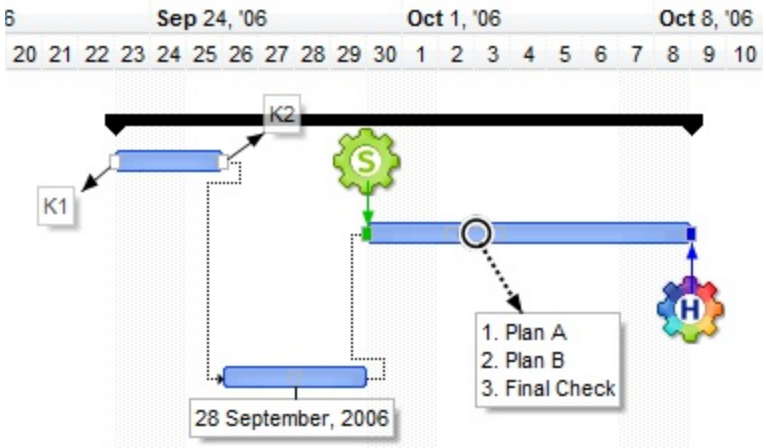
property Chart.Notes as Notes

Retrieves the Notes collection.

Type	Description
Notes	A Notes collection being accessed.

Use the Notes property to access the Notes collection of the Chart object. Use the Add method to add new boxes/notes related to a BAR or to a DATE. The NoteFromPoint property retrieves the note/box from the cursor. A note can be associated with a DATE in the chart or can be associated to a BAR in the chart. A note is a box that moves together with the related object. For instance, if a note is associated with the starting point of the bar (start date), and the user resizes the bar in the left side (so it changes the starting point of the bar), the related box/note is moved relatively too. The Notes object can be accessed through the Notes property of the Chart object. A Note or a Box can display HTML captions, images, icons, borders, links, and it is fully customizable. The note is composed by two parts, the starting part and end part, that can be linked together. The start part is related to the DATE or to the BAR, while the end part is related to the start part of the note, such us if the start part is moved, the end part is relatively moved. The user can move the end part around the start part, while the start part remains unchanged, or can move so the entire box is moved relatively to the object (DATE or BAR). Use the Items.ItemBar(exBarCaption) property to assign a HTML text, icons, pictures to a bar. Use the Items.ItemBar(exBarExtraCaption) property to add or associate extra captions to a bar.

The following screen shot shows notes//boxes associated to bars:



property Chart.OverlaidOnMoving as Boolean

Specifies whether the overlaid bars are re-arranged while the user moves or resizes at runtime a bar.

Type	Description
Boolean	A boolean expression that specifies whether the overlaid bars is applied while user moves or resizes a bar at runtime.

By default, the OverlaidOnMoving property is True. The OverlaidType property specifies whether the bars of the same type are re-arranged on the item. The performance of the control could be improved while using the overlaid feature, using the OverlaidOnMoving property on False. This property has effect only if the control display overlaying bars. The IntersectBars property determines if two bars intersects if returns 0. The ItemBar(exBarIntersectWith) property retrieves a collection of bars that interest with the current bar.

property Chart.OverviewBackColor as Color

Specifies the background color of the chart's overview.

Type	Description
Color	A Color expression that indicates the background color of the chart's overview.

Use the OverviewBackColor property to change the background color of the overview's overview. The OverviewVisible property specifies whether the overview's overview layout is visible or hidden. Use the BackColor property to change the background color for the chart area. Use the OverviewSelBackColor property to change the visual appearance of the selection in the overview area. The OverviewShowMarkTimeZones property shows the marked time-zones on the control's overview map. The OverviewSelTransparent property indicates the transparency to show the selection in the items or scale-zoom part of the overview map.

The following VB sample changes the overview's background color:

```
With G2antt1.Chart
   .OverviewBackColor = RGB(&H80, &H80, &H80)
End With
```

The following C++ sample changes the overview's background color:

```
m_g2antt.GetChart().SetOverviewBackColor( RGB(0x80,0x80,0x80) );
```

The following VB.NET sample changes the overview's background color:

```
With AxG2antt1.Chart
   .OverviewBackColor = ToUInt32(Color.FromArgb(&H80, &H80, &H80))
End With
```

where the ToUInt32 function converts a Color expression to an OLE_COLOR type:

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
   Dim i As Long
   i = c.R
   i = i + 256 * c.G
   i = i + 256 * 256 * c.B
   ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample changes the overview's background color:

```
axG2antt1.Chart.OverviewBackColor = ToUInt32(Color.FromArgb(0x80, 0x80, 0x80));
```

where the ToUInt32 function converts a Color expression to an OLE_COLOR type:

```
private UInt32 ToUInt32(Color c)
{
   long i;
   i = c.R;
   i = i + 256 * c.G;
   i = i + 256 * 256 * c.B;
   return Convert.ToUInt32(i);
}
```

The following VFP sample changes the overview's background color:

```
With thisform.G2antt1.Chart
   .OverviewBackColor = RGB(128, 128, 128)
EndWith
```

property Chart.OverviewHeight as Long

Indicates the height of the chart's overview.

Type	Description
Long	A long expression that indicates the height of the chart's overview area.

By default, the OverviewHeight property is 24 pixels. If the OverviewHeight property is 0, or the OverviewVisible property is False, the chart's overview area is hidden. The OverviewBackColor property specifies the background color for the overview area. Use the OverviewSelBackColor property to change the visual appearance of the selection in the overview area. The OverviewToolTip property specifies the format of the tooltip being displayed when the cursor hovers the overview area. The OverviewLevelLines property indicates the index of the level that displays the grid lines in the overview area.

property Chart.OverviewLevelLines as Long

Indicates the index of the level that displays the grid line in the chart's overview.

Type	Description
Long	A long expression that indicates the index of the level that displays the grid lines in the chart's overview area.

By default, the OverviewLevelLines property is -1. If the OverviewLevelLines property is -1, or indicates a non-existent level, no grid lines are shown in the chart's overview area. Use the OverviewLevelLines property to show grid lines in the chart's overview area. The OverviewVisible property shows or hides the chart's overview area. Use the Level property to access a level using its index. The LevelCount property indicates the number of levels being displayed in the control's header. Use the DrawGridLines property to specify the color of the grid lines in the overview area.

property Chart.OverviewSelBackColor as Color

Specifies the selection color of the chart's overview.

Type	Description
Color	A color expression that defines the selected items background color. The last 7 bits in the high significant byte of the color indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

Use the OverviewSelBackColor property specifies background color or the visual appearance for the selection in the chart's overview. The user can resize the chart by drag and drop the left or right resize-margins of the overview-selection, while the Background(exOverviewSelResize) property is not zero. The AllowResizeChart property specifies whether the user can enlarge (zoom-in,zoom-out) or resize the chart using the control's header, middle mouse button. The OverviewBackColor property specifies the background color for the overview area. The OverviewVisible property specifies whether the chart's overview layout is visible or hidden. Use the OverviewHeight property to specify the height in pixels, of the overview area. The OverviewToolTip property specifies the format of the tooltip being displayed when the cursor hovers the overview area. The OverviewLevelLines property indicates the index of the level that displays the grid lines in the overview area. The OverviewSelTransparent property indicates the transparency to show the selection in the items or scale-zoom part of the overview map. The Background(exOverviewSelUnit) property specifies the background color / visual appearance to display the selected unit within the control's overview.

property Chart.OverviewSelTransparent(Items as Boolean) as Long

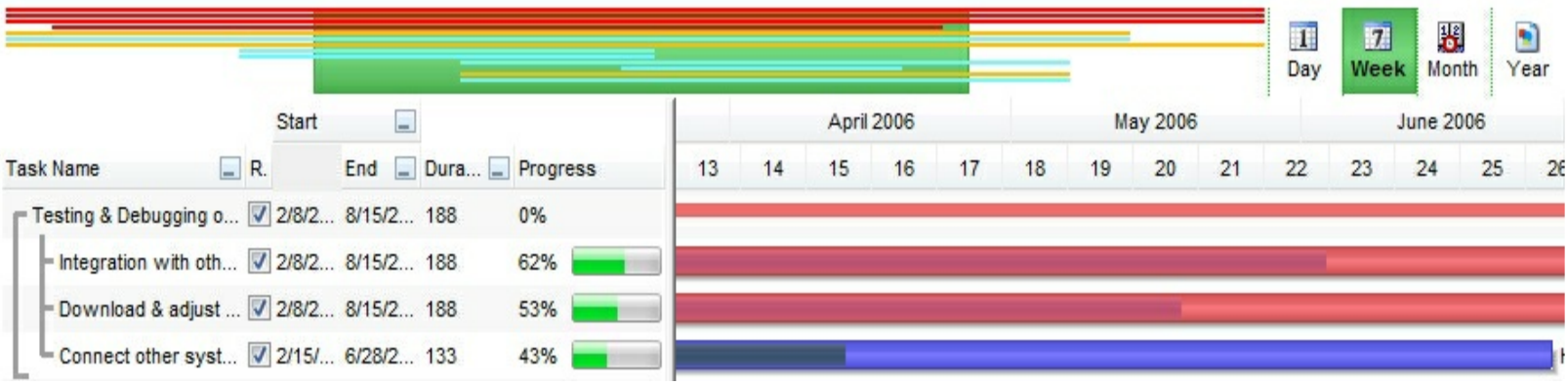
Specifies the percent of the transparency to display the selection in the overview parts of the control.

Type	Description
Items as Boolean	A Boolean expression that specifies whether the transparency for selection is changed in the items or scale-zooming part.
Long	A long expression that indicates the transparency to show the selection in the overview part of the control. The value should be from 0 to 100, as 0 opaque, 50 - semi-transparent , 100 fully transparent (not shown)

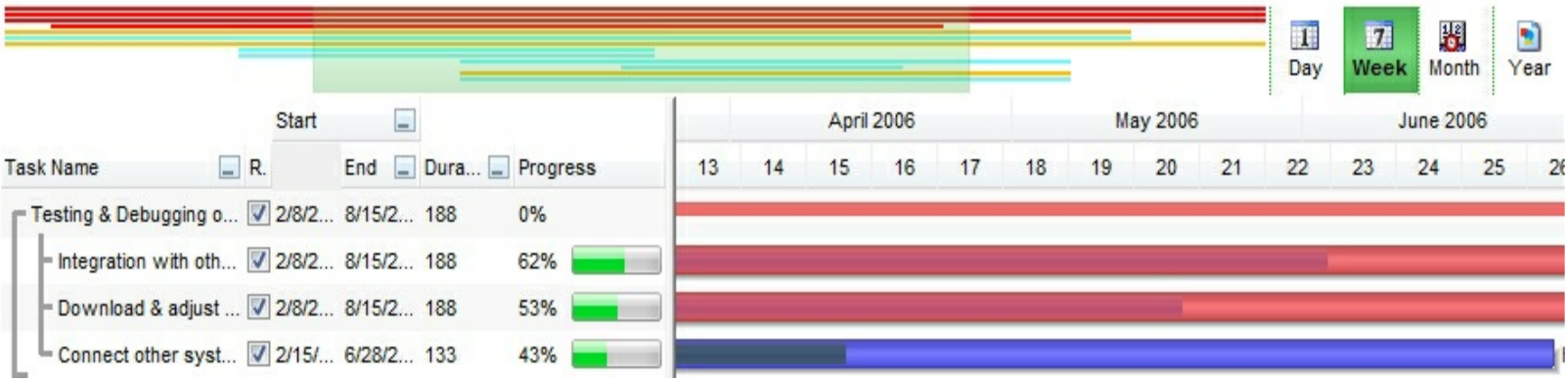
By default, the OverviewSelTransparent property is 0, which means that the selection is opaque. The OverviewVisible property specifies whether the chart's overview map is visible or hidden. Use the OverviewSelBackColor property to change the visual appearance of the selection in the overview area. The OverviewBackColor property specifies the background color for the overview area. The overview-map part can display the scale-zooming scale if the AllowOverviewZoom property is not exDisableZoom.

The following screen shots shows the overview-map part of the control with different values for the OverviewSelTransparent property:

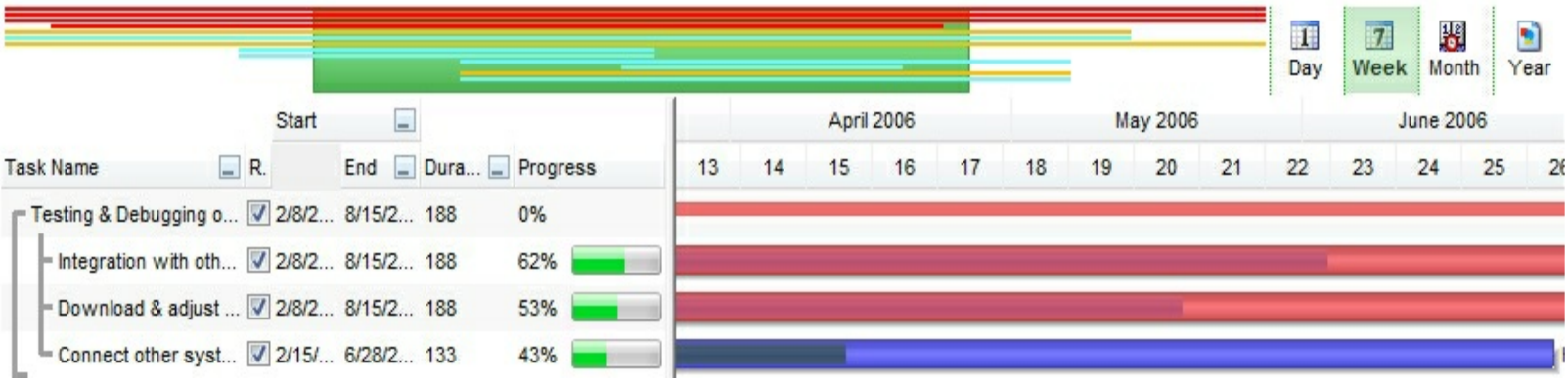
OverviewSelTransparent(True) = 0, OverviewSelTransparent(False) = 0



OverviewSelTransparent(True) = **70**, OverviewSelTransparent(False) = 0



OverviewSelTransparent(True) = 0, OverviewSelTransparent(False) = **70**



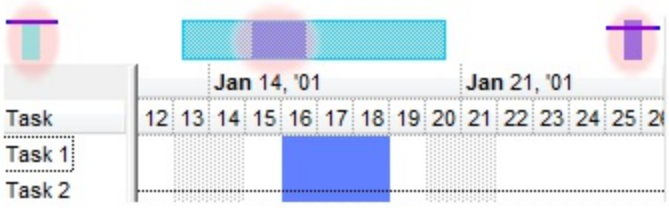
property Chart.OverviewShowMarkTimeZones as Boolean

Specifies whether the chart's overview shows the marked time-zones.

Type	Description
Boolean	A Boolean expression that specifies whether the chart's time- zones are shown in the control's overview part.

By default, the OverviewShowMarkTimeZones property is False. The OverviewVisible property indicates whether the control's overview part is shown or hidden. The MarkTimeZone method highlights different time-zones. Use the OverviewShowMarkTimeZones property to show the marked time-zones on the control's overview map. The color for the time-zone is being indicated by the Color parameter of the MarkTimeZone method. The time-zone is being shown only if the start and end dates points to a valid date, the Color is different then chart's background color (BackColor property), and the pattern for the time-zone is not empty. If the time-zone displays HTML captions they do not appear on the overview part of the control. A time-zone may be shown in the overview part of the control only if the Color parameter is the same as BackColor property, and the OverviewBackColor property is different than the chart's BackColor property.The OverviewShowSelectDates property specifies whether the selected dates are shown in the overview part of the control.

The following screen shot shows time-zones in the overview map:



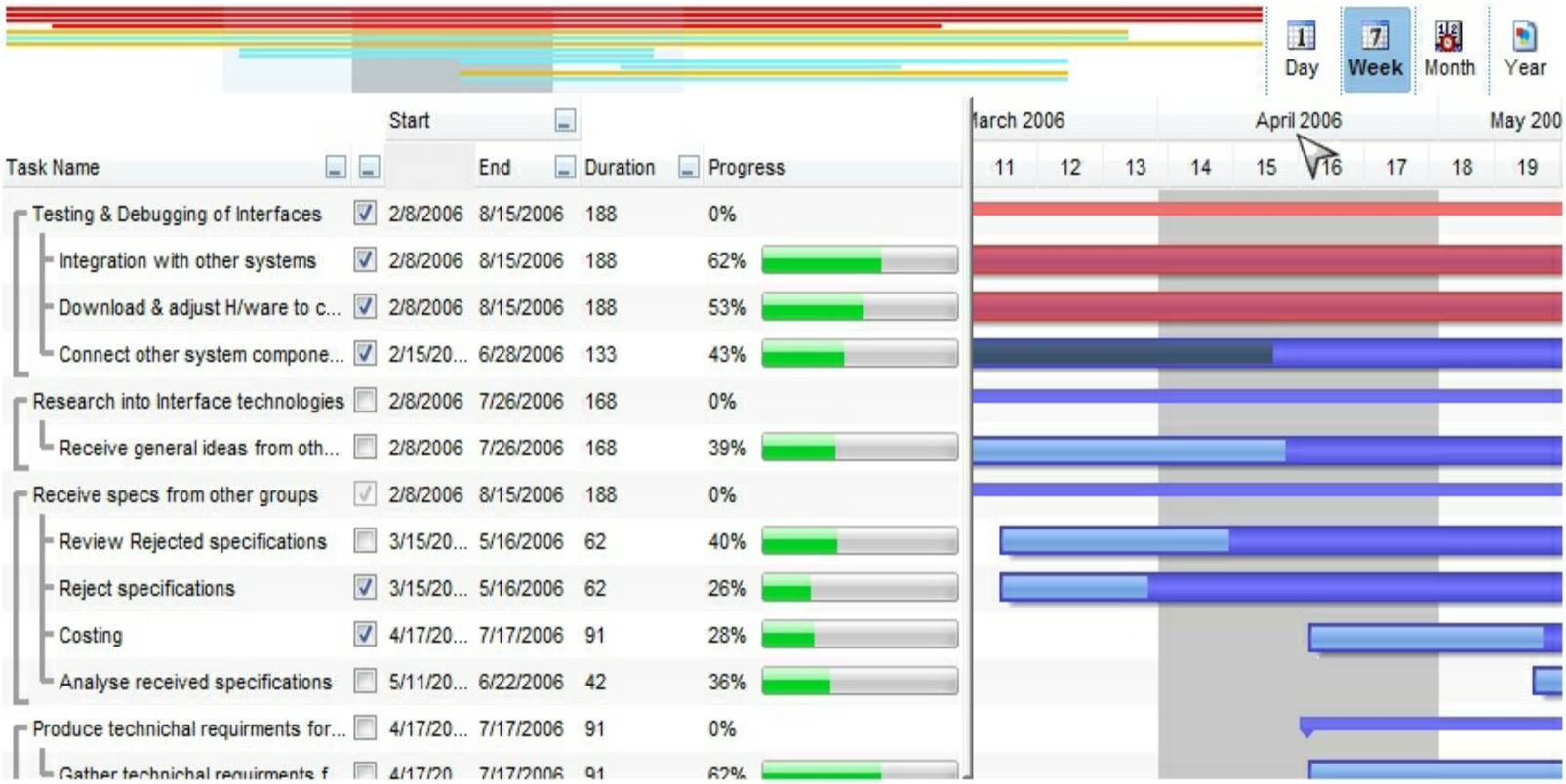
property Chart.OverviewShowSelectDates as Boolean

Specifies whether the chart's overview shows the selected dates.

Type	Description
Boolean	A Boolean expression that specifies whether the selected dates in the chart are shown in the overview-map part of the control.

By default, the OverviewShowSelectDates property is False. The OverviewShowSelectDates property specifies whether the selected dates are shown in the overview part of the control. The AllowSelectDate property indicates whether the user can select dates on the chart by clicking the its header. The OverviewVisible property specifies whether the chart's overview map is visible or hidden. Use the OverviewShowMarkTimeZones property to specify whether zones marked by the MarkTimeZone method are shown in the overview part of the control.

The following screen shot shows the selected April 2006 zone on the overview map:



property Chart.OverviewToolTip as String

Retrieves or sets a value that indicates the format of the tooltip being shown while the cursor hovers the chart's overview area.

Type	Description
String	A String expression that specifies the format of the tooltip being displayed when the cursor hovers the chart's overview area.

By default, the OverviewToolTip property is "<%ddd%> <%m%>/<%d%>/<%yyyy%> ". The OverviewVisible property specifies whether the chart's overview layout is visible or hidden. Use the OverviewHeight property to specify the height in pixels, of the overview area. Use the ToolTip property to specify the format of the toolip being displayed when the user scrolls the chart's content. The Tooltip(0, -5, , , , ,) event occurs once the overview's tooltip (Chart.OverviewToolTip) is about to be shown (-5 if the mouse pointer hovers the overview section of the chart)

The OverviewToolTip property supports the following built-in tags:

- <%**d**%> - Day of the month in one or two numeric digits, as needed (1 to 31).
- <%**dd**%> - Day of the month in two numeric digits (01 to 31).
- <%**d1**%> - First letter of the weekday (S to S). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**d2**%> - First two letters of the weekday (Su to Sa). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**d3**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**ddd**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays property to specify the name of the days in the week). You can use the <%**loc_ddd**%> that indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%**loc_ddd**%> - Indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%**dddd**%> - Full name of the weekday (Sunday to Saturday). (Use the WeekDays property to specify the name of the days in the week). You can use the <%**loc_dddd**%> that indicates day of week as its full name using the current user regional and language settings.

- <%**loc_dddd**%> - Indicates day of week as its full name using the current user regional and language settings.
- <%**i**%> - Displays the number instead the date. For instance, you can display numbers as 1000, 1001, 1002, 1003, instead dates. (the valid range is from -647,434 to 2,958,465)
- <%**w**%> - Day of the week (1 to 7).
- <%**ww**%> - Week of the year (1 to 53).
- <%**m**%> - Month of the year in one or two numeric digits, as needed (1 to 12).
- <%**mr**%> - Month of the year in Roman numerals, as needed (I to XII).
- <%**mm**%> - Month of the year in two numeric digits (01 to 12).
- <%**m1**%> - First letter of the month (J to D). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**m2**%> - First two letters of the month (Ja to De). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**m3**%> - First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**mmm**%> - First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmm**%> that indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%**loc_mmm**%> - Indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%**mmmm**%> - Full name of the month (January to December). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmmm**%> that indicates month as its full name using the current user regional and language settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user regional and language settings.
- <%**q**%> - Date displayed as the quarter of the year (1 to 4).
- <%**y**%> - Number of the day of the year (1 to 366).
- <%**yy**%> - Last two digits of the year (01 to 99).
- <%**yyyy**%> - Full year (0100 to 9999).
- <%**hy**%> - Date displayed as the half of the year (1 to 2).
- <%**loc_gg**%> - Indicates period/era using the current user regional and language settings.
- <%**loc_sdate**%> - Indicates the date in the short format using the current user regional and language settings.

- <%**loc_ldate**%> - Indicates the date in the long format using the current user regional and language settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user regional and language settings (/).
- <%**h**%> - Hour in one or two digits, as needed (0 to 23).
- <%**hh**%> - Hour in two digits (00 to 23).
- <%**h12**%> - Hour in 12-hour time format, in one or two digits - [0(12),11]
- <%**hh12**%> - hour in 12-hour time format, in two digits - [00(12),11]
- <%**n**%> - Minute in one or two digits, as needed (0 to 59).
- <%**nn**%> - Minute in two digits (00 to 59).
- <%**s**%> - Second in one or two digits, as needed (0 to 59).
- <%**ss**%> - Second in two digits (00 to 59).
- <%**AM/PM**%> - Twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate. (Use the AMPM property to specify the name of the AM and PM indicators). You can use the <%**loc_AM/PM**%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%**loc_A/P**%> that indicates the one character time marker such as A or P using the current user regional and language settings
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user regional and language settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user regional and language settings.
- <%**loc_time**%> - Indicates the time using the current user regional and language settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%**loc_tsep**%> - indicates the time separator using the current user regional and language settings (:)
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

The following tags are displayed based on the user's Regional and Language Options:

- <%**loc_sdate**%> - Indicates the date in the short format using the current user settings.
- <%**loc_ldate**%> - Indicates the date in the long format using the current user settings.
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**loc_ddd**%> - Indicates day of week as a three-letters abbreviation using the current user settings.
- <%**loc_dddd**%> - Indicates day of week as its full name using the current user settings.
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**loc_mmm**%> - Indicates month as a three-letters abbreviation using the current user settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user settings.
- <%**loc_gg**%> - Indicates period/era using the current user settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user settings.
- <%**loc_time**%> - Indicates the time using the current user settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user settings.
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user settings.
- <%**loc_tsep**%> - Indicates the time separator using the current user settings
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

property Chart.OverviewVisible as OverviewVisibleEnum

Specifies whether the chart's overview layout is visible or hidden.

Type	Description
OverviewVisibleEnum	An OverviewVisibleEnum expression that indicates whether the chart's overview area is visible or hidden.

By default, the OverviewVisible property is exOverviewHidden (0), so the chart's overview portion is not visible. The overview layout/map It is a view that is displayed at the top of the control and shows the whole timeline, with all objects within its view (a high-level view). It displays a 'select' box (the light blue box) that the user can drag to any location within the overview and then that area of the chart is shown at normal scale within the chart view. Use the OverviewHeight property to specify the height in pixels, of the overview area. The OverviewBackColor property specifies the background color for the overview area. Use the OverviewSelBackColor property to change the visual appearance of the selection in the overview area. The OverviewToolTip property specifies the format of the tooltip being displayed when the cursor hovers the overview area. The OverviewLevelLines property indicates the index of the level that displays the grid lines in the overview area. If the Label property is empty, the unit is not displayed in the zooming scale, if the AllowOverviewZoom property is not exDisableZoom. The OverviewZoomCaption property indicates the caption being displayed in each zooming unit. The OverviewShowMarkTimeZones property shows the marked time-zones on the control's overview map.

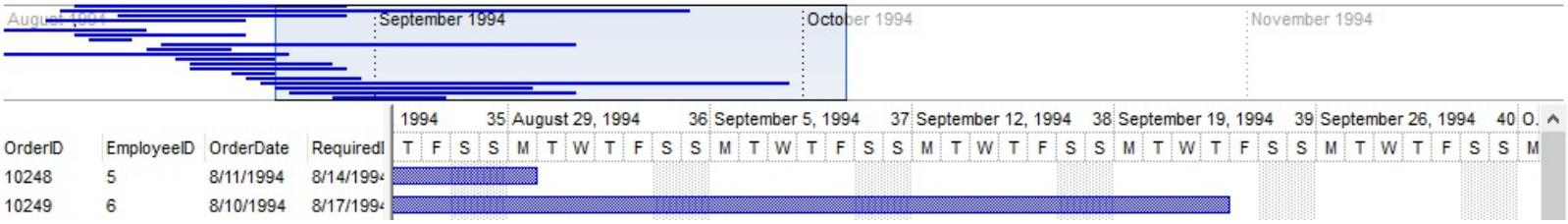
The color to specify the bar in the overview area is determined as follows:

- If ItemBar(exBarOverviewColor) property is not 0, the exBarOverviewColor indicates the color to show the bar in the overview area, else
- If OverviewColor property is not 0, the OverviewColor property indicates the color to show the bar in the overview area, else
- If the ItemBar(exBarColor) is not 0, the exBarColor indicates the color to show the bar in the overview area, else
- The Color property of the Bar indicates the color to show the bar in the overview part of the control.

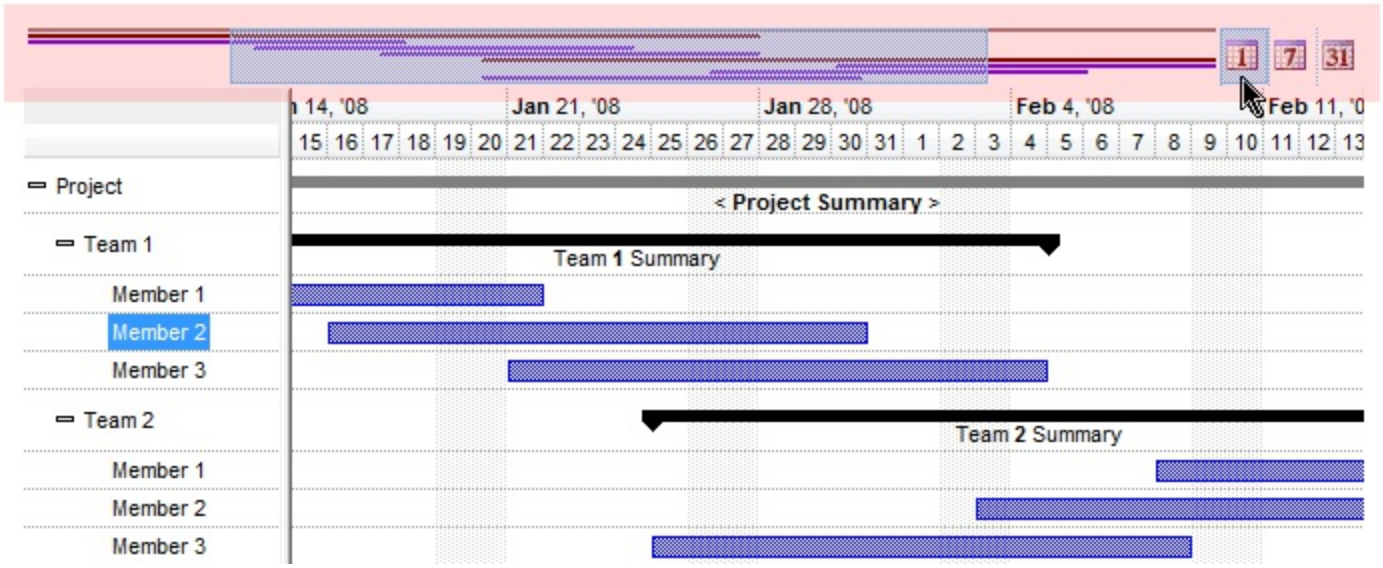
(The bar is represented into the control's overview only if its determined color is not -1)

The color to specify the time-zone in the overview area is indicates by the Color parameter of the MarkTimeZone method. The MarkTimeZone method may be used to highlights different parts of the chart by specifying the range of dates.

The following screen shot shows the overview part of the control (the top zone):



The following screen shot shows the overview part of the control (the red zone):



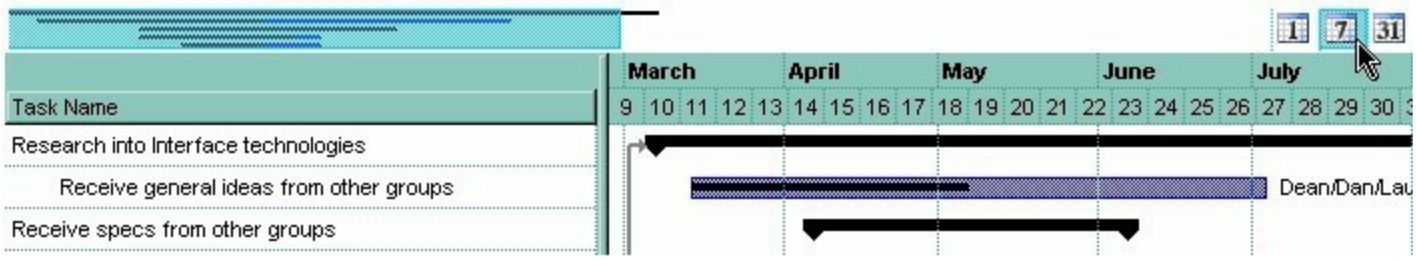
property Chart.OverviewZoomCaption as String

Specifies the captions for each zooming unit.

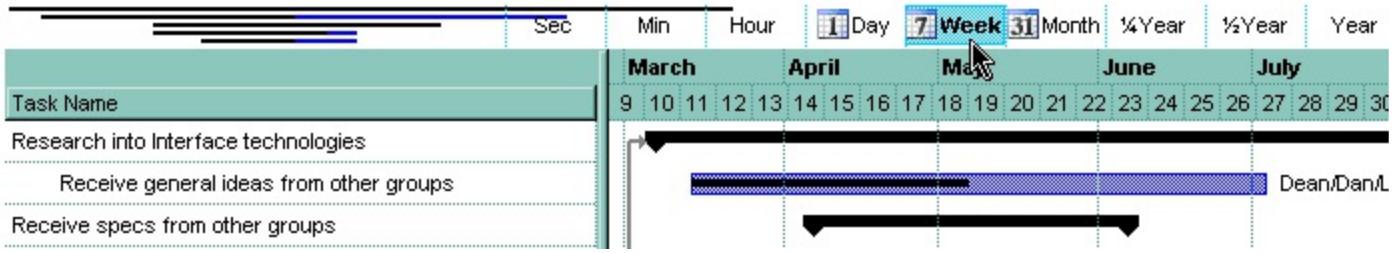
Type	Description
String	A string expression that defines a list of captions (one for each unit) being displayed in the zoom scale, separated by | character. The list should contain a caption for each unit, from the exYear to exSecond. For instance, if you want to show nothing for exHalfYear zooming unit, the OverviewZoomCaption should be: "Year||źYear...", and so on

By default, the OverviewZoomCaption property is "**Year|˝Year|źYear|Month|Third|Week|Day|Hour|Min|Sec**". The OverviewZoomCaption property supports HTML tags, and so the zooming units may display icons or/and pictures using the <img> tag. The OverviewZoomUnit property indicates the width in pixels of the zooming unit. The zooming scale displays the list of visible units. A visible unit is an unit whose Label property is not empty. So, the Label property indicates the zooming units in the zoom scale. Use the OverviewVisible property to show or hide the control's overview area.

The following picture shows the zooming scale on the overview area [exAlwaysZoom] (you can click the 1, 7 or 31, and the chart is scaled to days, weeks or moths):



*The following picture shows the control when the user **right** clicks the overview area (as the chart displays weeks) [exZoomOnRClick]:*



For instance, in the OverviewZoomCaption property is "Year|˝Year|źYear|<img>3</img>Month|Third|<img>2</img>Week|<img>1</img>Day|Hour|Min|Sec". The

Day, Month and Week units displays an icon too. Use the Images method to load a list of icons to your control. Use the HTMLPicture property to use custom sized pictures to your HTML captions.

property Chart.OverviewZoomUnit as Long

Indicates the width in pixels of the zooming unit in the overview.

Type	Description
Long	A long expression that indicates the width in pixels of the zooming unit.

By default, the OverviewZoomUnit property is 42 pixels. The OverviewZoomUnit property indicates the width in pixels of the zooming unit. Use the OverviewVisible property to show or hide the control's overview area. Use the AllowOverviewZoom property to show or hide the zooming scale on the overview area. The zooming scale displays the list of visible units. A visible unit is an unit whose Label property is not empty. So, the Label property indicates the zooming units in the zoom scale. The OverviewZoomCaption property indicates the caption being displayed in each zooming unit. The LabelToolTip retrieves or sets a value that indicates the predefined format of the level's tooltip for a specified unit.

The zooming scale may be displayed on the overview area only if:

- AllowOverviewZoom property is not exDisableZoom
- OverviewVisible property is True
- OverviewHeight property is greater than 0
- there are at least two visible units, that has the Label property not empty.

property Chart.PaneWidth(Right as Boolean) as Long

Specifies the width for the left or side pane.

Type	Description
Right as Boolean	A Boolean expression that indicates whether the left (items area) or right (chart area) area is changed.
Long	A Long expression that indicates the width of the pane.

Use the PaneWidth property to specify the width of the control (items area) or chart area. Use the AddBar method to add bars to the item. The bars are always shown in the chart area. Use the HeaderVisible property to show or hide the control's header. Use the SortBarVisible property to specify whether the control's sort bar is visible or it is hidden. Use the LevelCount property to specify the number of levels being displayed in the chart's header. Use the Level property to access the level in the chart area. Use the BackColor property to specify the items's background color. Use the ForeColor property to specify the item's foreground color. Use the BackColor property to specify the chart's background color. Use the ForeColor property to specify the chart's foreground color. The HistogramBoundsChanged event notifies your application when the location and the size of the chart's histogram is changed, so you can use it to add your legend for the histogram in a panel component. Use the OnResizeControl(exResizeList) or OnResizeControl(exResizeChart) property to specify whether the left or right part gets resized when the control gets resized. The controls vertical splitter is hidden if the OnControlResize property is exResizeChart + exDisableSplitter (129) and the PaneWidth(False) property is 0.

Use the OnResizeControl property to allow:

- resizing the list area when the control is resized (by default)
- resizing the chart area when the control is resized.

You can also use the OnResizeControl property to prevent:

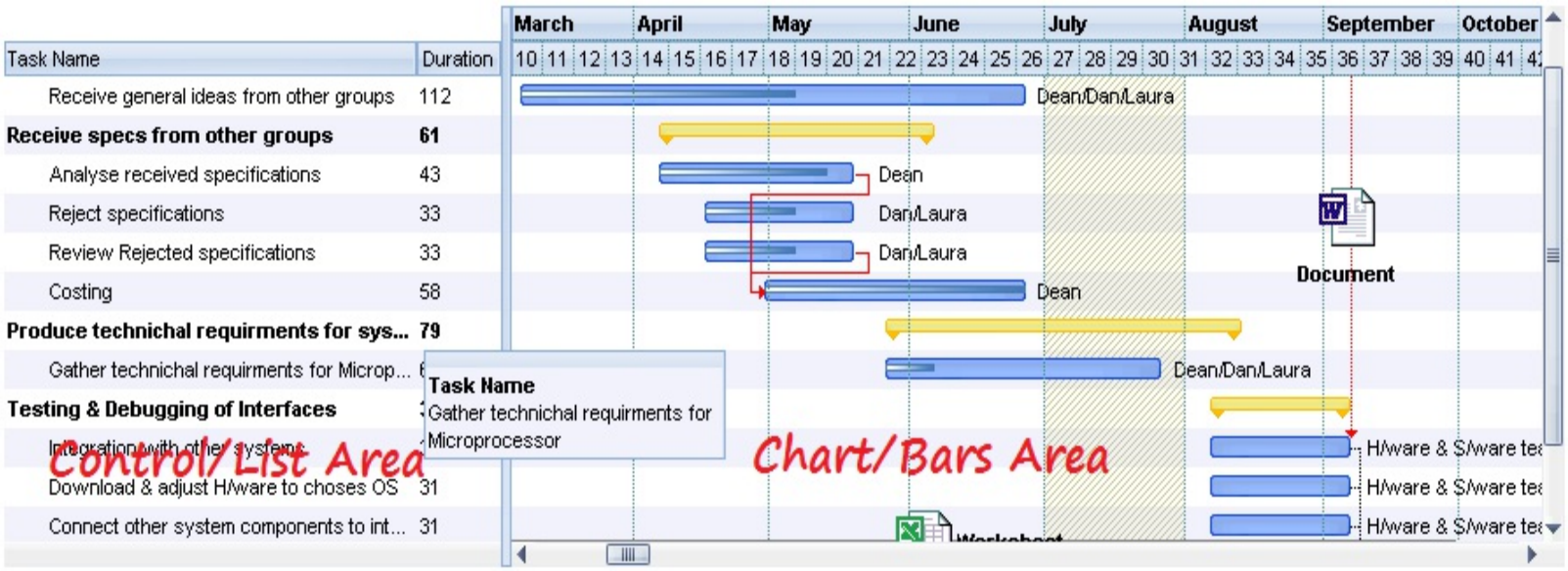
- resizing the list/chart using the control's splitter.
- resizing the chart's histogram.

The following VFP sample changes the width of the control's area:

```
with thisform.G2antt1.Chart
   .PaneWidth(0) = 256
endwith
```

The following VFP sample changes the width of the chart's area:

```
with thisform.G2antt1.Chart
    .PaneWidth(1) = 256
endwith
```

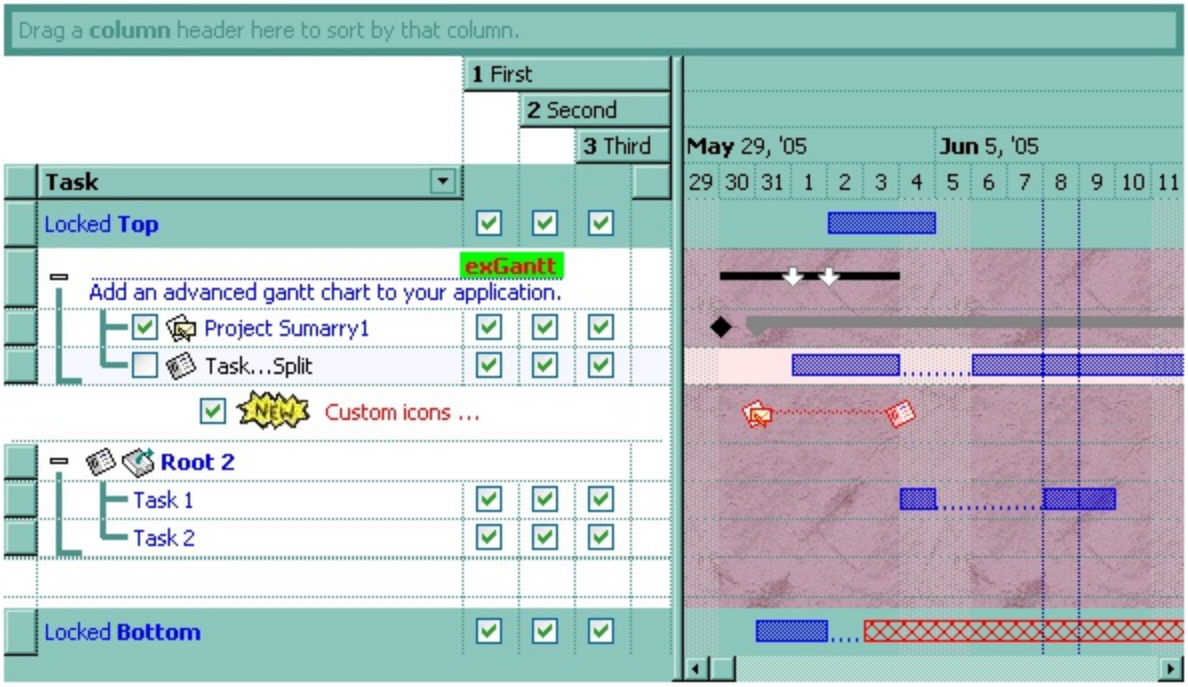


property Chart.Picture as IPictureDisp

Retrieves or sets a graphic to be displayed in the chart.

Type	Description
IPictureDisp	A Picture object that's displayed on the control's background.

By default, the chart has no picture associated. The control uses the PictureDisplay property to determine how the picture is displayed on the chart's background. Use the PictureLevelHeader property to specify the picture on the control's levels header bar. Use the CellPicture property to assign a picture to a cell. Use the BackColor property to specify the control's background color. Use the Picture property to assign a picture on the control's background.



property Chart.PictureDisplay as PictureDisplayEnum

Retrieves or sets a value that indicates the way how the graphic is displayed on the chart's background

Type	Description
PictureDisplayEnum	A PictureDisplayEnum expression that indicates the way how the picture is displayed in the chart's area.

By default, the PictureDisplay property is exTile. The PictureDisplay property specifies how the Picture is displayed on the chart's background. If the chart has no picture associated the PictureDisplay property has no effect. Use the CellPicture property to assign a picture to a cell. Use the BackColor property to specify the control's background color. Use the BackColor property to specify the chart's background color.

method Chart.Redo ()

Redoes the next action in the chart's Redo queue.

Type	Description

Call the Redo method to Redo the last chart operation. The Redo method have effect only if the AllowUndoRedo property is True. The CTRL+Y redoes the next action in the chart's Redo queue, while the CTRL+Z performs the last undo operation. Call the Undo method to Undo the last chart operation. The CanUndo property retrieves a value that indicates whether the chart may perform the last Undo operation. The CanRedo property retrieves a value that specifies whether the chart can execute the next operation in the chart's Redo queue. The ChartStartChanging(exUndo/exRedo) / ChartEndChanging(exUndo/exRedo) event notifies your application whenever an Undo/Redo operation is performed.

The records of the Undo/Redo queue may contain actions in the following format:

- "**AddBar**;ITEMINDEX;KEY", indicates that a new bar has been created
- "**RemoveBar**;ITEMINDEX;KEY", indicates that a bar has been removed
- "**MoveBar**;ITEMINDEX;KEY", indicates that a bar has been moved or resized
- "**PercentChange**;ITEMINDEX;KEY", indicates that the bar's percent has been changed
- "**UpdateBar**;ITEMINDEX;KEY", indicates that one or more properties of the bar has been updated (ItemBar property, this operation can be added only using the StartUpdateBar / EndUpdateBar methods)
- "**ParentChangeBar**;ITEMINDEX;KEY", indicates that the bar's parent has been changed
- "**GroupBars**;ITEMINDEXA;KEYA;STARTA;ITEMINDEXB;KEYB;STARTB", specifies that two bars has been grouped
- "**UngroupBars**;ITEMINDEXA;KEYA;STARTA;ITEMINDEXB;KEYB;STARTB", specifies that two bars has been ungrouped
- "**DefineSummaryBars**;SUMMARYITEMINDEX;SUMMARYKEY;ITEMINDEX;KEY", indicates that a bar has been defined as a child of a summary bar
- "**UndefineSummaryBars**;SUMMARYITEMINDEX;SUMMARYKEY;ITEMINDEX;KEY", indicates that a bar has been removed from the summary bar's children
- "**AddLink**;KEY", indicates that a new link has been created
- "**RemoveLink**;KEY", indicates that a link has been removed
- "**UpdateLink**;KEY", specifies that one of more properties of the link has been updated (Link property, this operation can be added only using the StartUpdateLink / EndUpdateLink methods)

The records of the Undo/Redo queue may contain actions in the following format (available starting from 23.0):

- "**AddItem**;ITEMINDEX", indicates that a new item has been created
- "**RemoveItem**;ITEMINDEX", indicates that an item has been removed
- "**ChangeItemPos**;ITEMINDEX", indicates that an item changes its position or / and parent
- "**ChangeCellValue**;ITEMINDEX;CELLINDEX", indicates that the cell's value has been changed
- "**ChangeCellState**;ITEMINDEX;CELLINDEX", indicates that the cell's state has been changed

Also, the Undo/Redo queue may include:

- "**StartBlock**", specifies that a block of operations begins
- "**EndBlock**", specifies that a block of operations ends

The RedoListAction property lists the Redo actions that can be performed in the chart. Use the RedoRemoveAction method to remove the first action from the redo queue.

property Chart.RedoListAction ([Action as Variant], [Count as Variant]) as String

Lists the Redo actions that can be performed in the chart.

Type	Description
Action as Variant	[optional] A long expression that specifies the action being listed. If missing or -1, all actions are listed. The Action parameter can be one of the following: • exChartUndoRedoAddBar(0) ~ "**AddBar**;ITEMINDEX;KEY", indicates that a new bar has been created • exChartUndoRedoRemoveBar(1) ~ "**RemoveBar**;ITEMINDEX;KEY", indicates that a bar has been removed • exChartUndoRedoMoveBar(2) ~ "**MoveBar**;ITEMINDEX;KEY", indicates that a bar has been moved or resized • exChartUndoRedoPercentChange(3) ~ "**PercentChange**;ITEMINDEX;KEY", indicates that the bar's percent has been changed • exChartUndoRedoUpdateBar(4) ~ "**UpdateBar**;ITEMINDEX;KEY", indicates that one or more properties of the bar has been updated (ItemBar property, this operation can be added only using the StartUpdateBar / EndUpdateBar methods) • exChartUndoRedoParentChangeBar(5) ~ "**ParentChangeBar**;ITEMINDEX;KEY", indicates that the bar's parent has been changed • exChartUndoRedoGroupBars(6) ~ "**GroupBars**;ITEMINDEXA;KEYA;STARTA;ITEMINDE specifies that two bars has been grouped • exChartUndoRedoUngroupBars(7) ~ "**UngroupBars**;ITEMINDEXA;KEYA;STARTA;ITEMIND specifies that two bars has been ungrouped • exChartUndoRedoDefineSummaryBars(8) ~ "**DefineSummaryBars**;SUMMARYITEMINDEX;SUMM indicates that a bar has been defined as a child of a summary bar • exChartUndoRedoUndefineSummaryBars(9) ~ "**UndefineSummaryBars**;SUMMARYITEMINDEX;SUM

	indicates that a bar has been removed from the summary bar's children • exChartUndoRedoAddLink(10) ~ "**AddLink**;KEY", indicates that a new link has been created • exChartUndoRedoRemoveLink(11) ~ "**RemoveLink**;KEY", indicates that a link has been removed • exChartUndoRedoUpdateLink(12) ~ "**UpdateLink**;KEY", specifies that one of more properties of the link has been updated (Link property, this operation can be added only using the StartUpdateLink / EndUpdateLink methods) • exListUndoRedoAddItem(13) ~ "**AddItem**;ITEMINDEX", indicates that a new item has been created • exListUndoRedoRemoveItem(14) ~ "**RemoveItem**;ITEMINDEX", indicates that an item has been removed • exListUndoRedoChangeItemPos(15) ~ "**ChangeItemPos**;ITEMINDEX", indicates that an item changes its position or / and parent • exListUndoRedoChangeCellValue(16) ~ "**ChangeCellValue**;ITEMINDEX;CELLINDEX", indicates that the cell's value has been changed • exListUndoRedoChangeCellState(17) ~ "**ChangeCellState**;ITEMINDEX;CELLINDEX", indicates that the cell's state has been changed For instance, RedoListAction(0) shows only AddBar actions in the redo stack.
Count as Variant	[optional] A long expression that indicates the number of actions being listed. If missing or -1, all actions are listed. For instance, RedoListAction(0,1) shows only the first AddBar action being added to the redo stack.
String	A String expression that lists the Redo actions that may be performed.

The RedoListAction property lists the Redo actions that can be performed in the chart. The ChartStartChanging(exUndo/exRedo) / ChartEndChanging(exUndo/exRedo) event notifies your application whenever an Undo/Redo operation is performed. The UndoListAction property lists the actions that the user may perform by doing Undo operations. The CanRedo property specifies whether a redo operation can be performed if CTRL+Y key is

pressed. Use the RedoRemoveAction method to remove the first action from the redo queue.

The records of the Undo/Redo queue may contain actions in the following format:

- "**AddBar**;ITEMINDEX;KEY", indicates that a new bar has been created
- "**RemoveBar**;ITEMINDEX;KEY", indicates that a bar has been removed
- "**MoveBar**;ITEMINDEX;KEY", indicates that a bar has been moved or resized
- "**PercentChange**;ITEMINDEX;KEY", indicates that the bar's percent has been changed
- "**UpdateBar**;ITEMINDEX;KEY", indicates that one or more properties of the bar has been updated (ItemBar property, this operation can be added only using the StartUpdateBar / EndUpdateBar methods)
- "**ParentChangeBar**;ITEMINDEX;KEY", indicates that the bar's parent has been changed
- "**GroupBars**;ITEMINDEXA;KEYA;STARTA;ITEMINDEXB;KEYB;STARTB", specifies that two bars has been grouped
- "**UngroupBars**;ITEMINDEXA;KEYA;STARTA;ITEMINDEXB;KEYB;STARTB", specifies that two bars has been ungrouped
- "**DefineSummaryBars**;SUMMARYITEMINDEX;SUMMARYKEY;ITEMINDEX;KEY", indicates that a bar has been defined as a child of a summary bar
- "**UndefineSummaryBars**;SUMMARYITEMINDEX;SUMMARYKEY;ITEMINDEX;KEY", indicates that a bar has been removed from the summary bar's children
- "**AddLink**;KEY", indicates that a new link has been created
- "**RemoveLink**;KEY", indicates that a link has been removed
- "**UpdateLink**;KEY", specifies that one of more properties of the link has been updated (Link property, this operation can be added only using the StartUpdateLink / EndUpdateLink methods)

The records of the Undo/Redo queue may contain actions in the following format (available starting from 23.0):

- "**AddItem**;ITEMINDEX", indicates that a new item has been created
- "**RemoveItem**;ITEMINDEX", indicates that an item has been removed
- "**ChangeItemPos**;ITEMINDEX", indicates that an item changes its position or / and parent
- "**ChangeCellValue**;ITEMINDEX;CELLINDEX", indicates that the cell's value has been changed
- "**ChangeCellState**;ITEMINDEX;CELLINDEX", indicates that the cell's state has been changed

Also, the Undo/Redo queue may include:

- "**StartBlock**", specifies that a block of operations begins
- "**EndBlock**", specifies that a block of operations ends

Each action is on a single line, and each field is separated by ; character. The lines are separated by "\r\n" characters (vbCrLf in VB).

For instance,

- An AddLink action in the Undo stack means that an Undo operation will perform a RemoveLink action.
- A RemoveLink action in the Undo stack means that an Undo operation will perform an AddLink action.
- An AddLink action in the Redo stack means that a Redo operation will perform an AddLink action.
- A RemoveLink action in the Redo stack means that a Redo operation will perform a RemoveLink action.

Here's a sample format of the RedoListAction property may get:

```
AddBar;1;
AddBar;2;
DefineSummaryBars;1;;2;
AddBar;3;E
DefineSummaryBars;1;;3;E
AddLink;L1
GroupBars;2;;0;3;E;-1
GroupBars;2;;0;3;E;-1
AddBar;4;E
AddLink;L2
GroupBars;3;E;0;4;E;-1
GroupBars;3;E;0;4;E;-1
DefineSummaryBars;1;;4;E
AddBar;4;
AddBar;3;
DefineSummaryBars;4;;3;
AddBar;2;E
DefineSummaryBars;4;;2;E
AddBar;1;E
DefineSummaryBars;4;;1;E
MoveBar;1;
```

```
StartBlock
MoveBar;1;E
MoveBar;2;E
MoveBar;3;
MoveBar;4;
EndBlock
```

The following VB sample splits the RedoListAction value and adds each action to a listbox control:

```
List1.Clear
Dim s() As String
s = Split(G2antt1.Chart.RedoListAction, vbCrLf)
For i = LBound(s) To UBound(s)
        List1.AddItem s(i)
Next
```

method Chart.RedoRemoveAction ([Action as Variant], [Count as Variant])

Removes the last the redo actions that can be performed in the chart.

Type	Description
Action as Variant	[optional] A long expression that specifies the action being removed. If missing or -1, all actions are removed. The Action parameter can be one of the following: • exChartUndoRedoAddBar(0) ~ "**AddBar**;ITEMINDEX;KEY", indicates that a new bar has been created • exChartUndoRedoRemoveBar(1) ~ "**RemoveBar**;ITEMINDEX;KEY", indicates that a bar has been removed • exChartUndoRedoMoveBar(2) ~ "**MoveBar**;ITEMINDEX;KEY", indicates that a bar has been moved or resized • exChartUndoRedoPercentChange(3) ~ "**PercentChange**;ITEMINDEX;KEY", indicates that the bar's percent has been changed • exChartUndoRedoUpdateBar(4) ~ "**UpdateBar**;ITEMINDEX;KEY", indicates that one or more properties of the bar has been updated (ItemBar property, this operation can be added only using the StartUpdateBar / EndUpdateBar methods) • exChartUndoRedoParentChangeBar(5) ~ "**ParentChangeBar**;ITEMINDEX;KEY", indicates that the bar's parent has been changed • exChartUndoRedoGroupBars(6) ~ "**GroupBars**;ITEMINDEXA;KEYA;STARTA;ITEMINDE specifies that two bars has been grouped • exChartUndoRedoUngroupBars(7) ~ "**UngroupBars**;ITEMINDEXA;KEYA;STARTA;ITEMIND specifies that two bars has been ungrouped • exChartUndoRedoDefineSummaryBars(8) ~ "**DefineSummaryBars**;SUMMARYITEMINDEX;SUMM indicates that a bar has been defined as a child of a summary bar • exChartUndoRedoUndefineSummaryBars(9) ~ "**UndefineSummaryBars**;SUMMARYITEMINDEX;SUM

	indicates that a bar has been removed from the summary bar's children • exChartUndoRedoAddLink(10) ~ "**AddLink**;KEY", indicates that a new link has been created • exChartUndoRedoRemoveLink(11) ~ "**RemoveLink**;KEY", indicates that a link has been removed • exChartUndoRedoUpdateLink(12) ~ "**UpdateLink**;KEY", specifies that one of more properties of the link has been updated (Link property, this operation can be added only using the StartUpdateLink / EndUpdateLink methods) • exListUndoRedoAddItem(13) ~ "**AddItem**;ITEMINDEX", indicates that a new item has been created • exListUndoRedoRemoveItem(14) ~ "**RemoveItem**;ITEMINDEX", indicates that an item has been removed • exListUndoRedoChangeItemPos(15) ~ "**ChangeItemPos**;ITEMINDEX", indicates that an item changes its position or / and parent • exListUndoRedoChangeCellValue(16) ~ "**ChangeCellValue**;ITEMINDEX;CELLINDEX", indicates that the cell's value has been changed • exListUndoRedoChangeCellState(17) ~ "**ChangeCellState**;ITEMINDEX;CELLINDEX", indicates that the cell's state has been changed For instance, RedoListAction(0) removes only AddBar actions in the redo stack.
Count as Variant	[optional] A long expression that indicates the number of actions to be removed. If missing or -1, all actions are removed. For instance, RedoListAction(0,1) removes only the first AddBar action from the redo stack.

Use the RedoRemoveAction method to remove the first action from the redo queue. Use the RedoRemoveAction() (with no parameters) to remove all redo actions. The RedoListAction property retrieves the list of actions that an redo operation can perform. The UndoRemoveAction method removes the last action to be performed if the Undo method is invoked.

The records of the Undo/Redo queue may contain actions in the following format:

- "**AddBar**;ITEMINDEX;KEY", indicates that a new bar has been created
- "**RemoveBar**;ITEMINDEX;KEY", indicates that a bar has been removed
- "**MoveBar**;ITEMINDEX;KEY", indicates that a bar has been moved or resized
- "**PercentChange**;ITEMINDEX;KEY", indicates that the bar's percent has been changed
- "**UpdateBar**;ITEMINDEX;KEY", indicates that one or more properties of the bar has been updated (ItemBar property, this operation can be added only using the StartUpdateBar / EndUpdateBar methods)
- "**ParentChangeBar**;ITEMINDEX;KEY", indicates that the bar's parent has been changed
- "**GroupBars**;ITEMINDEXA;KEYA;STARTA;ITEMINDEXB;KEYB;STARTB", specifies that two bars has been grouped
- "**UngroupBars**;ITEMINDEXA;KEYA;STARTA;ITEMINDEXB;KEYB;STARTB", specifies that two bars has been ungrouped
- "**DefineSummaryBars**;SUMMARYITEMINDEX;SUMMARYKEY;ITEMINDEX;KEY", indicates that a bar has been defined as a child of a summary bar
- "**UndefineSummaryBars**;SUMMARYITEMINDEX;SUMMARYKEY;ITEMINDEX;KEY", indicates that a bar has been removed from the summary bar's children
- "**AddLink**;KEY", indicates that a new link has been created
- "**RemoveLink**;KEY", indicates that a link has been removed
- "**UpdateLink**;KEY", specifies that one of more properties of the link has been updated (Link property, this operation can be added only using the StartUpdateLink / EndUpdateLink methods)

The records of the Undo/Redo queue may contain actions in the following format (available starting from 23.0):

- "**AddItem**;ITEMINDEX", indicates that a new item has been created
- "**RemoveItem**;ITEMINDEX", indicates that an item has been removed
- "**ChangeItemPos**;ITEMINDEX", indicates that an item changes its position or / and parent
- "**ChangeCellValue**;ITEMINDEX;CELLINDEX", indicates that the cell's value has been changed
- "**ChangeCellState**;ITEMINDEX;CELLINDEX", indicates that the cell's state has been changed

Also, the Undo/Redo queue may include:

- "**StartBlock**", specifies that a block of operations begins
- "**EndBlock**", specifies that a block of operations ends

.

method Chart.RemoveNonworkingDate (Date as Variant)

Removes a nonworking date.

Type	Description
Date as Variant	A Date or a string expression that indicates the date/expression being unmarked as nonworking day, exactly how it was previously added using the AddNonworkingDate method.

Use the RemoveNonworkingDate method to unmark a specified nonworking date, being previously added using the AddNonworkingDate method. Use the ClearNonworkingDates method to remove all nonworking dates. Use the IsDateVisible property to specify whether a date fits the chart's area. Use the IsNonworkingDate property to check whether the date is already highlighted as nonworking day. The NonworkingDays property specifies the days being marked as nonworking in a week. Use the NonworkingDaysPattern property to specify the pattern being used to fill non-working days. The NonworkingDaysColor property specifies the color being used to fill the non-working days.

method Chart.RemoveSelection ()

Removes the selected objects (bars or links) within the chart.

Type	Description

Use the RemoveSelection method to remove the objects (bars, links) in the chart's selection. For instance, call the RemoveSelection method when the user presses the Delete key. The AllowSelectObjects property allows users to select at runtime the bars and links in the chart area. Use the ItemBar(exBarSelected) property to select or unselect programmatically a bar. Use the Link(exLinkSelected) property to select or unselect programmatically a link. The RemoveSelection method calls the RemoveBar to remove a bar from the selection, or RemoveLink method to remove a link from the selection. Call the AllowSelectObjects property to empty the selection, as AllowSelectObjects = AllowSelectObjects. The SelectedObjects property returns the bars/links selected in the chart section of the control. The RemoveSelection method removes the selected items (including the descendents). The RemoveSelection method removes the selected links/bars from the chart if exists, else it removes the selected items (including the descendents).

The following VB sample removes the selected links only:

```
With G2antt1
  .BeginUpdate
  With .Items
    For Each l In .SelectedObjects(exSelectLinksOnly)
      G2antt1.Template = "Items.RemoveLink(" & l & ")"
    Next
  End With
  .EndUpdate
End With
```

The following VB sample removes the selected bars only:

```
With G2antt1
  .BeginUpdate
  With .Items
    For Each b In .SelectedObjects(exSelectBarsOnly)
      G2antt1.Template = "Items.RemoveBar(" & b & ")"
    Next
  End With
  .EndUpdate
```

End With

When a bar is removed, any link related to it will be removed.

method Chart.RemoveTimeZone (Key as Variant)

Removes a time-zone being highlighted using the MarkTimeZone method.

Type	Description
Key as Variant	A String expression that specifies the key of the date-time zone to be removed. The Key parameter of the supports a pattern with wild characters as *,?,# or [], if the Key starts with "<" and ends on ">" aka "<Z*>" which indicates all date-time zones with the key Z or starts on Z. If the Key parameter specifies a pattern, it removes all date-time zones, with the key that matches the giving pattern.

A zone can be used to highlight a range of dates, specifying the start and end zone. Use the RemoveTimeZone method to deletes the time zone being added previously using the MarkTimeZone method. The TimeZoneFromPoint property retrieves the key of the time-zone from the cursor. The TimeZoneInfo property retrieves information about the time-zone giving its key. The MarkTodayColor property specifies the color to mark the today date. Use the SelectDate property to select a date by clicking the chart's header. Use the MarkNowColor property to show a vertical bar that indicates the current date-time in the control's chart, from seconds to seconds, minutes, and so on. For instance, the Chart.RemoveTimeZone("<*>") removes/clear all date-time zones.

property Chart.ResizeUnitCount as Long

Specifies the number of time units while resizing, moving or creating bars by dragging.

Type	Description
Long	A long expression that indicates the number of time units while resizing, moving or creating bars by dragging.

Use the ResizeUnitScale and ResizeUnitCount properties to specify a different resizing units. If these properties are not used, the UnitScale property indicates the resizing time units. For instance, using the ResizeUnitScale and ResizeUnitCount properties you can let the user resizing the bars up to hours even if the chart displays days. Use the ResizeUnitScale and ResizeUnitCount properties to refine the way how the user resizes or creates new bars.

property Chart.ResizeUnitScale as UnitEnum

Retrieves or sets a value that indicates the base time unit while resizing, moving or creating the bars by dragging.

Type	Description
UnitEnum	A UnitEnum expression that specifies the base time unit while resizing, moving or creating the bars by dragging.

Use the ResizeUnitScale and ResizeUnitCount properties to specify a different resizing units. If these properties are not used, the UnitScale property indicates the resizing time units. For instance, using the ResizeUnitScale and ResizeUnitCount properties you can let the user resizing the bars up to hours even if the chart displays days. Use the ResizeUnitScale and ResizeUnitCount properties to refine the way how the user resizes or creates new bars. The InsideUnit property specifies the time scale unit being used to paint the inside zoom units.

Your application can provide some options to help user while performing moving or resizing the bars at runtime as follow:

- grid lines, that can be shown only when moving or resizing, using the ChartStartChanging and ChartEndChanging events
- select date, to specify the margins of the area you want to highlight
- ticker, that shows the cursor's position in the chart, or while resizing, it shows the size and the position of the bar
- ability to specify a resizing/moving unit, different that the displayed one ie while the chart displays days, you can specify the resizing unit on hours.
- inside zoom, that can be used to magnify the portion of the chart being selected

property Chart.ScrollBar as Boolean

Shows or hides the chart's horizontal scroll bar.

Type	Description
Boolean	A Boolean expression that indicates whether the horizontal scroll bar is visible in the chart.

Use the ScrollBar property to show or hide the chart's scroll bar. The FirstVisibleDate property indicates the first visible date. The ToolTip property indicates the tooltip being shown when the user clicks the thumb of the chart's scrollbar. Use the FirstVisibleDate property to indicate the first visible date when the chart contains no scroll bar. Use the ScrollTo method to ensure that a date fits the chart's client area. Use the Zoom method to zoom the chart to an interval of dates. Use the ScrollBars property.

property Chart.ScrollRange(Pos as ScrollRangeEnum) as Variant

Specifies the range of dates to scroll within.

Type	Description
Pos as ScrollRangeEnum	A ScrollrangeEnum expression that indicates whether the starting or ending position of the range is beging requested or changed.
Variant	A Variant expression that indicates the date or the time when the range beings or ends.

By default, the ScrollRange(exStartDate) and ScrollRange(exEndDate) properties are empty. The control scrolls the chart within specified range, only if the ScrollRange(exStartDate) and ScrollRange(exEndDate) are not empty and indicates a valid date-time value. If the ScrollRange(exStartDate) and ScrollRange(exEndDate) properties indicates the same valid value, the ScrollRange limits the view to specified unit. For instance, if both are set on #1/1/2001# the view is limited to full day, in case it is zoomed to hours, minutes or seconds. The ScrollRange property rearranges the FirstVisibleDate property, so it fits the range. The FirstVisibleDate indicates the first visible date or time in the chart. Use the ScrollTo method to scroll to a specified date. For instance, let's say that ScrollRange(exStartDate) is #5/1/2007#, ScrollRange(exEndDate) is #10/1/2007#, and the FirstVisibleDate is #7/1/2007#. This would move the first visible day to July 1st, but also move the horizontal scroll bar halfway across the chart. This way, it would be clear to users where they are in relation to the full schedule. The DateChange event notifies whether the first visible date is changed. Use the ScrollPartEnum property to disable specified parts in the chart's scroll bar.

If you want to limit just a margin of the chart, you can handle the DateChange event to specify the correct value for the FirstVisibleDate property of the Chart object like in the following VB6 sample:

```
Private Function Max(a, b)
   If (a < b) Then
      Max = b
   Else
      Max = a
   End If
End Function

Private Sub G2antt1_DateChange()
   With G2antt1
```

```
    Dim dLimit As Date
    dLimit = #1/1/2010#
    .Chart.FirstVisibleDate = Max(dLimit, .Chart.FirstVisibleDate)
    .ScrollPartEnable(exHChartScroll, exLeftBPart) = .Chart.FirstVisibleDate > dLimit
  End With
End Sub
```

The sample limits the FirstVisibleDate property of the chart so it won't be less than January 1st of 2010, and disables the left scroll button in the chart if FirstVisibleDate property is at limit.

The following VB sample disables the left and right scroll bar buttons, and specifies a range of date to scroll within:

```
With G2antt1
  .Columns.Add "Task"
  With .Chart
    .LevelCount = 2
    .PaneWidth(0) = 56
    .ScrollRange(exStartDate) = "1/1/2001"
    .ScrollRange(exEndDate) = "1/31/2001"
    .FirstVisibleDate = "1/12/2001"
  End With
  With .Items
    h = .AddItem("Task 1")
    .AddBar h,"Task","1/15/2001","1/18/2001","K1"
    h = .AddItem("Task 1")
    .AddBar h,"Task","1/5/2001","1/11/2001","K1"
  End With
End With
```

The following VB.NET sample disables the left and right scroll bar buttons, and specifies a range of date to scroll within:

```
Dim h
With AxG2antt1
  .Columns.Add "Task"
  With .Chart
    .LevelCount = 2
```

```
    .PaneWidth(0) = 56
    .ScrollRange(EXG2ANTTLib.ScrollRangeEnum.exStartDate) = "1/1/2001"
    .ScrollRange(EXG2ANTTLib.ScrollRangeEnum.exEndDate) = "1/31/2001"
    .FirstVisibleDate = "1/12/2001"
  End With
  With .Items
    h = .AddItem("Task 1")
    .AddBar h,"Task","1/15/2001","1/18/2001","K1"
    h = .AddItem("Task 1")
    .AddBar h,"Task","1/5/2001","1/11/2001","K1"
  End With
End With
```

The following C# sample disables the left and right scroll bar buttons, and specifies a range of date to scroll within:

```
axG2antt1.Columns.Add("Task");
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
  var_Chart.LevelCount = 2;
  var_Chart.set_PaneWidth(0 != 0,56);
  var_Chart.set_ScrollRange(EXG2ANTTLib.ScrollRangeEnum.exStartDate,"1/1/2001");
  var_Chart.set_ScrollRange(EXG2ANTTLib.ScrollRangeEnum.exEndDate,"1/31/2001");
  var_Chart.FirstVisibleDate = "1/12/2001";
EXG2ANTTLib.Items var_Items = axG2antt1.Items;
  int h = var_Items.AddItem("Task 1");
  var_Items.AddBar(h,"Task","1/15/2001","1/18/2001","K1",null);
  h = var_Items.AddItem("Task 1");
  var_Items.AddBar(h,"Task","1/5/2001","1/11/2001","K1",null);
```

The following C++ sample disables the left and right scroll bar buttons, and specifies a range of date to scroll within:

```
/*
  Copy and paste the following directives to your header file as
  it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

  #import "D:\\Exontrol\\ExG2antt\\project\\Debug\\ExG2antt.dll"
  using namespace EXG2ANTTLib;
```

```
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->GetColumns()->Add(L"Task");
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
    var_Chart->PutLevelCount(2);
    var_Chart->PutPaneWidth(0,56);
    var_Chart->PutScrollRange(EXG2ANTTLib::exStartDate,"1/1/2001");
    var_Chart->PutScrollRange(EXG2ANTTLib::exEndDate,"1/31/2001");
    var_Chart->PutFirstVisibleDate("1/12/2001");
EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems();
    long h = var_Items->AddItem("Task 1");
    var_Items->AddBar(h,"Task","1/15/2001","1/18/2001","K1",vtMissing);
    h = var_Items->AddItem("Task 1");
    var_Items->AddBar(h,"Task","1/5/2001","1/11/2001","K1",vtMissing);
```

The following VFP sample disables the left and right scroll bar buttons, and specifies a range of date to scroll within:

```
Dim h
With AxG2antt1
    .Columns.Add "Task"
    With .Chart
        .LevelCount = 2
        .PaneWidth(0) = 56
        .ScrollRange(EXG2ANTTLib.ScrollRangeEnum.exStartDate) = "1/1/2001"
        .ScrollRange(EXG2ANTTLib.ScrollRangeEnum.exEndDate) = "1/31/2001"
        .FirstVisibleDate = "1/12/2001"
    End With
    With .Items
        h = .AddItem("Task 1")
        .AddBar h,"Task","1/15/2001","1/18/2001","K1"
        h = .AddItem("Task 1")
        .AddBar h,"Task","1/5/2001","1/11/2001","K1"
    End With
End With
```

method Chart.ScrollTo (Date as Date, [Align as Variant])

Scrolls the chart so the specified date is visible.

Type	Description
Date as Date	A Date expression that indicates the date being ensured that's visible.
Align as Variant	An AlignmentEnum expression that indicates where the date will be placed.

Use the ScrollTo method to ensure that specified date fits the chart's area. The FirstVisibleDate property specifies the first visible date. The ScrollTo method fires the DateChange event if the first visible date is changed. Use the Zoom method to zoom the chart to a specified interval of dates. Use the PaneWidth property to specify the width of the chart. Use the Scroll method to scroll vertically the control. Use the EnsureVisibleItem method to ensure that a specified item fits the control's client area. Use the ScrollPos property to get the control's scroll position. Use the EnsureVisibleColumn method to ensure that a specified column fits the control's client area.

The following VB sample ensures that the "6/1/2005" is listed in the center of the chart:

```
With G2antt1.Chart
   .ScrollTo #6/1/2005#, AlignmentEnum.CenterAlignment
End With
```

The following C++ sample ensures that the "6/1/2005" is listed in the center of the chart:

```
COleDateTime date( 2005, 6, 1, 0, 0, 0 );
CChart chart = m_g2antt.GetChart();
chart.ScrollTo( date.m_dt, COleVariant( (long)1 ) );
```

The following VB.NET sample ensures that the "6/1/2005" is listed in the center of the chart:

```
With AxG2antt1.Chart
   .ScrollTo(DateTime.Parse("6/1/2005"), EXG2ANTTLib.AlignmentEnum.CenterAlignment)
End With
```

The following C# sample ensures that the "6/1/2005" is listed in the center of the chart:

```
axG2antt1.Chart.ScrollTo(DateTime.Parse("6/1/2005"),
EXG2ANTTLib.AlignmentEnum.CenterAlignment);
```

The following VFP sample ensures that the "6/1/2005" is listed in the center of the chart:

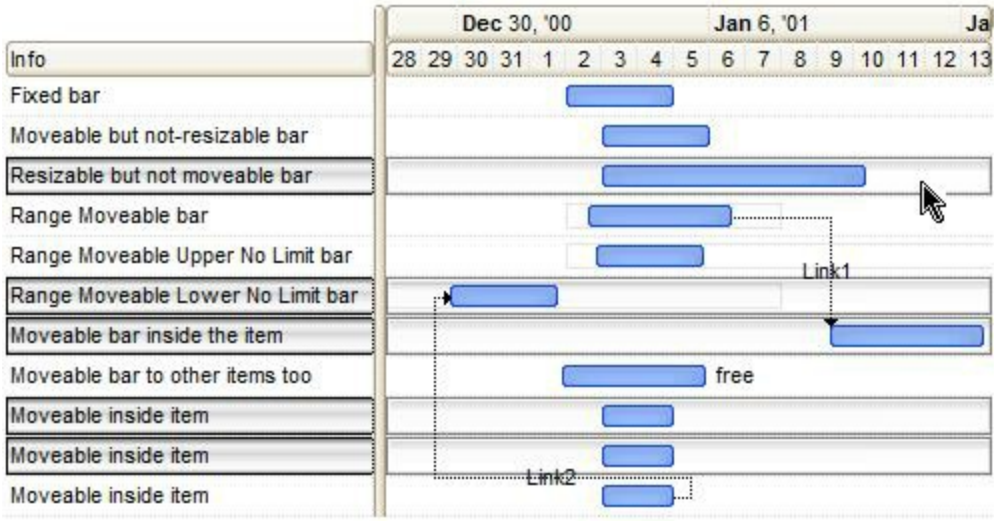
```
With thisform.G2antt1.Chart
    .ScrollTo( "6/2/2005", 1 )
EndWith
```

property Chart.SelBackColor as Color

Retrieves or sets a value that indicates the selection background color.

Type	Description
Color	A color expression that indicates the background color to display the selected items in the chart area. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

By default, the SelBackColor property is the same as chart's background color that's specified by BackColor property of the Chart object. In other words, by default, the chart does not display a different background color for selected items in the chart area. The SelBackColor property of the Chart object changes the background for the selected items in the chart area. Use the SelBackColor property to change the selection background color in the list area. Use the SelForeColor property to change the foreground color of the selected items in the chart area. *The SelBackColor property is applied ONLY if the SelBackColor property is different that the BackColor property.* Use the SelectOnClick property to disable selecting new items when the user clicks the chart area. The SelBarColor property specifies the color to highlight the selected bars.



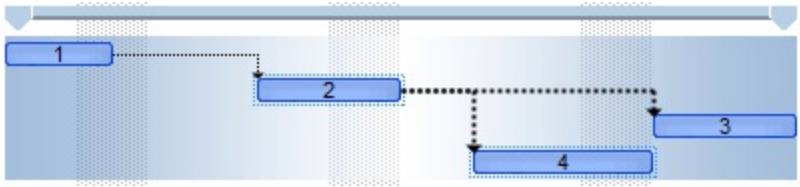
property Chart.SelBarColor as Color

Retrieves or sets a value that indicates the color or EBN object to display the selected bars.

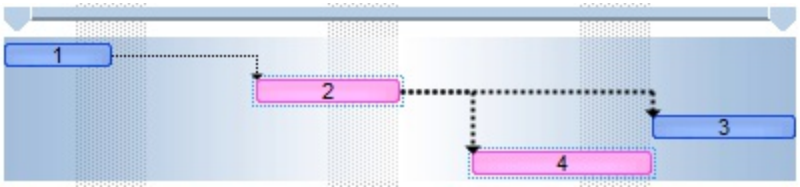
Type	Description
Color	A Color expression that specifies the color for selected bars or the EBN object to show the selected bar. If a color expression is used, the chart displays the bars using a different color as it would be specify using the ItemBar(exBarColor) property. If the color refers an EBN object it is displayed over the selected bars. You can use the CP option (COPY option) of the EBN objects to enlarge the EBN object being displayed on the selected bars. Use the Add method to add new EBN/skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

By default, the SelBarColor property is 0. If the SelBarColor property is 0, the default frame is shown around the selected bars as shown in the following screen shot. Use the SelBarColor property to specify a different color or EBN object to display the selected bars. The SelForeColor property retrieves or sets a value that indicates the selection foreground color. The SelLinkColor property specifies the color to show the selected link (or for rectangular links an EBN object to highlights the link, without changing the link's color). The SelBackColor property indicates the visual appearance for the item's background on the chart area.

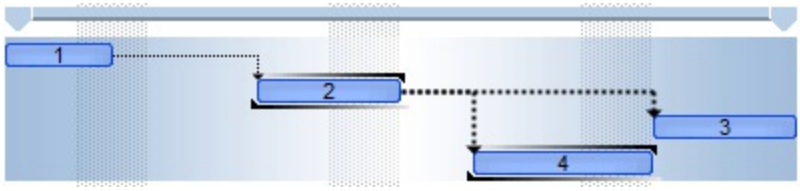
The following screen shot shows the selected bars (2 and 4) when SelBarColor property is 0 (by default)



The following screen shot shows the selected bars (2 and 4) when SelBarColor property is RGB(255,0,0) (red)



The following screen shot shows the selected bars (2 and 4) when SelBarColor property is 0x5000000 and using this  EBN file.



The following VB sample assign a different EBN object to display the selected bars:

```
With G2antt1
   .BeginUpdate
   With .VisualAppearance
      .Add 4, "select.ebn"
      .Add 5, "CP:4 -4 -3 3 2"
   End With
   .Chart.SelBarColor = &H5000000
   .EndUpdate
End With
```

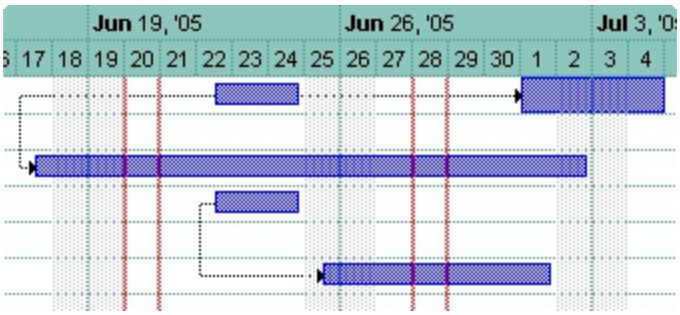
property Chart.SelectDate(Date as Date) as Boolean

Selects or unselects a specific date in the chart.

Type	Description
Date as Date	A DATE expression that indicates the
Boolean	A Boolean expression that specifies whether the Date is selected or not.

Use the SelectDate property to select dates programmatically. The SelectedDates property can be used to retrieve all selected dates, or to select a collection of dates. Use the UnselectDates method to unselect all dates in the chart. Use the SelectDate property to select or unselect a new date, or to find if a specified date is selected or it is not selected. The user can select dates by clicking the chart's header. Use the SelectLevel property to specify the area being highlighted when a date is selected. You can select multiple dates keeping the CTRL key and clicking a new date. The MarkSelectDateColor property specifies the color being used to highlight the selected dates. If the MarkSelectDateColor property is identical with the BackColor property of the Chart object, the selected dates are not shown. The ChartEndChanging(exSelectDate) event notifies your application when the user selects a new date by clicking the header of the chart. Use the MarkTimeZone method to highlight different time-zones.

In the following screen shot the red lines marks the selected dates (June 20 and June 28):



property Chart.SelectDates as Variant

Indicates a collection of date-time units being selected.

Type	Description
Variant	A Safe array that includes the dates being selected.

The SelectedDates property can be used to retrieve all selected dates, or to select a collection of dates. The SelectDate property may be used to select or unselect a specified date-time unit. Use the UnselectDates method to unselect all dates in the chart. The ChartEndChanging(exSelectDate) event notifies your application when the user selects a new date by clicking the header of the chart. Use the SelectLevel property to specify the area being highlighted when a date is selected. You can select multiple dates keeping the CTRL key and clicking a new date. The MarkSelectDateColor property specifies the color being used to highlight the selected dates. If the MarkSelectDateColor property is identical with the BackColor property of the Chart object, the selected dates are not shown. Use the MarkTimeZone method to highlight different time-zones.

The following VB sample displays the selected dates:

```
Private Sub G2antt1_ChartEndChanging(ByVal Operation As
EXG2ANTTLibCtl.BarOperationEnum)
   If (Operation = exSelectDate) Then
      Debug.Print "Selected"
      For Each d In G2antt1.Chart.SelectDates
         Debug.Print d
      Next
   End If
End Sub
```

The following VB sample changes the collection of selected dates:

```
With G2antt1.Chart
   .SelectLevel = 1
   .SelectDates = Array(#6/22/2005#, #6/23/2005#)
End With
```

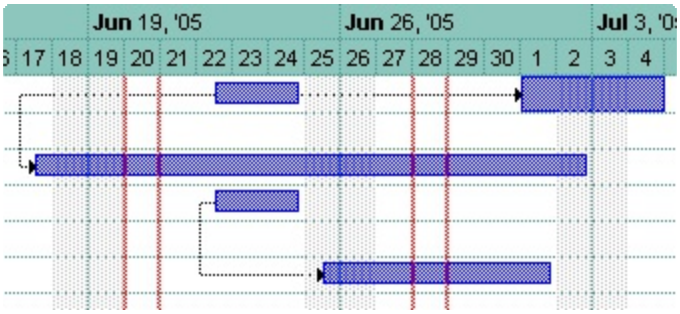
property Chart.SelectLevel as Long

Indicates the index of the level that highlights the selected dates.

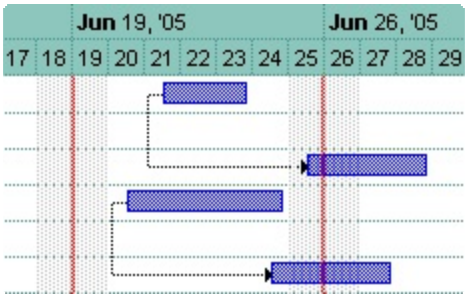
Type	Description
Long	A long expression that indicates the index of the level being selected.

Use the SelectLevel property to specify the area being highlighted when a date is selected. For instance, if you click a date in the first level (in the chart's header), the chart displays the selected date accordingly to the selected level. Use the SelectDate property to select or unselect a new date, or to find if a specified date is selected or not. Use the LevelFromPoint property to retrieve the index of the level from the cursor. You can select multiple dates keeping the CTRL key and clicking a new date. The MarkSelectDateColor property specifies the color being used to highlight the selected dates. If the MarkSelectDateColor property is identical with the BackColor property of the Chart object, the selected dates are not shown.

In the following screen shot the red lines marks the selected dates (June 20 and June 28, as the user clicks the June 20, 28 dates in the second level (index 1) where the days are displayed):



In the following screen shot the red lines marks the selected week (June 19 to June 26, as the user clicks the June 19, `05 week in the first level (index 0) where weeks are displayed):



property Chart.SelectOnClick as Boolean

Specifies whether an item gets selected once the user clicks the chart area.

Type	Description
Boolean	A Boolean expression that specifies whether the user selects an item when a bar is clicked.

By default, the SelectOnClick property is True. By default, the item being clicked gets selected no matter of what part of the control is clicked: list or chart area. Use the SelectOnClick property to disable selecting new items when the user clicks the chart area. The SingleSel property specifies whether the control supports multiple selection. For instance, this property can be useful once you use the HistogramView property on exHistogramSelectedItems (so the selected items are displayed in the histogram). User may click the chart area, so new bars are created or creating new links, without changing the selection and so the items being displayed in the histogram.

property Chart.SelForeColor as Color

Retrieves or sets a value that indicates the selection foreground color.

Type	Description
Color	A color expression that specifies the foreground color for selected items that's displayed on the chart area.

By default, the SelForeColor property is the same as chart's foreground color that's specified by ForeColor property of the Chart object. In other words, by default, the chart does not display a different foreground color for selected items in the chart area. The SelForeColor property of the Chart object changes the foreground for the selected items **in the chart area**. Use the SelForeColor property to change the selection foreground color in the list area. Use the SelBackColor property to change the background color of the selected items in the chart area. *The SelForeColor property is applied ONLY if the SelForeColor property is different that the ForeColor property.*

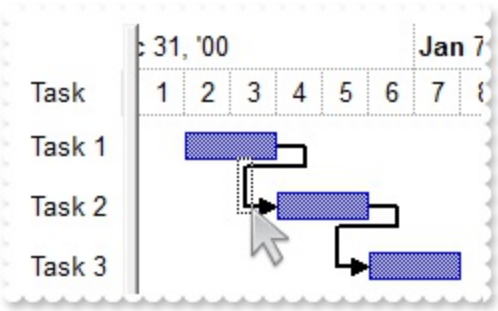
property Chart.SelLinkColor as Color

Specifies the color to show the selected link (or for rectangular links an EBN object to highlights the link, without changing the link's color).

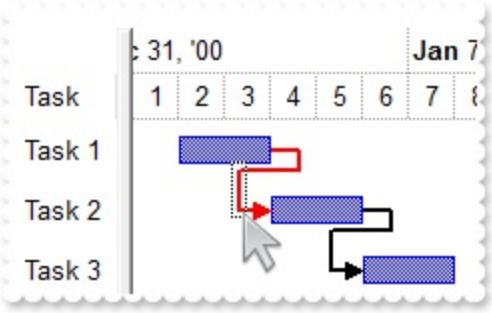
Type	Description
Color	A Color expression that specifies the color for selected bars or the EBN object to show the selected bar. If a color expression is used, the chart displays the bars using a different color as it would be specify using the ItemBar(exBarColor) property. If the color refers an EBN object it is displayed over the selected bars. You can use the CP option (COPY option) of the EBN objects to enlarge the EBN object being displayed on the selected bars. Use the Add method to add new EBN/skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

By default, the SelLinkColor property is 0. Use the SelLinkColor property to specify a different color or EBN object to display the selected links. The SelLinkColor property specifies the color to show the selected link (or for rectangular links an EBN object to highlights the link, without changing the link's color). The SelBarColor property retrieves or sets a value that indicates the color or EBN object to display the selected bars.

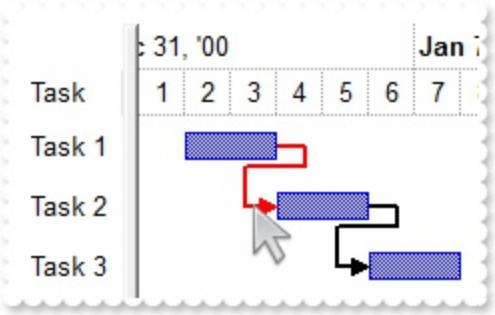
The following screen shot shows how a selected rectangular-link is displayed by default:



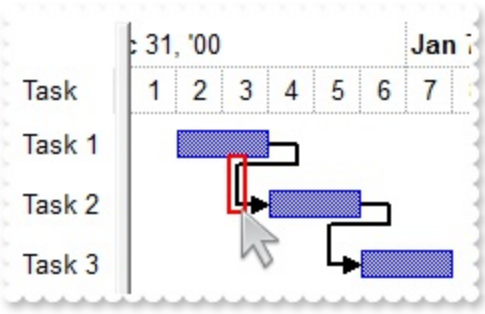
The following screen shot shows how a selected rectangular-link is displayed with a different color plus the default frame around:



The following screen shot shows how a selected rectangular-link is displayed with a different (without the default frame):



The following screen shot shows how a selected rectangular-link is displayed with a frame (EBN object):



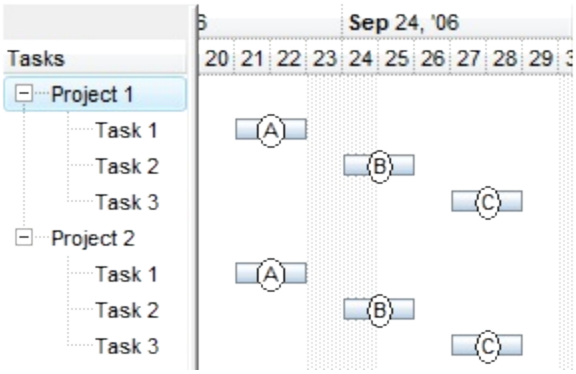
property Chart.ShowCollapsedBars as Boolean

Gets or sets a value that indicates whether the collapsed items displays their child bars.

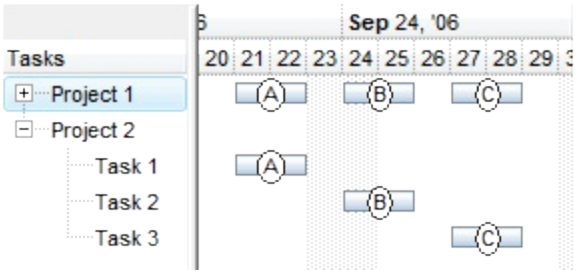
Type	Description
Boolean	A Boolean expression that specifies whether the child bars are displayed in the parent item if it is collapsed.

By default, the ShowCollapsedBars property is False. By default, the child bars are not shown in the parent item. Use the ShowCollapsedBars property to display child bars when an item is collapsed. If the ShowCollapsedBars property is True, a collapsed item always displays its child bars. Use the InsertItem method to insert a child items. Use the AddBar method to add a new bar to the item. Currently, the control includes all child-bars (recursively), while previously only the direct-child were included.

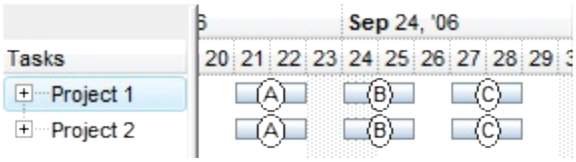
The following screen shot shows the bars as the items are not collapsed (ShowCollapsedBars property is True) :



Once the Project 1 item gets collapsed we get (ShowCollapsedBars property is True) :



or both items collapsed we get (ShowCollapsedBars property is True) :



The following VB sample shows the child bars for collapsed items:

```
With G2antt1
```

```
   .BeginUpdate
   .LinesAtRoot = exLinesAtRoot
   .Columns.Add "Tasks"
   With .Chart
      .FirstVisibleDate = #9/20/2006#
      .ShowCollapsedBars = True
      .LevelCount = 2
      .PaneWidth(0) = 96
   End With
   With .Items
      h = .AddItem("Project 1")
      h1 = .InsertItem(h,0,"Task 1")
      .AddBar h1,"Task",#9/21/2006#,#9/23/2006#,"A"
      h2 = .InsertItem(h,0,"Task 2")
      .AddBar h2,"Task",#9/24/2006#,#9/26/2006#,"B"
      h3 = .InsertItem(h,0,"Task 3")
      .AddBar h3,"Task",#9/27/2006#,#9/29/2006#,"C"
      h = .AddItem("Project 2")
      h1 = .InsertItem(h,0,"Task 1")
      .AddBar h1,"Task",#9/21/2006#,#9/23/2006#,"A"
      h2 = .InsertItem(h,0,"Task 2")
      .AddBar h2,"Task",#9/24/2006#,#9/26/2006#,"B"
      h3 = .InsertItem(h,0,"Task 3")
      .AddBar h3,"Task",#9/27/2006#,#9/29/2006#,"C"
      .ExpandItem(h) = True
   End With
   .EndUpdate
End With
```

The following VB.NET sample shows the child bars for collapsed items:

```
Dim h,h1,h2,h3
With AxG2antt1
   .BeginUpdate
   .LinesAtRoot = EXG2ANTTLib.LinesAtRootEnum.exLinesAtRoot
   .Columns.Add "Tasks"
   With .Chart
```

```
    .FirstVisibleDate = #9/20/2006#
    .ShowCollapsedBars = True
    .LevelCount = 2
    .PaneWidth(0) = 96
  End With
  With .Items
    h = .AddItem("Project 1")
    h1 = .InsertItem(h,0,"Task 1")
    .AddBar h1,"Task",#9/21/2006#,#9/23/2006#,"A"
    h2 = .InsertItem(h,0,"Task 2")
    .AddBar h2,"Task",#9/24/2006#,#9/26/2006#,"B"
    h3 = .InsertItem(h,0,"Task 3")
    .AddBar h3,"Task",#9/27/2006#,#9/29/2006#,"C"
    h = .AddItem("Project 2")
    h1 = .InsertItem(h,0,"Task 1")
    .AddBar h1,"Task",#9/21/2006#,#9/23/2006#,"A"
    h2 = .InsertItem(h,0,"Task 2")
    .AddBar h2,"Task",#9/24/2006#,#9/26/2006#,"B"
    h3 = .InsertItem(h,0,"Task 3")
    .AddBar h3,"Task",#9/27/2006#,#9/29/2006#,"C"
    .ExpandItem(h) = True
  End With
  .EndUpdate
End With
```

The following C++ sample shows the child bars for collapsed items:

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
```

```
spG2antt1->PutLinesAtRoot(EXG2ANTTLib::exLinesAtRoot);
spG2antt1->GetColumns()->Add(L"Tasks");
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
    var_Chart->PutFirstVisibleDate("9/20/2006");
    var_Chart->PutShowCollapsedBars(VARIANT_TRUE);
    var_Chart->PutLevelCount(2);
    var_Chart->PutPaneWidth(0,96);
EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems();
    long h = var_Items->AddItem("Project 1");
    long h1 = var_Items->InsertItem(h,long(0),"Task 1");
    var_Items->AddBar(h1,"Task","9/21/2006","9/23/2006","A",vtMissing);
    long h2 = var_Items->InsertItem(h,long(0),"Task 2");
    var_Items->AddBar(h2,"Task","9/24/2006","9/26/2006","B",vtMissing);
    long h3 = var_Items->InsertItem(h,long(0),"Task 3");
    var_Items->AddBar(h3,"Task","9/27/2006","9/29/2006","C",vtMissing);
    h = var_Items->AddItem("Project 2");
    h1 = var_Items->InsertItem(h,long(0),"Task 1");
    var_Items->AddBar(h1,"Task","9/21/2006","9/23/2006","A",vtMissing);
    h2 = var_Items->InsertItem(h,long(0),"Task 2");
    var_Items->AddBar(h2,"Task","9/24/2006","9/26/2006","B",vtMissing);
    h3 = var_Items->InsertItem(h,long(0),"Task 3");
    var_Items->AddBar(h3,"Task","9/27/2006","9/29/2006","C",vtMissing);
    var_Items->PutExpandItem(h,VARIANT_TRUE);
spG2antt1->EndUpdate();
```

The following C# sample shows the child bars for collapsed items:

```
axG2antt1.BeginUpdate();
axG2antt1.LinesAtRoot = EXG2ANTTLib.LinesAtRootEnum.exLinesAtRoot;
axG2antt1.Columns.Add("Tasks");
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
    var_Chart.FirstVisibleDate = "9/20/2006";
    var_Chart.ShowCollapsedBars = true;
    var_Chart.LevelCount = 2;
    var_Chart.set_PaneWidth(0 != 0,96);
EXG2ANTTLib.Items var_Items = axG2antt1.Items;
    int h = var_Items.AddItem("Project 1");
```

```
    int h1 = var_Items.InsertItem(h,0,"Task 1");
    var_Items.AddBar(h1,"Task","9/21/2006","9/23/2006","A",null);
    int h2 = var_Items.InsertItem(h,0,"Task 2");
    var_Items.AddBar(h2,"Task","9/24/2006","9/26/2006","B",null);
    int h3 = var_Items.InsertItem(h,0,"Task 3");
    var_Items.AddBar(h3,"Task","9/27/2006","9/29/2006","C",null);
    h = var_Items.AddItem("Project 2");
    h1 = var_Items.InsertItem(h,0,"Task 1");
    var_Items.AddBar(h1,"Task","9/21/2006","9/23/2006","A",null);
    h2 = var_Items.InsertItem(h,0,"Task 2");
    var_Items.AddBar(h2,"Task","9/24/2006","9/26/2006","B",null);
    h3 = var_Items.InsertItem(h,0,"Task 3");
    var_Items.AddBar(h3,"Task","9/27/2006","9/29/2006","C",null);
    var_Items.set_ExpandItem(h,true);
axG2antt1.EndUpdate();
```

The following VFP sample shows the child bars for collapsed items:

```
with thisform.G2antt1
    .BeginUpdate
    .LinesAtRoot = -1
    .Columns.Add("Tasks")
    with .Chart
        .FirstVisibleDate = {^2006-9-20}
        .ShowCollapsedBars = .T.
        .LevelCount = 2
        .PaneWidth(0) = 96
    endwith
    with .Items
        h = .AddItem("Project 1")
        h1 = .InsertItem(h,0,"Task 1")
        .AddBar(h1,"Task",{^2006-9-21},{^2006-9-23},"A")
        h2 = .InsertItem(h,0,"Task 2")
        .AddBar(h2,"Task",{^2006-9-24},{^2006-9-26},"B")
        h3 = .InsertItem(h,0,"Task 3")
        .AddBar(h3,"Task",{^2006-9-27},{^2006-9-29},"C")
        h = .AddItem("Project 2")
```

```
    h1 = .InsertItem(h,0,"Task 1")
    .AddBar(h1,"Task",{^2006-9-21},{^2006-9-23},"A")
    h2 = .InsertItem(h,0,"Task 2")
    .AddBar(h2,"Task",{^2006-9-24},{^2006-9-26},"B")
    h3 = .InsertItem(h,0,"Task 3")
    .AddBar(h3,"Task",{^2006-9-27},{^2006-9-29},"C")
    .DefaultItem = h
    .ExpandItem(0) = .T.
  endwith
  .EndUpdate
endwith
```

The following Delphi sample shows the child bars for collapsed items:

```
with AxG2antt1 do
begin
  BeginUpdate();
  LinesAtRoot := EXG2ANTTLib.LinesAtRootEnum.exLinesAtRoot;
  Columns.Add('Tasks');
  with Chart do
  begin
    FirstVisibleDate := '9/20/2006';
    ShowCollapsedBars := True;
    LevelCount := 2;
    PaneWidth[0 <> 0] := 96;
  end;
  with Items do
  begin
    h := AddItem('Project 1');
    h1 := InsertItem(h,TObject(0),'Task 1');
    AddBar(h1,'Task','9/21/2006','9/23/2006','A',Nil);
    h2 := InsertItem(h,TObject(0),'Task 2');
    AddBar(h2,'Task','9/24/2006','9/26/2006','B',Nil);
    h3 := InsertItem(h,TObject(0),'Task 3');
    AddBar(h3,'Task','9/27/2006','9/29/2006','C',Nil);
    h := AddItem('Project 2');
    h1 := InsertItem(h,TObject(0),'Task 1');
```

```
      AddBar(h1,'Task','9/21/2006','9/23/2006','A',Nil);
      h2 := InsertItem(h,TObject(0),'Task 2');
      AddBar(h2,'Task','9/24/2006','9/26/2006','B',Nil);
      h3 := InsertItem(h,TObject(0),'Task 3');
      AddBar(h3,'Task','9/27/2006','9/29/2006','C',Nil);
      ExpandItem[h] := True;
   end;
   EndUpdate();
end
```

property Chart.ShowEmptyBars as Long

Specifies whether empty bars are shown or hidden.

Type	Description
Long	A long expression that specifies the number of time units being added to the end of each bar. An empty bar has the start and end dates identical.

By default, the ShowEmptyBars property is 0. Use the ShowEmptyBars to show the bars, even if the Start and End date are identical. In other words, if this property is 1, the bars will be shown from the start date to end date plus 1 tim-unit, where the time unit is indicated by the ShowEmptyBarsUnit property. For instance, if the ShowEmptyBars property is 1, a task bar from *1/1/2001* to *1/2/2001* shows two days, else if the ShowEmptyBars property is 0, the same task bar highlights only a single day.

Use the AddBar method to assign a bar to an item. Use the ItemBar(exBarStart) and ItemBar(exBarEnd)/ItemBar(exBarEndInclusive) properties to specify the start and end dates for a bar.

!We do not recommend using the ShowEmptyBars property on a non-zero value, if the chart displays bars with ItemBar(exBarKeepWorkingCount) property on False (default). If your chart displays bars with ItemBar(exBarKeepWorkingCount) property on True, you can use the ItemBar(exBarEndInclusive) to display the end date to be last visible date of the bar. In this case, for instance, a a task bar from 1/1/2001 to 1/3/2001 shows two days, the exBarEnd displays 1/3/2001, while the exBarEndInclusive displays 1/2/2001.

The following samples show how to add bars with the same starting and ending point. This is recommended for bars with **NO** exBarKeepWorkingCount. Here you will find the samples for bars with exBarKeepWorkingCount property set.

VBA (MS Access, Excell...)

```
With G2antt1
   .BeginUpdate
   .MarkSearchColumn = False
   With .Columns
      .Add "Tasks"
      .Add("Start").Def(18) = 1
      .Add("End").Def(18) = 543
   End With
   With .Chart
```

```
    .FirstVisibleDate = #9/20/2006#
    .LevelCount = 2
    .PaneWidth(0) = 256
    .ShowEmptyBars = 1
  End With
  With .Items
    .AllowCellValueToItemBar = True
    h = .AddItem("Task 1")
    .AddBar h,"Task",#9/21/2006#,#9/21/2006#
  End With
  .EndUpdate
End With
```

VB6

```
With G2antt1
  .BeginUpdate
  .MarkSearchColumn = False
  With .Columns
    .Add "Tasks"
    .Add("Start").Def(exCellValueToItemBarProperty) = 1
    .Add("End").Def(exCellValueToItemBarProperty) = 543
  End With
  With .Chart
    .FirstVisibleDate = #9/20/2006#
    .LevelCount = 2
    .PaneWidth(0) = 256
    .ShowEmptyBars = 1
  End With
  With .Items
    .AllowCellValueToItemBar = True
    h = .AddItem("Task 1")
    .AddBar h,"Task",#9/21/2006#,#9/21/2006#
  End With
  .EndUpdate
End With
```

VB.NET

```
Dim h
With Exg2antt1
   .BeginUpdate()
   .MarkSearchColumn = False
   With .Columns
      .Add("Tasks")

.Add("Start").set_Def(exontrol.EXG2ANTTLib.DefColumnEnum.exCellValueToItemB



.Add("End").set_Def(exontrol.EXG2ANTTLib.DefColumnEnum.exCellValueToItemBa

   End With
   With .Chart
      .FirstVisibleDate = #9/20/2006#
      .LevelCount = 2
      .set_PaneWidth(False,256)
      .ShowEmptyBars = 1
   End With
   With .Items
      .AllowCellValueToItemBar = True
      h = .AddItem("Task 1")
      .AddBar(h,"Task",#9/21/2006#,#9/21/2006#)
   End With
   .EndUpdate()
End With
```

VB.NET for /COM

```
Dim h
With AxG2antt1
   .BeginUpdate()
   .MarkSearchColumn = False
   With .Columns
      .Add("Tasks")

.Add("Start").Def(EXG2ANTTLib.DefColumnEnum.exCellValueToItemBarProperty)
```

```
= 1

.Add("End").Def(EXG2ANTTLib.DefColumnEnum.exCellValueToItemBarProperty)
= 543
   End With
   With .Chart
      .FirstVisibleDate = #9/20/2006#
      .LevelCount = 2
      .PaneWidth(False) = 256
      .ShowEmptyBars = 1
   End With
   With .Items
      .AllowCellValueToItemBar = True
      h = .AddItem("Task 1")
      .AddBar(h,"Task",#9/21/2006#,#9/21/2006#)
   End With
   .EndUpdate()
End With
```

C++

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control
Library'

   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
spG2antt1->PutMarkSearchColumn(VARIANT_FALSE);
EXG2ANTTLib::IColumnsPtr var_Columns = spG2antt1->GetColumns();
   var_Columns->Add(L"Tasks");
   ((EXG2ANTTLib::IColumnPtr)(var_Columns->Add(L"Start")))-
>PutDef(EXG2ANTTLib::exCellValueToItemBarProperty,long(1));
```

```
   ((EXG2ANTTLib::IColumnPtr)(var_Columns->Add(L"End")))-
>PutDef(EXG2ANTTLib::exCellValueToItemBarProperty,long(543));
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutFirstVisibleDate("9/20/2006");
   var_Chart->PutLevelCount(2);
   var_Chart->PutPaneWidth(VARIANT_FALSE,256);
   var_Chart->PutShowEmptyBars(1);
EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems();
   var_Items->PutAllowCellValueToItemBar(VARIANT_TRUE);
   long h = var_Items->AddItem("Task 1");
   var_Items->AddBar(h,"Task","9/21/2006","9/21/2006",vtMissing,vtMissing);
spG2antt1->EndUpdate();
```

C++ Builder

```
G2antt1->BeginUpdate();
G2antt1->MarkSearchColumn = false;
Exg2anttlib_tlb::IColumnsPtr var_Columns = G2antt1->Columns;
   var_Columns->Add(L"Tasks");
   var_Columns->Add(L"Start")-
>set_Def(Exg2anttlib_tlb::DefColumnEnum::exCellValueToItemBarProperty,TVarian

   var_Columns->Add(L"End")-
>set_Def(Exg2anttlib_tlb::DefColumnEnum::exCellValueToItemBarProperty,TVarian

Exg2anttlib_tlb::IChartPtr var_Chart = G2antt1->Chart;
   var_Chart->set_FirstVisibleDate(TVariant(TDateTime(2006,9,20).operator
double()));
   var_Chart->LevelCount = 2;
   var_Chart->set_PaneWidth(false,256);
   var_Chart->ShowEmptyBars = 1;
Exg2anttlib_tlb::IItemsPtr var_Items = G2antt1->Items;
   var_Items->AllowCellValueToItemBar = true;
   long h = var_Items->AddItem(TVariant("Task 1"));
   var_Items->AddBar(h,TVariant("Task"),TVariant(TDateTime(2006,9,21).operator
double()),TVariant(TDateTime(2006,9,21).operator
```

```
double()),TNoParam(),TNoParam());
G2antt1->EndUpdate();
```

C#

```
exg2antt1.BeginUpdate();
exg2antt1.MarkSearchColumn = false;
exontrol.EXG2ANTTLib.Columns var_Columns = exg2antt1.Columns;
   var_Columns.Add("Tasks");
   (var_Columns.Add("Start") as
exontrol.EXG2ANTTLib.Column).set_Def(exontrol.EXG2ANTTLib.DefColumnEnum.

   (var_Columns.Add("End") as
exontrol.EXG2ANTTLib.Column).set_Def(exontrol.EXG2ANTTLib.DefColumnEnum.

exontrol.EXG2ANTTLib.Chart var_Chart = exg2antt1.Chart;
   var_Chart.FirstVisibleDate =
Convert.ToDateTime("9/20/2006",System.Globalization.CultureInfo.GetCultureInfo
US"));
   var_Chart.LevelCount = 2;
   var_Chart.set_PaneWidth(false,256);
   var_Chart.ShowEmptyBars = 1;
exontrol.EXG2ANTTLib.Items var_Items = exg2antt1.Items;
   var_Items.AllowCellValueToItemBar = true;
   int h = var_Items.AddItem("Task 1");

var_Items.AddBar(h,"Task",Convert.ToDateTime("9/21/2006",System.Globalization.
US")),Convert.ToDateTime("9/21/2006",System.Globalization.CultureInfo.GetCultu
US")),null,null);
exg2antt1.EndUpdate();
```

JavaScript

```
<OBJECT classid="clsid:CD481F4D-2D25-4759-803F-752C568F53B7"
id="G2antt1"></OBJECT>
```

```
<SCRIPT LANGUAGE="JScript">
   G2antt1.BeginUpdate();
   G2antt1.MarkSearchColumn = false;
   var var_Columns = G2antt1.Columns;
      var_Columns.Add("Tasks");
      var_Columns.Add("Start").Def(18) = 1;
      var_Columns.Add("End").Def(18) = 543;
   var var_Chart = G2antt1.Chart;
      var_Chart.FirstVisibleDate = "9/20/2006";
      var_Chart.LevelCount = 2;
      var_Chart.PaneWidth(0) = 256;
      var_Chart.ShowEmptyBars = 1;
   var var_Items = G2antt1.Items;
      var_Items.AllowCellValueToItemBar = true;
      var h = var_Items.AddItem("Task 1");
      var_Items.AddBar(h,"Task","9/21/2006","9/21/2006",null,null);
   G2antt1.EndUpdate();
</SCRIPT>
```

C# for /COM

```
axG2antt1.BeginUpdate();
axG2antt1.MarkSearchColumn = false;
EXG2ANTTLib.Columns var_Columns = axG2antt1.Columns;
   var_Columns.Add("Tasks");
   (var_Columns.Add("Start") as
EXG2ANTTLib.Column).set_Def(EXG2ANTTLib.DefColumnEnum.exCellValueToItem

   (var_Columns.Add("End") as
EXG2ANTTLib.Column).set_Def(EXG2ANTTLib.DefColumnEnum.exCellValueToItem

EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.FirstVisibleDate =
Convert.ToDateTime("9/20/2006",System.Globalization.CultureInfo.GetCultureInfo
US"));
   var_Chart.LevelCount = 2;
   var_Chart.set_PaneWidth(false,256);
```

```
   var_Chart.ShowEmptyBars = 1;
EXG2ANTTLib.Items var_Items = axG2antt1.Items;
   var_Items.AllowCellValueToItemBar = true;
   int h = var_Items.AddItem("Task 1");

var_Items.AddBar(h,"Task",Convert.ToDateTime("9/21/2006",System.Globalization.
US")),Convert.ToDateTime("9/21/2006",System.Globalization.CultureInfo.GetCultu
US")),null,null);
axG2antt1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   COM com_Chart,com_Columns,com_Items;
   anytype var_Chart,var_Columns,var_Items;
   int h;
   ;

   super();

   exg2antt1.BeginUpdate();
   exg2antt1.MarkSearchColumn(false);
   var_Columns = exg2antt1.Columns(); com_Columns = var_Columns;
      com_Columns.Add("Tasks");

COM::createFromVariant(com_Columns.Add("Start")).Def(18/*exCellValueToItemBa



COM::createFromVariant(com_Columns.Add("End")).Def(18/*exCellValueToItemBar

   var_Chart = exg2antt1.Chart(); com_Chart = var_Chart;

com_Chart.FirstVisibleDate(COMVariant::createFromDate(str2Date("9/20/2006",21

      com_Chart.LevelCount(2);
```

```
   /*should be called during the form's activate method*/
com_Chart.PaneWidth(0,256);
      com_Chart.ShowEmptyBars(1);
   var_Items = exg2antt1.Items(); com_Items = var_Items;
      com_Items.AllowCellValueToItemBar(true);
      h = com_Items.AddItem("Task 1");

com_Items.AddBar(h,"Task",COMVariant::createFromDate(str2Date("9/21/2006",21

   exg2antt1.EndUpdate();
}

/*
public void activate(boolean _active)
{
   ;

   super(_active);

   exg2antt1.Chart().PaneWidth(0,256);
}
*/
```

Delphi 8 (.NET only)

```
with AxG2antt1 do
begin
   BeginUpdate();
   MarkSearchColumn := False;
   with Columns do
   begin
      Add('Tasks');
      (Add('Start') as
EXG2ANTTLib.Column).Def[EXG2ANTTLib.DefColumnEnum.exCellValueToItemBarF
 := TObject(1);
      (Add('End') as
EXG2ANTTLib.Column).Def[EXG2ANTTLib.DefColumnEnum.exCellValueToItemBarF
```

```
 := TObject(543);
    end;
    with Chart do
    begin
        FirstVisibleDate := '9/20/2006';
        LevelCount := 2;
        PaneWidth[False] := 256;
        ShowEmptyBars := 1;
    end;
    with Items do
    begin
        AllowCellValueToItemBar := True;
        h := AddItem('Task 1');
        AddBar(h,'Task','9/21/2006','9/21/2006',Nil,Nil);
    end;
    EndUpdate();
end
```

Delphi (standard)

```
with G2antt1 do
begin
    BeginUpdate();
    MarkSearchColumn := False;
    with Columns do
    begin
        Add('Tasks');
        (IUnknown(Add('Start')) as
EXG2ANTTLib_TLB.Column).Def[EXG2ANTTLib_TLB.exCellValueToItemBarProperty]
 := OleVariant(1);
        (IUnknown(Add('End')) as
EXG2ANTTLib_TLB.Column).Def[EXG2ANTTLib_TLB.exCellValueToItemBarProperty]
 := OleVariant(543);
    end;
    with Chart do
    begin
        FirstVisibleDate := '9/20/2006';
```

```
    LevelCount := 2;
    PaneWidth[False] := 256;
    ShowEmptyBars := 1;
  end;
  with Items do
  begin
    AllowCellValueToItemBar := True;
    h := AddItem('Task 1');
    AddBar(h,'Task','9/21/2006','9/21/2006',Null,Null);
  end;
  EndUpdate();
end
```

VFP

```
with thisform.G2antt1
  .BeginUpdate
  .MarkSearchColumn = .F.
  with .Columns
    .Add("Tasks")
    .Add("Start").Def(18) = 1
    .Add("End").Def(18) = 543
  endwith
  with .Chart
    .FirstVisibleDate = {^2006-9-20}
    .LevelCount = 2
    .PaneWidth(0) = 256
    .ShowEmptyBars = 1
  endwith
  with .Items
    .AllowCellValueToItemBar = .T.
    h = .AddItem("Task 1")
    .AddBar(h,"Task",{^2006-9-21},{^2006-9-21})
  endwith
  .EndUpdate
endwith
```

dBASE Plus

```
local h,oG2antt,var_Chart,var_Column,var_Column1,var_Columns,var_Items

oG2antt = form.Activex1.nativeObject
oG2antt.BeginUpdate()
oG2antt.MarkSearchColumn = false
var_Columns = oG2antt.Columns
   var_Columns.Add("Tasks")
   // var_Columns.Add("Start").Def(18) = 1
   var_Column = var_Columns.Add("Start")
   with (oG2antt)
      TemplateDef = [Dim var_Column]
      TemplateDef = var_Column
      Template = [var_Column.Def(18) = 1]
   endwith
   // var_Columns.Add("End").Def(18) = 543
   var_Column1 = var_Columns.Add("End")
   with (oG2antt)
      TemplateDef = [Dim var_Column1]
      TemplateDef = var_Column1
      Template = [var_Column1.Def(18) = 543]
   endwith
var_Chart = oG2antt.Chart
   var_Chart.FirstVisibleDate = "09/20/2006"
   var_Chart.LevelCount = 2
   // var_Chart.PaneWidth(false) = 256
   with (oG2antt)
      TemplateDef = [Dim var_Chart]
      TemplateDef = var_Chart
      Template = [var_Chart.PaneWidth(false) = 256]
   endwith
   var_Chart.ShowEmptyBars = 1
var_Items = oG2antt.Items
   var_Items.AllowCellValueToItemBar = true
   h = var_Items.AddItem("Task 1")
   var_Items.AddBar(h,"Task","09/21/2006","09/21/2006")
oG2antt.EndUpdate()
```

XBasic (Alpha Five)

```
Dim h as N
Dim oG2antt as P
Dim var_Chart as P
Dim var_Column as P
Dim var_Column1 as P
Dim var_Columns as P
Dim var_Items as P

oG2antt = topparent:CONTROL_ACTIVEX1.activex
oG2antt.BeginUpdate()
oG2antt.MarkSearchColumn = .f.
var_Columns = oG2antt.Columns
   var_Columns.Add("Tasks")
   ' var_Columns.Add("Start").Def(18) = 1
   var_Column = var_Columns.Add("Start")
   oG2antt.TemplateDef = "Dim var_Column"
   oG2antt.TemplateDef = var_Column
   oG2antt.Template = "var_Column.Def(18) = 1"

   ' var_Columns.Add("End").Def(18) = 543
   var_Column1 = var_Columns.Add("End")
   oG2antt.TemplateDef = "Dim var_Column1"
   oG2antt.TemplateDef = var_Column1
   oG2antt.Template = "var_Column1.Def(18) = 543"

var_Chart = oG2antt.Chart
   var_Chart.FirstVisibleDate = {09/20/2006}
   var_Chart.LevelCount = 2
   ' var_Chart.PaneWidth(.f.) = 256
   oG2antt.TemplateDef = "Dim var_Chart"
   oG2antt.TemplateDef = var_Chart
   oG2antt.Template = "var_Chart.PaneWidth(False) = 256"

   var_Chart.ShowEmptyBars = 1
var_Items = oG2antt.Items
```

```
   var_Items.AllowCellValueToItemBar = .t.
   h = var_Items.AddItem("Task 1")
   var_Items.AddBar(h,"Task",{09/21/2006},{09/21/2006})
oG2antt.EndUpdate()
```

Visual Objects

```
local var_Chart as IChart
local var_Columns as IColumns
local var_Items as IItems
local h as USUAL

oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:MarkSearchColumn := false
var_Columns := oDCOCX_Exontrol1:Columns
   var_Columns:Add("Tasks")
   IColumn{var_Columns:Add("Start")}:[Def,exCellValueToItemBarProperty] := 1
   IColumn{var_Columns:Add("End")}:[Def,exCellValueToItemBarProperty] := 543
var_Chart := oDCOCX_Exontrol1:Chart
   var_Chart:FirstVisibleDate := SToD("20060920")
   var_Chart:LevelCount := 2
   var_Chart:[PaneWidth,false] := 256
   var_Chart:ShowEmptyBars := 1
var_Items := oDCOCX_Exontrol1:Items
   var_Items:AllowCellValueToItemBar := true
   h := var_Items:AddItem("Task 1")
   var_Items:AddBar(h,"Task",SToD("20060921"),SToD("20060921"),nil,nil)
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
OleObject oG2antt,var_Chart,var_Columns,var_Items
any h

oG2antt = ole_1.Object
oG2antt.BeginUpdate()
```

```
oG2antt.MarkSearchColumn = false
var_Columns = oG2antt.Columns
   var_Columns.Add("Tasks")
   var_Columns.Add("Start").Def(18,1)
   var_Columns.Add("End").Def(18,543)
var_Chart = oG2antt.Chart
   var_Chart.FirstVisibleDate = 2006-09-20
   var_Chart.LevelCount = 2
   var_Chart.PaneWidth(false,256)
   var_Chart.ShowEmptyBars = 1
var_Items = oG2antt.Items
   var_Items.AllowCellValueToItemBar = true
   h = var_Items.AddItem("Task 1")
   var_Items.AddBar(h,"Task",2006-09-21,2006-09-21)
oG2antt.EndUpdate()
```

The following samples show how to add bars with the same starting and ending point. This is recommended for bars with exBarKeepWorkingCount property set.

VBA (MS Access, Excell...)

```
With G2antt1
   .BeginUpdate
   .MarkSearchColumn = False
   With .Columns
      .Add "Tasks"
      .Add("Start").Def(18) = 1
      .Add("End").Def(18) = 543
   End With
   With .Chart
      .FirstVisibleDate = #9/20/2006#
      .LevelCount = 2
      .PaneWidth(0) = 256
      .ShowEmptyBars = 0
   End With
   With .Items
      .AllowCellValueToItemBar = True
      h = .AddItem("Task 1")
```

```
    .AddBar h,"Task",#9/21/2006#,#9/21/2006#
    .ItemBar(h,"",543) = .ItemBar(h,"",1)
    .ItemBar(h,"",20) = True
  End With
  .EndUpdate
End With
```

VB6

```
With G2antt1
  .BeginUpdate
  .MarkSearchColumn = False
  With .Columns
    .Add "Tasks"
    .Add("Start").Def(exCellValueToItemBarProperty) = 1
    .Add("End").Def(exCellValueToItemBarProperty) = 543
  End With
  With .Chart
    .FirstVisibleDate = #9/20/2006#
    .LevelCount = 2
    .PaneWidth(0) = 256
    .ShowEmptyBars = 0
  End With
  With .Items
    .AllowCellValueToItemBar = True
    h = .AddItem("Task 1")
    .AddBar h,"Task",#9/21/2006#,#9/21/2006#
    .ItemBar(h,"",exBarEndInclusive) = .ItemBar(h,"",exBarStart)
    .ItemBar(h,"",exBarKeepWorkingCount) = True
  End With
  .EndUpdate
End With
```

VB.NET

```
Dim h
With Exg2antt1
  .BeginUpdate()
```

```
   .MarkSearchColumn = False
   With .Columns
      .Add("Tasks")

.Add("Start").set_Def(exontrol.EXG2ANTTLib.DefColumnEnum.exCellValueToItemB



.Add("End").set_Def(exontrol.EXG2ANTTLib.DefColumnEnum.exCellValueToItemBa

   End With
   With .Chart
      .FirstVisibleDate = #9/20/2006#
      .LevelCount = 2
      .set_PaneWidth(False,256)
      .ShowEmptyBars = 0
   End With
   With .Items
      .AllowCellValueToItemBar = True
      h = .AddItem("Task 1")
      .AddBar(h,"Task",#9/21/2006#,#9/21/2006#)

.set_ItemBar(h,"",exontrol.EXG2ANTTLib.ItemBarPropertyEnum.exBarEndInclusive



.set_ItemBar(h,"",exontrol.EXG2ANTTLib.ItemBarPropertyEnum.exBarKeepWorkin

   End With
   .EndUpdate()
End With
```

VB.NET for /COM

```
Dim h
With AxG2antt1
   .BeginUpdate()
   .MarkSearchColumn = False
   With .Columns
```

```
        .Add("Tasks")

.Add("Start").Def(EXG2ANTTLib.DefColumnEnum.exCellValueToItemBarProperty)
= 1

.Add("End").Def(EXG2ANTTLib.DefColumnEnum.exCellValueToItemBarProperty)
= 543
    End With
    With .Chart
        .FirstVisibleDate = #9/20/2006#
        .LevelCount = 2
        .PaneWidth(False) = 256
        .ShowEmptyBars = 0
    End With
    With .Items
        .AllowCellValueToItemBar = True
        h = .AddItem("Task 1")
        .AddBar(h,"Task",#9/21/2006#,#9/21/2006#)
        .ItemBar(h,"",EXG2ANTTLib.ItemBarPropertyEnum.exBarEndInclusive) =
.ItemBar(h,"",EXG2ANTTLib.ItemBarPropertyEnum.exBarStart)

.ItemBar(h,"",EXG2ANTTLib.ItemBarPropertyEnum.exBarKeepWorkingCount) =
True
    End With
    .EndUpdate()
End With
```

C++

```
/*
    Copy and paste the following directives to your header file as
    it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control
Library'

    #import <ExG2antt.dll>
    using namespace EXG2ANTTLib;
*/
```

```
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)->GetControlUnknown();
spG2antt1->BeginUpdate();
spG2antt1->PutMarkSearchColumn(VARIANT_FALSE);
EXG2ANTTLib::IColumnsPtr var_Columns = spG2antt1->GetColumns();
   var_Columns->Add(L"Tasks");
   ((EXG2ANTTLib::IColumnPtr)(var_Columns->Add(L"Start")))->PutDef(EXG2ANTTLib::exCellValueToItemBarProperty,long(1));
   ((EXG2ANTTLib::IColumnPtr)(var_Columns->Add(L"End")))->PutDef(EXG2ANTTLib::exCellValueToItemBarProperty,long(543));
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutFirstVisibleDate("9/20/2006");
   var_Chart->PutLevelCount(2);
   var_Chart->PutPaneWidth(VARIANT_FALSE,256);
   var_Chart->PutShowEmptyBars(0);
EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems();
   var_Items->PutAllowCellValueToItemBar(VARIANT_TRUE);
   long h = var_Items->AddItem("Task 1");
   var_Items->AddBar(h,"Task","9/21/2006","9/21/2006",vtMissing,vtMissing);
   var_Items->PutItemBar(h,"",EXG2ANTTLib::exBarEndInclusive,var_Items->GetItemBar(h,"",EXG2ANTTLib::exBarStart));
   var_Items->PutItemBar(h,"",EXG2ANTTLib::exBarKeepWorkingCount,VARIANT_TRUE);
spG2antt1->EndUpdate();
```

C++ Builder

```
G2antt1->BeginUpdate();
G2antt1->MarkSearchColumn = false;
Exg2anttlib_tlb::IColumnsPtr var_Columns = G2antt1->Columns;
   var_Columns->Add(L"Tasks");
   var_Columns->Add(L"Start")->set_Def(Exg2anttlib_tlb::DefColumnEnum::exCellValueToItemBarProperty,TVarian

   var_Columns->Add(L"End")->set_Def(Exg2anttlib_tlb::DefColumnEnum::exCellValueToItemBarProperty,TVarian
```

```
Exg2anttlib_tlb::IChartPtr var_Chart = G2antt1->Chart;
   var_Chart->set_FirstVisibleDate(TVariant(TDateTime(2006,9,20).operator
double()));
   var_Chart->LevelCount = 2;
   var_Chart->set_PaneWidth(false,256);
   var_Chart->ShowEmptyBars = 0;
Exg2anttlib_tlb::IItemsPtr var_Items = G2antt1->Items;
   var_Items->AllowCellValueToItemBar = true;
   long h = var_Items->AddItem(TVariant("Task 1"));
   var_Items->AddBar(h,TVariant("Task"),TVariant(TDateTime(2006,9,21).operator
double()),TVariant(TDateTime(2006,9,21).operator
double()),TNoParam(),TNoParam());
   var_Items-
>set_ItemBar(h,TVariant(""),Exg2anttlib_tlb::ItemBarPropertyEnum::exBarEndInclu
>get_ItemBar(h,TVariant(""),Exg2anttlib_tlb::ItemBarPropertyEnum::exBarStart)));
   var_Items-
>set_ItemBar(h,TVariant(""),Exg2anttlib_tlb::ItemBarPropertyEnum::exBarKeepWo

G2antt1->EndUpdate();
```

C#

```
exg2antt1.BeginUpdate();
exg2antt1.MarkSearchColumn = false;
exontrol.EXG2ANTTLib.Columns var_Columns = exg2antt1.Columns;
   var_Columns.Add("Tasks");
   (var_Columns.Add("Start") as
exontrol.EXG2ANTTLib.Column).set_Def(exontrol.EXG2ANTTLib.DefColumnEnum.

   (var_Columns.Add("End") as
exontrol.EXG2ANTTLib.Column).set_Def(exontrol.EXG2ANTTLib.DefColumnEnum.

exontrol.EXG2ANTTLib.Chart var_Chart = exg2antt1.Chart;
   var_Chart.FirstVisibleDate =
Convert.ToDateTime("9/20/2006",System.Globalization.CultureInfo.GetCultureInfo
```

```
US"));
   var_Chart.LevelCount = 2;
   var_Chart.set_PaneWidth(false,256);
   var_Chart.ShowEmptyBars = 0;
exontrol.EXG2ANTTLib.Items var_Items = exg2antt1.Items;
   var_Items.AllowCellValueToItemBar = true;
   int h = var_Items.AddItem("Task 1");

var_Items.AddBar(h,"Task",Convert.ToDateTime("9/21/2006",System.Globalization.
US")),Convert.ToDateTime("9/21/2006",System.Globalization.CultureInfo.GetCultu
US")),null,null);

var_Items.set_ItemBar(h,"",exontrol.EXG2ANTTLib.ItemBarPropertyEnum.exBarEnd



var_Items.set_ItemBar(h,"",exontrol.EXG2ANTTLib.ItemBarPropertyEnum.exBarKee

exg2antt1.EndUpdate();
```

JavaScript

```
<OBJECT classid="clsid:CD481F4D-2D25-4759-803F-752C568F53B7"
id="G2antt1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
   G2antt1.BeginUpdate();
   G2antt1.MarkSearchColumn = false;
   var var_Columns = G2antt1.Columns;
      var_Columns.Add("Tasks");
      var_Columns.Add("Start").Def(18) = 1;
      var_Columns.Add("End").Def(18) = 543;
   var var_Chart = G2antt1.Chart;
      var_Chart.FirstVisibleDate = "9/20/2006";
      var_Chart.LevelCount = 2;
      var_Chart.PaneWidth(0) = 256;
      var_Chart.ShowEmptyBars = 0;
```

```
   var var_Items = G2antt1.Items;
      var_Items.AllowCellValueToItemBar = true;
      var h = var_Items.AddItem("Task 1");
      var_Items.AddBar(h,"Task","9/21/2006","9/21/2006",null,null);
      var_Items.ItemBar(h,"",543) = var_Items.ItemBar(h,"",1);
      var_Items.ItemBar(h,"",20) = true;
   G2antt1.EndUpdate();
</SCRIPT>
```

C# for /COM

```
axG2antt1.BeginUpdate();
axG2antt1.MarkSearchColumn = false;
EXG2ANTTLib.Columns var_Columns = axG2antt1.Columns;
   var_Columns.Add("Tasks");
   (var_Columns.Add("Start") as
EXG2ANTTLib.Column).set_Def(EXG2ANTTLib.DefColumnEnum.exCellValueToItem

   (var_Columns.Add("End") as
EXG2ANTTLib.Column).set_Def(EXG2ANTTLib.DefColumnEnum.exCellValueToItem

EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.FirstVisibleDate =
Convert.ToDateTime("9/20/2006",System.Globalization.CultureInfo.GetCultureInfo
US"));
   var_Chart.LevelCount = 2;
   var_Chart.set_PaneWidth(false,256);
   var_Chart.ShowEmptyBars = 0;
EXG2ANTTLib.Items var_Items = axG2antt1.Items;
   var_Items.AllowCellValueToItemBar = true;
   int h = var_Items.AddItem("Task 1");

var_Items.AddBar(h,"Task",Convert.ToDateTime("9/21/2006",System.Globalization.
US")),Convert.ToDateTime("9/21/2006",System.Globalization.CultureInfo.GetCultu
US")),null,null);

var_Items.set_ItemBar(h,"",EXG2ANTTLib.ItemBarPropertyEnum.exBarEndInclusive
```

```
var_Items.set_ItemBar(h,"",EXG2ANTTLib.ItemBarPropertyEnum.exBarKeepWorkir

axG2antt1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    COM com_Chart,com_Columns,com_Items;
    anytype var_Chart,var_Columns,var_Items;
    int h;
    ;

    super();

    exg2antt1.BeginUpdate();
    exg2antt1.MarkSearchColumn(false);
    var_Columns = exg2antt1.Columns(); com_Columns = var_Columns;
        com_Columns.Add("Tasks");

COM::createFromVariant(com_Columns.Add("Start")).Def(18/*exCellValueToItemBa



COM::createFromVariant(com_Columns.Add("End")).Def(18/*exCellValueToItemBar

    var_Chart = exg2antt1.Chart(); com_Chart = var_Chart;

com_Chart.FirstVisibleDate(COMVariant::createFromDate(str2Date("9/20/2006",21

        com_Chart.LevelCount(2);
    /*should be called during the form's activate method*/
com_Chart.PaneWidth(0,256);
        com_Chart.ShowEmptyBars(0);
    var_Items = exg2antt1.Items(); com_Items = var_Items;
```

```
        com_Items.AllowCellValueToItemBar(true);
        h = com_Items.AddItem("Task 1");

com_Items.AddBar(h,"Task",COMVariant::createFromDate(str2Date("9/21/2006",21


com_Items.ItemBar(h,"",543/*exBarEndInclusive*/,com_Items.ItemBar(h,"",1/*exBa


com_Items.ItemBar(h,"",20/*exBarKeepWorkingCount*/,COMVariant::createFromB

    exg2antt1.EndUpdate();
}

/*
public void activate(boolean _active)
{
    ;

    super(_active);

    exg2antt1.Chart().PaneWidth(0,256);
}
*/
```

Delphi 8 (.NET only)

```
with AxG2antt1 do
begin
    BeginUpdate();
    MarkSearchColumn := False;
    with Columns do
    begin
        Add('Tasks');
        (Add('Start') as
EXG2ANTTLib.Column).Def[EXG2ANTTLib.DefColumnEnum.exCellValueToItemBarF
 := TObject(1);
```

```
      (Add('End') as
EXG2ANTTLib.Column).Def[EXG2ANTTLib.DefColumnEnum.exCellValueToItemBarF
 := TObject(543);
   end;
   with Chart do
   begin
      FirstVisibleDate := '9/20/2006';
      LevelCount := 2;
      PaneWidth[False] := 256;
      ShowEmptyBars := 0;
   end;
   with Items do
   begin
      AllowCellValueToItemBar := True;
      h := AddItem('Task 1');
      AddBar(h,'Task','9/21/2006','9/21/2006',Nil,Nil);
      ItemBar[h,'',EXG2ANTTLib.ItemBarPropertyEnum.exBarEndInclusive] :=
ItemBar[h,'',EXG2ANTTLib.ItemBarPropertyEnum.exBarStart];
      ItemBar[h,'',EXG2ANTTLib.ItemBarPropertyEnum.exBarKeepWorkingCount]
:= TObject(True);
   end;
   EndUpdate();
end
```

Delphi (standard)

```
with G2antt1 do
begin
   BeginUpdate();
   MarkSearchColumn := False;
   with Columns do
   begin
      Add('Tasks');
      (IUnknown(Add('Start')) as
EXG2ANTTLib_TLB.Column).Def[EXG2ANTTLib_TLB.exCellValueToItemBarProperty]
 := OleVariant(1);
      (IUnknown(Add('End')) as
```

```
EXG2ANTTLib_TLB.Column).Def[EXG2ANTTLib_TLB.exCellValueToItemBarProperty]
 := OleVariant(543);
   end;
   with Chart do
   begin
      FirstVisibleDate := '9/20/2006';
      LevelCount := 2;
      PaneWidth[False] := 256;
      ShowEmptyBars := 0;
   end;
   with Items do
   begin
      AllowCellValueToItemBar := True;
      h := AddItem('Task 1');
      AddBar(h,'Task','9/21/2006','9/21/2006',Null,Null);
      ItemBar[h,'',EXG2ANTTLib_TLB.exBarEndInclusive] :=
ItemBar[h,'',EXG2ANTTLib_TLB.exBarStart];
      ItemBar[h,'',EXG2ANTTLib_TLB.exBarKeepWorkingCount] :=
OleVariant(True);
   end;
   EndUpdate();
end
```

VFP

```
with thisform.G2antt1
   .BeginUpdate
   .MarkSearchColumn = .F.
   with .Columns
      .Add("Tasks")
      .Add("Start").Def(18) = 1
      .Add("End").Def(18) = 543
   endwith
   with .Chart
      .FirstVisibleDate = {^2006-9-20}
      .LevelCount = 2
      .PaneWidth(0) = 256
```

```
    .ShowEmptyBars = 0
  endwith
  with .Items
    .AllowCellValueToItemBar = .T.
    h = .AddItem("Task 1")
    .AddBar(h,"Task",{^2006-9-21},{^2006-9-21})
    .ItemBar(h,"",543) = .ItemBar(h,"",1)
    .ItemBar(h,"",20) = .T.
  endwith
  .EndUpdate
endwith
```

dBASE Plus

```
local h,oG2antt,var_Chart,var_Column,var_Column1,var_Columns,var_Items

oG2antt = form.Activex1.nativeObject
oG2antt.BeginUpdate()
oG2antt.MarkSearchColumn = false
var_Columns = oG2antt.Columns
  var_Columns.Add("Tasks")
  // var_Columns.Add("Start").Def(18) = 1
  var_Column = var_Columns.Add("Start")
  with (oG2antt)
    TemplateDef = [Dim var_Column]
    TemplateDef = var_Column
    Template = [var_Column.Def(18) = 1]
  endwith
  // var_Columns.Add("End").Def(18) = 543
  var_Column1 = var_Columns.Add("End")
  with (oG2antt)
    TemplateDef = [Dim var_Column1]
    TemplateDef = var_Column1
    Template = [var_Column1.Def(18) = 543]
  endwith
var_Chart = oG2antt.Chart
  var_Chart.FirstVisibleDate = "09/20/2006"
```

```
   var_Chart.LevelCount = 2
   // var_Chart.PaneWidth(false) = 256
   with (oG2antt)
      TemplateDef = [Dim var_Chart]
      TemplateDef = var_Chart
      Template = [var_Chart.PaneWidth(false) = 256]
   endwith
   var_Chart.ShowEmptyBars = 0
var_Items = oG2antt.Items
   var_Items.AllowCellValueToItemBar = true
   h = var_Items.AddItem("Task 1")
   var_Items.AddBar(h,"Task","09/21/2006","09/21/2006")
   // var_Items.ItemBar(h,"",543) = var_Items.ItemBar(h,"",1)
   with (oG2antt)
      TemplateDef = [Dim var_Items,h]
      TemplateDef = var_Items
      TemplateDef = h
      Template = [var_Items.ItemBar(h,"",543) = var_Items.ItemBar(h,"",1)]
   endwith
   // var_Items.ItemBar(h,"",20) = true
   with (oG2antt)
      TemplateDef = [Dim var_Items,h]
      TemplateDef = var_Items
      TemplateDef = h
      Template = [var_Items.ItemBar(h,"",20) = true]
   endwith
oG2antt.EndUpdate()
```

XBasic (Alpha Five)

```
Dim h as N
Dim oG2antt as P
Dim var_Chart as P
Dim var_Column as P
Dim var_Column1 as P
Dim var_Columns as P
```

```
Dim var_Items as P

oG2antt = topparent:CONTROL_ACTIVEX1.activex
oG2antt.BeginUpdate()
oG2antt.MarkSearchColumn = .f.
var_Columns = oG2antt.Columns
   var_Columns.Add("Tasks")
   ' var_Columns.Add("Start").Def(18) = 1
   var_Column = var_Columns.Add("Start")
   oG2antt.TemplateDef = "Dim var_Column"
   oG2antt.TemplateDef = var_Column
   oG2antt.Template = "var_Column.Def(18) = 1"

   ' var_Columns.Add("End").Def(18) = 543
   var_Column1 = var_Columns.Add("End")
   oG2antt.TemplateDef = "Dim var_Column1"
   oG2antt.TemplateDef = var_Column1
   oG2antt.Template = "var_Column1.Def(18) = 543"

var_Chart = oG2antt.Chart
   var_Chart.FirstVisibleDate = {09/20/2006}
   var_Chart.LevelCount = 2
   ' var_Chart.PaneWidth(.f.) = 256
   oG2antt.TemplateDef = "Dim var_Chart"
   oG2antt.TemplateDef = var_Chart
   oG2antt.Template = "var_Chart.PaneWidth(False) = 256"

   var_Chart.ShowEmptyBars = 0
var_Items = oG2antt.Items
   var_Items.AllowCellValueToItemBar = .t.
   h = var_Items.AddItem("Task 1")
   var_Items.AddBar(h,"Task",{09/21/2006},{09/21/2006})
   ' var_Items.ItemBar(h,"",543) = var_Items.ItemBar(h,"",1)
   oG2antt.TemplateDef = "Dim var_Items,h"
   oG2antt.TemplateDef = var_Items
   oG2antt.TemplateDef = h
   oG2antt.Template = "var_Items.ItemBar(h,\"\",543) =
```

```
var_Items.ItemBar(h,\"\",1)"

   ' var_Items.ItemBar(h,"",20) = .t.
   oG2antt.TemplateDef = "Dim var_Items,h"
   oG2antt.TemplateDef = var_Items
   oG2antt.TemplateDef = h
   oG2antt.Template = "var_Items.ItemBar(h,\"\",20) = True"

oG2antt.EndUpdate()
```

Visual Objects

```
local var_Chart as IChart
local var_Columns as IColumns
local var_Items as IItems
local h as USUAL

oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:MarkSearchColumn := false
var_Columns := oDCOCX_Exontrol1:Columns
   var_Columns:Add("Tasks")
   IColumn{var_Columns:Add("Start")}:[Def,exCellValueToItemBarProperty] := 1
   IColumn{var_Columns:Add("End")}:[Def,exCellValueToItemBarProperty] := 543
var_Chart := oDCOCX_Exontrol1:Chart
   var_Chart:FirstVisibleDate := SToD("20060920")
   var_Chart:LevelCount := 2
   var_Chart:[PaneWidth,false] := 256
   var_Chart:ShowEmptyBars := 0
var_Items := oDCOCX_Exontrol1:Items
   var_Items:AllowCellValueToItemBar := true
   h := var_Items:AddItem("Task 1")
   var_Items:AddBar(h,"Task",SToD("20060921"),SToD("20060921"),nil,nil)
   var_Items:[ItemBar,h,"",exBarEndInclusive] := var_Items:[ItemBar,h,"",exBarStart]
   var_Items:[ItemBar,h,"",exBarKeepWorkingCount] := true
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
OleObject oG2antt,var_Chart,var_Columns,var_Items
any h

oG2antt = ole_1.Object
oG2antt.BeginUpdate()
oG2antt.MarkSearchColumn = false
var_Columns = oG2antt.Columns
   var_Columns.Add("Tasks")
   var_Columns.Add("Start").Def(18,1)
   var_Columns.Add("End").Def(18,543)
var_Chart = oG2antt.Chart
   var_Chart.FirstVisibleDate = 2006-09-20
   var_Chart.LevelCount = 2
   var_Chart.PaneWidth(false,256)
   var_Chart.ShowEmptyBars = 0
var_Items = oG2antt.Items
   var_Items.AllowCellValueToItemBar = true
   h = var_Items.AddItem("Task 1")
   var_Items.AddBar(h,"Task",2006-09-21,2006-09-21)
   var_Items.ItemBar(h,"",543,var_Items.ItemBar(h,"",1))
   var_Items.ItemBar(h,"",20,true)
oG2antt.EndUpdate()
```

property Chart.ShowEmptyBarsUnit as UnitEnum

Specifies the unit to be added to the end date, so empty bars are shown.

Type	Description
UnitEnum	An UnitEnum expression that indicates the time unit being added to each bar, when the ShowEmptyBars property is not zero.

By default, the ShowEmptyBarsUnit property is exDay. This property has effect only, if the ShowEmptyBars property is not zero. For instance, if your chart displays seconds, the ShowEmptyBarsUnit property must be set on exSeconds, else else if the ShowEmptyBars property is 1, the ending date for each bar is not show correctly, as 1 day is added to a second. For instance, if the ShowEmptyBars property is 1 and ShowEmptyBarsUnit is exDay, a task bar from *1/1/2001* to *1/2/2001* shows two days, else if the ShowEmptyBars property is 0, the same task bar highlights only a single day. Use the AddBar method to assign a bar to an item. Use the ItemBar(exBarStart) and ItemBar(exBarEnd) properties to specify the start and end dates for a bar.

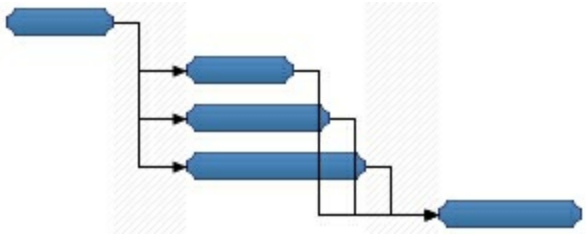
property Chart.ShowLinks as ShowExtendedLinksEnum

Retrieves or sets a value that indicates whether the links between bars are visible or hidden.

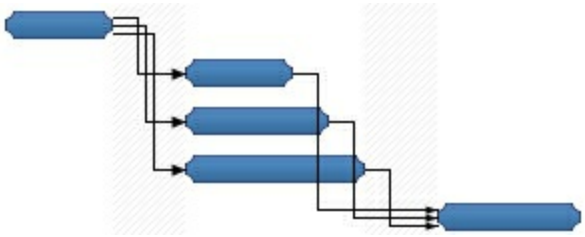
Type	Description
ShowExtendedLinksEnum	A Boolean/ShowExtendedLinksEnum expression that indicates whether the chart shows the lines between bars.

By default, the ShowLinks property is True. Use the ShowLinks property to hide all links between bars. Use the Link(exLinkVisible) property to hide a specific link between two bars. Use the LinkColor property to specify the color for all links in the chart area. Use the LinkStyle property to specify the style for all links in the chart area. Use the LinkWidth property to specify the width of the pen, in pixels, to draw the links between bars. Use the AddLink method to link a bar with another. Use the Link(exLinkShowDir) property to hide the arrow that indicates the direction of the link. Use the FirstLink and NextLink properties to enumerate the links in the control. Use the AllowLinkBars property to specify whether the user can link the bars using the mouse. The ShowLinksColor property specifies the color to display the links when link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. The ShowLinksWidth property specifies the width to show the links when the link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. The ShowLinksStyle property specifies the width to show the links when the link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars.

The following screen shot shows the default links:



The following screen shot shows the extended links:



The extended links are shown when two or more links starts or ends on the same bar, so they will be shown distinctly, rather than showing them one over another.

property Chart.ShowLinksColor(Links as ShowLinksEnum) as Color

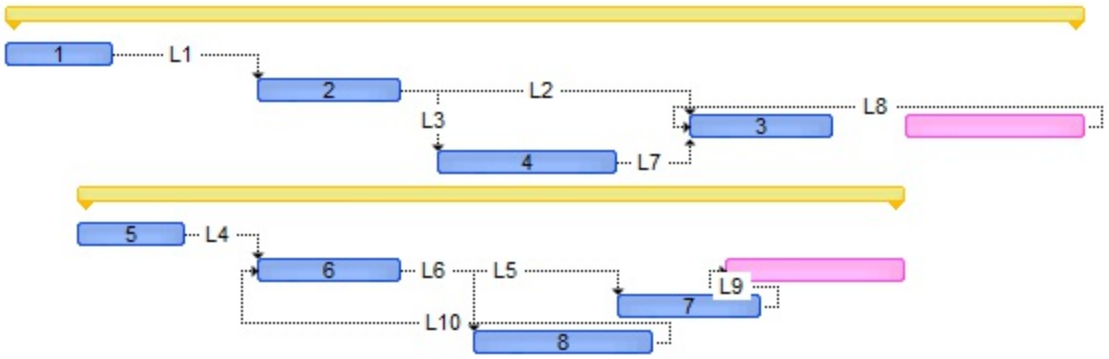
Retrieves or sets a value that indicates the color to display the links based on the user selection.

Type	Description
Links as ShowLinksEnum	A ShowLinksEnum expression that specifies the links color being accessed.
Color	A Color expression that specifies the color to display the links.

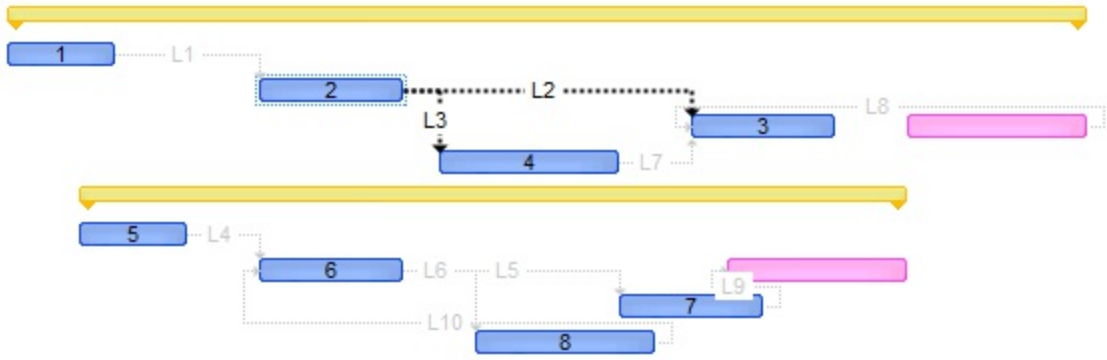
By default the ShowLinksColor property is 0, which means that it has no effect. The ShowLinksColor property specifies the color to display the links when link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. The ShowLinksWidth property specifies the width to show the links when the link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. The ShowLinksStyle property specifies the width to show the links when the link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. ***You can hide the links if the ShowLinksColor property is the same as the chart's background color being specified by the BackColor property.***

The ShowLinks property specifies whether the chart's links are visible or hidden. Use the LinkColor property to specify the color for all links in the chart area. Use the LinkStyle property to specify the style for all links in the chart area. Use the LinkWidth property to specify the width of the pen, in pixels, to draw the links between bars. Use the AddLink method to add programmatically a link a bar with another.

The following screen shows the links when no bar is selected:



The following screen shows the links when the bar 2 is selected:



The following VB sample makes the links that starts from selected bar being wider, while the rest being transparent:

```
With G2antt1
   .BeginUpdate
   .Columns.Add "Tasks"
   With .Chart
      .FirstVisibleDate = #9/20/2006#
      .PaneWidth(0) = 64
      .ShowLinksWidth(exShowLinksStartFrom) = 2
      .ShowLinksStyle(exShowLinksStartFrom) = exLinkDot
      .ShowLinksColor(exShowLinksEndTo) = RGB(200,200,200)
      .ShowLinksColor(exShowUnselectedLinks) = RGB(200,200,200)
   End With
   With .Items
      h1 = .AddItem("Task 1")
      .AddBar h1,"Task",#9/21/2006#,#9/23/2006#
      h2 = .AddItem("Task 2")
      .AddBar h2,"Task",#9/25/2006#,#9/27/2006#
      .ItemBar(h2,"",exBarSelected) = True
      .AddLink "L1",h1,"",h2,""
      .Link("L1",exLinkText) = "L1"
      h3 = .AddItem("Task 3")
      .AddBar h3,"Task",#9/29/2006#,#10/2/2006#
      .AddLink "L2",h2,"",h3,""
      .Link("L2",exLinkText) = "L2"
   End With
   .EndUpdate
End With
```

The following VB.NET sample makes the links that starts from selected bar being wider, while the rest being transparent:

```
Dim h1,h2,h3
With AxG2antt1
   .BeginUpdate
   .Columns.Add "Tasks"
   With .Chart
      .FirstVisibleDate = #9/20/2006#
      .PaneWidth(0) = 64
      .ShowLinksWidth(EXG2ANTTLib.ShowLinksEnum.exShowLinksStartFrom) = 2
      .ShowLinksStyle(EXG2ANTTLib.ShowLinksEnum.exShowLinksStartFrom) =
EXG2ANTTLib.LinkStyleEnum.exLinkDot
      .ShowLinksColor(EXG2ANTTLib.ShowLinksEnum.exShowLinksEndTo) = 13158600
      .ShowLinksColor(EXG2ANTTLib.ShowLinksEnum.exShowUnselectedLinks) =
13158600
   End With
   With .Items
      h1 = .AddItem("Task 1")
      .AddBar h1,"Task",#9/21/2006#,#9/23/2006#
      h2 = .AddItem("Task 2")
      .AddBar h2,"Task",#9/25/2006#,#9/27/2006#
      .ItemBar(h2,"",EXG2ANTTLib.ItemBarPropertyEnum.exBarSelected) = True
      .AddLink "L1",h1,"",h2,""
      .Link("L1",EXG2ANTTLib.LinkPropertyEnum.exLinkText) = "L1"
      h3 = .AddItem("Task 3")
      .AddBar h3,"Task",#9/29/2006#,#10/2/2006#
      .AddLink "L2",h2,"",h3,""
      .Link("L2",EXG2ANTTLib.LinkPropertyEnum.exLinkText) = "L2"
   End With
   .EndUpdate
End With
```

The following C++ sample makes the links that starts from selected bar being wider, while the rest being transparent:

```
/*
   Copy and paste the following directives to your header file as
```

```
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
spG2antt1->GetColumns()->Add(L"Tasks");
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutFirstVisibleDate("9/20/2006");
   var_Chart->PutPaneWidth(0,64);
   var_Chart->PutShowLinksWidth(EXG2ANTTLib::exShowLinksStartFrom,2);
   var_Chart-
>PutShowLinksStyle(EXG2ANTTLib::exShowLinksStartFrom,EXG2ANTTLib::exLinkDot);
   var_Chart->PutShowLinksColor(EXG2ANTTLib::exShowLinksEndTo,RGB(200,200,200));
   var_Chart-
>PutShowLinksColor(EXG2ANTTLib::exShowUnselectedLinks,RGB(200,200,200));
EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems();
   long h1 = var_Items->AddItem("Task 1");
   var_Items->AddBar(h1,"Task","9/21/2006","9/23/2006",vtMissing,vtMissing);
   long h2 = var_Items->AddItem("Task 2");
   var_Items->AddBar(h2,"Task","9/25/2006","9/27/2006",vtMissing,vtMissing);
   var_Items->PutItemBar(h2,"",EXG2ANTTLib::exBarSelected,VARIANT_TRUE);
   var_Items->AddLink("L1",h1,"",h2,"");
   var_Items->PutLink("L1",EXG2ANTTLib::exLinkText,"L1");
   long h3 = var_Items->AddItem("Task 3");
   var_Items->AddBar(h3,"Task","9/29/2006","10/2/2006",vtMissing,vtMissing);
   var_Items->AddLink("L2",h2,"",h3,"");
   var_Items->PutLink("L2",EXG2ANTTLib::exLinkText,"L2");
spG2antt1->EndUpdate();
```

The following C# sample makes the links that starts from selected bar being wider, while the rest being transparent:

```
axG2antt1.BeginUpdate();
axG2antt1.Columns.Add("Tasks");
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
```

```
   var_Chart.FirstVisibleDate = "9/20/2006";
   var_Chart.set_PaneWidth(0 != 0,64);

var_Chart.set_ShowLinksWidth(EXG2ANTTLib.ShowLinksEnum.exShowLinksStartFrom,2);

var_Chart.set_ShowLinksStyle(EXG2ANTTLib.ShowLinksEnum.exShowLinksStartFrom,EXG2


var_Chart.set_ShowLinksColor(EXG2ANTTLib.ShowLinksEnum.exShowLinksEndTo,1315860


var_Chart.set_ShowLinksColor(EXG2ANTTLib.ShowLinksEnum.exShowUnselectedLinks,131

EXG2ANTTLib.Items var_Items = axG2antt1.Items;
   int h1 = var_Items.AddItem("Task 1");
   var_Items.AddBar(h1,"Task","9/21/2006","9/23/2006",null,null);
   int h2 = var_Items.AddItem("Task 2");
   var_Items.AddBar(h2,"Task","9/25/2006","9/27/2006",null,null);
   var_Items.set_ItemBar(h2,"",EXG2ANTTLib.ItemBarPropertyEnum.exBarSelected,true);
   var_Items.AddLink("L1",h1,"",h2,"");
   var_Items.set_Link("L1",EXG2ANTTLib.LinkPropertyEnum.exLinkText,"L1");
   int h3 = var_Items.AddItem("Task 3");
   var_Items.AddBar(h3,"Task","9/29/2006","10/2/2006",null,null);
   var_Items.AddLink("L2",h2,"",h3,"");
   var_Items.set_Link("L2",EXG2ANTTLib.LinkPropertyEnum.exLinkText,"L2");
axG2antt1.EndUpdate();
```

The following VFP sample makes the links that starts from selected bar being wider, while the rest being transparent:

```
with thisform.G2antt1
   .BeginUpdate
   .Columns.Add("Tasks")
   with .Chart
      .FirstVisibleDate = {^2006-9-20}
      .PaneWidth(0) = 64
      .ShowLinksWidth(1) = 2
      .ShowLinksStyle(1) = 2
```

```
    .ShowLinksColor(2) = RGB(200,200,200)
    .ShowLinksColor(4) = RGB(200,200,200)
  endwith
  with .Items
    h1 = .AddItem("Task 1")
    .AddBar(h1,"Task",{^2006-9-21},{^2006-9-23})
    h2 = .AddItem("Task 2")
    .AddBar(h2,"Task",{^2006-9-25},{^2006-9-27})
    .DefaultItem = h2
    .ItemBar(0,"",257) = .T.
    .AddLink("L1",h1,"",h2,"")
    .Link("L1",12) = "L1"
    h3 = .AddItem("Task 3")
    .AddBar(h3,"Task",{^2006-9-29},{^2006-10-2})
    .AddLink("L2",h2,"",h3,"")
    .Link("L2",12) = "L2"
  endwith
  .EndUpdate
endwith
```

The following Delphi sample makes the links that starts from selected bar being wider, while the rest being transparent:

```
with AxG2antt1 do
begin
  BeginUpdate();
  Columns.Add('Tasks');
  with Chart do
  begin
    FirstVisibleDate := '9/20/2006';
    PaneWidth[0 <> 0] := 64;
    ShowLinksWidth[EXG2ANTTLib.ShowLinksEnum.exShowLinksStartFrom] := 2;
    ShowLinksStyle[EXG2ANTTLib.ShowLinksEnum.exShowLinksStartFrom] :=
EXG2ANTTLib.LinkStyleEnum.exLinkDot;
    ShowLinksColor[EXG2ANTTLib.ShowLinksEnum.exShowLinksEndTo] := 13158600;
    ShowLinksColor[EXG2ANTTLib.ShowLinksEnum.exShowUnselectedLinks] :=
13158600;
  end;
```

```
  with Items do
  begin
    h1 := AddItem('Task 1');
    AddBar(h1,'Task','9/21/2006','9/23/2006',Nil,Nil);
    h2 := AddItem('Task 2');
    AddBar(h2,'Task','9/25/2006','9/27/2006',Nil,Nil);
    ItemBar[h2,'',EXG2ANTTLib.ItemBarPropertyEnum.exBarSelected] := TObject(True);
    AddLink('L1',h1,'',h2,'');
    Link['L1',EXG2ANTTLib.LinkPropertyEnum.exLinkText] := 'L1';
    h3 := AddItem('Task 3');
    AddBar(h3,'Task','9/29/2006','10/2/2006',Nil,Nil);
    AddLink('L2',h2,'',h3,'');
    Link['L2',EXG2ANTTLib.LinkPropertyEnum.exLinkText] := 'L2';
  end;
  EndUpdate();
end
```

property Chart.ShowLinksStyle(Links as ShowLinksEnum) as LinkStyleEnum

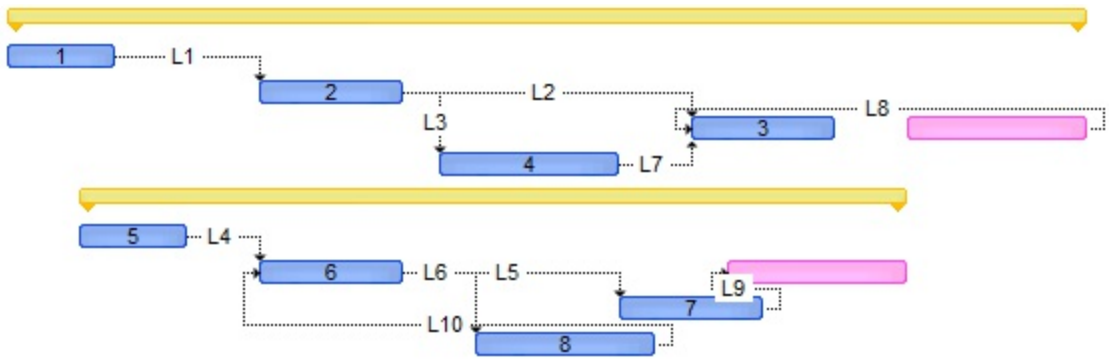
Retrieves or sets a value that indicates the style to display the links based on the user selection.

Type	Description
Links as ShowLinksEnum	A ShowLinksEnum expression that specifies the links style being accessed.
LinkStyleEnum	A LinkStyleEnum expression that specifies the link to display the links.

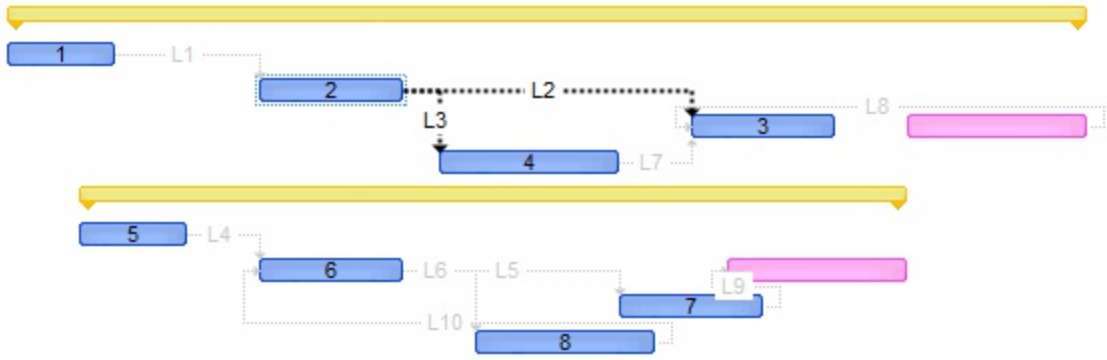
By default the ShowLinksStyle property is 0, which means that it has no effect. The ShowLinksStyle property specifies the width to show the links when the link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. The ShowLinksColor property specifies the color to display the links when link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. The ShowLinksWidth property specifies the width to show the links when the link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. *You can hide the links if the ShowLinksColor property is the same as the chart's background color being specified by the BackColor property.*

The ShowLinks property specifies whether the chart's links are visible or hidden. Use the LinkColor property to specify the color for all links in the chart area. Use the LinkStyle property to specify the style for all links in the chart area. Use the LinkWidth property to specify the width of the pen, in pixels, to draw the links between bars. Use the AddLink method to add programmatically a link a bar with another.

The following screen shows the links when no bar is selected:



The following screen shows the links when the bar 2 is selected:



The following VB sample makes the links that starts from selected bar being wider, while the rest being transparent:

```
With G2antt1
   .BeginUpdate
   .Columns.Add "Tasks"
   With .Chart
      .FirstVisibleDate = #9/20/2006#
      .PaneWidth(0) = 64
      .ShowLinksWidth(exShowLinksStartFrom) = 2
      .ShowLinksStyle(exShowLinksStartFrom) = exLinkDot
      .ShowLinksColor(exShowLinksEndTo) = RGB(200,200,200)
      .ShowLinksColor(exShowUnselectedLinks) = RGB(200,200,200)
   End With
   With .Items
      h1 = .AddItem("Task 1")
      .AddBar h1,"Task",#9/21/2006#,#9/23/2006#
      h2 = .AddItem("Task 2")
      .AddBar h2,"Task",#9/25/2006#,#9/27/2006#
      .ItemBar(h2,"",exBarSelected) = True
      .AddLink "L1",h1,"",h2,""
      .Link("L1",exLinkText) = "L1"
      h3 = .AddItem("Task 3")
      .AddBar h3,"Task",#9/29/2006#,#10/2/2006#
      .AddLink "L2",h2,"",h3,""
      .Link("L2",exLinkText) = "L2"
   End With
   .EndUpdate
End With
```

The following VB.NET sample makes the links that starts from selected bar being wider, while the rest being transparent:

```
Dim h1,h2,h3
With AxG2antt1
   .BeginUpdate
   .Columns.Add "Tasks"
   With .Chart
      .FirstVisibleDate = #9/20/2006#
      .PaneWidth(0) = 64
      .ShowLinksWidth(EXG2ANTTLib.ShowLinksEnum.exShowLinksStartFrom) = 2
      .ShowLinksStyle(EXG2ANTTLib.ShowLinksEnum.exShowLinksStartFrom) =
EXG2ANTTLib.LinkStyleEnum.exLinkDot
      .ShowLinksColor(EXG2ANTTLib.ShowLinksEnum.exShowLinksEndTo) = 13158600
      .ShowLinksColor(EXG2ANTTLib.ShowLinksEnum.exShowUnselectedLinks) =
13158600
   End With
   With .Items
      h1 = .AddItem("Task 1")
      .AddBar h1,"Task",#9/21/2006#,#9/23/2006#
      h2 = .AddItem("Task 2")
      .AddBar h2,"Task",#9/25/2006#,#9/27/2006#
      .ItemBar(h2,"",EXG2ANTTLib.ItemBarPropertyEnum.exBarSelected) = True
      .AddLink "L1",h1,"",h2,""
      .Link("L1",EXG2ANTTLib.LinkPropertyEnum.exLinkText) = "L1"
      h3 = .AddItem("Task 3")
      .AddBar h3,"Task",#9/29/2006#,#10/2/2006#
      .AddLink "L2",h2,"",h3,""
      .Link("L2",EXG2ANTTLib.LinkPropertyEnum.exLinkText) = "L2"
   End With
   .EndUpdate
End With
```

The following C++ sample makes the links that starts from selected bar being wider, while the rest being transparent:

```
/*
   Copy and paste the following directives to your header file as
```

```
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
spG2antt1->GetColumns()->Add(L"Tasks");
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutFirstVisibleDate("9/20/2006");
   var_Chart->PutPaneWidth(0,64);
   var_Chart->PutShowLinksWidth(EXG2ANTTLib::exShowLinksStartFrom,2);
   var_Chart-
>PutShowLinksStyle(EXG2ANTTLib::exShowLinksStartFrom,EXG2ANTTLib::exLinkDot);
   var_Chart->PutShowLinksColor(EXG2ANTTLib::exShowLinksEndTo,RGB(200,200,200));
   var_Chart-
>PutShowLinksColor(EXG2ANTTLib::exShowUnselectedLinks,RGB(200,200,200));
EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems();
   long h1 = var_Items->AddItem("Task 1");
   var_Items->AddBar(h1,"Task","9/21/2006","9/23/2006",vtMissing,vtMissing);
   long h2 = var_Items->AddItem("Task 2");
   var_Items->AddBar(h2,"Task","9/25/2006","9/27/2006",vtMissing,vtMissing);
   var_Items->PutItemBar(h2,"",EXG2ANTTLib::exBarSelected,VARIANT_TRUE);
   var_Items->AddLink("L1",h1,"",h2,"");
   var_Items->PutLink("L1",EXG2ANTTLib::exLinkText,"L1");
   long h3 = var_Items->AddItem("Task 3");
   var_Items->AddBar(h3,"Task","9/29/2006","10/2/2006",vtMissing,vtMissing);
   var_Items->AddLink("L2",h2,"",h3,"");
   var_Items->PutLink("L2",EXG2ANTTLib::exLinkText,"L2");
spG2antt1->EndUpdate();
```

The following C# sample makes the links that starts from selected bar being wider, while the rest being transparent:

```
axG2antt1.BeginUpdate();
axG2antt1.Columns.Add("Tasks");
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
```

```
   var_Chart.FirstVisibleDate = "9/20/2006";
   var_Chart.set_PaneWidth(0 != 0,64);

var_Chart.set_ShowLinksWidth(EXG2ANTTLib.ShowLinksEnum.exShowLinksStartFrom,2);

var_Chart.set_ShowLinksStyle(EXG2ANTTLib.ShowLinksEnum.exShowLinksStartFrom,EXG2


var_Chart.set_ShowLinksColor(EXG2ANTTLib.ShowLinksEnum.exShowLinksEndTo,1315860


var_Chart.set_ShowLinksColor(EXG2ANTTLib.ShowLinksEnum.exShowUnselectedLinks,131

EXG2ANTTLib.Items var_Items = axG2antt1.Items;
   int h1 = var_Items.AddItem("Task 1");
   var_Items.AddBar(h1,"Task","9/21/2006","9/23/2006",null,null);
   int h2 = var_Items.AddItem("Task 2");
   var_Items.AddBar(h2,"Task","9/25/2006","9/27/2006",null,null);
   var_Items.set_ItemBar(h2,"",EXG2ANTTLib.ItemBarPropertyEnum.exBarSelected,true);
   var_Items.AddLink("L1",h1,"",h2,"");
   var_Items.set_Link("L1",EXG2ANTTLib.LinkPropertyEnum.exLinkText,"L1");
   int h3 = var_Items.AddItem("Task 3");
   var_Items.AddBar(h3,"Task","9/29/2006","10/2/2006",null,null);
   var_Items.AddLink("L2",h2,"",h3,"");
   var_Items.set_Link("L2",EXG2ANTTLib.LinkPropertyEnum.exLinkText,"L2");
axG2antt1.EndUpdate();
```

The following VFP sample makes the links that starts from selected bar being wider, while the rest being transparent:

```
with thisform.G2antt1
   .BeginUpdate
   .Columns.Add("Tasks")
   with .Chart
      .FirstVisibleDate = {^2006-9-20}
      .PaneWidth(0) = 64
      .ShowLinksWidth(1) = 2
      .ShowLinksStyle(1) = 2
```

```
    .ShowLinksColor(2) = RGB(200,200,200)
    .ShowLinksColor(4) = RGB(200,200,200)
  endwith
  with .Items
    h1 = .AddItem("Task 1")
    .AddBar(h1,"Task",{^2006-9-21},{^2006-9-23})
    h2 = .AddItem("Task 2")
    .AddBar(h2,"Task",{^2006-9-25},{^2006-9-27})
    .DefaultItem = h2
    .ItemBar(0,"",257) = .T.
    .AddLink("L1",h1,"",h2,"")
    .Link("L1",12) = "L1"
    h3 = .AddItem("Task 3")
    .AddBar(h3,"Task",{^2006-9-29},{^2006-10-2})
    .AddLink("L2",h2,"",h3,"")
    .Link("L2",12) = "L2"
  endwith
  .EndUpdate
endwith
```

The following Delphi sample makes the links that starts from selected bar being wider, while the rest being transparent:

```
with AxG2antt1 do
begin
  BeginUpdate();
  Columns.Add('Tasks');
  with Chart do
  begin
    FirstVisibleDate := '9/20/2006';
    PaneWidth[0 <> 0] := 64;
    ShowLinksWidth[EXG2ANTTLib.ShowLinksEnum.exShowLinksStartFrom] := 2;
    ShowLinksStyle[EXG2ANTTLib.ShowLinksEnum.exShowLinksStartFrom] :=
EXG2ANTTLib.LinkStyleEnum.exLinkDot;
    ShowLinksColor[EXG2ANTTLib.ShowLinksEnum.exShowLinksEndTo] := 13158600;
    ShowLinksColor[EXG2ANTTLib.ShowLinksEnum.exShowUnselectedLinks] :=
13158600;
  end;
```

```
  with Items do
  begin
    h1 := AddItem('Task 1');
    AddBar(h1,'Task','9/21/2006','9/23/2006',Nil,Nil);
    h2 := AddItem('Task 2');
    AddBar(h2,'Task','9/25/2006','9/27/2006',Nil,Nil);
    ItemBar[h2,'',EXG2ANTTLib.ItemBarPropertyEnum.exBarSelected] := TObject(True);
    AddLink('L1',h1,'',h2,'');
    Link['L1',EXG2ANTTLib.LinkPropertyEnum.exLinkText] := 'L1';
    h3 := AddItem('Task 3');
    AddBar(h3,'Task','9/29/2006','10/2/2006',Nil,Nil);
    AddLink('L2',h2,'',h3,'');
    Link['L2',EXG2ANTTLib.LinkPropertyEnum.exLinkText] := 'L2';
  end;
  EndUpdate();
end
```

property Chart.ShowLinksWidth(Links as ShowLinksEnum) as Long

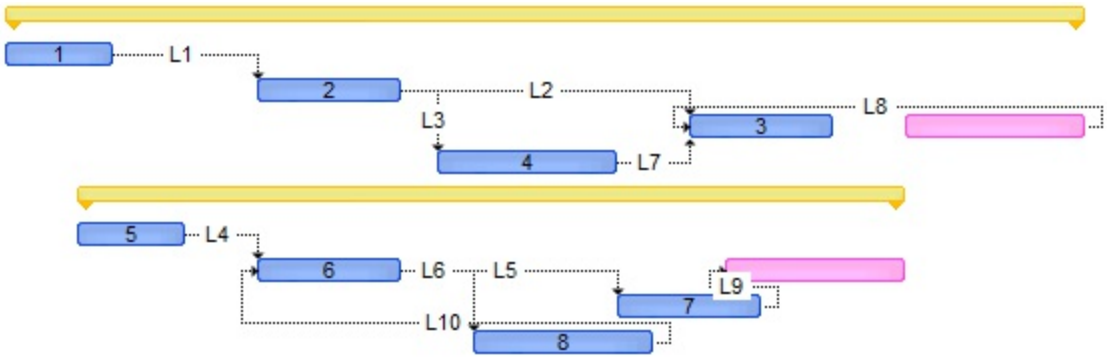
Retrieves or sets a value that indicates the width to display the links based on the user selection.

Type	Description
Links as ShowLinksEnum	A ShowLinksEnum expression that specifies the links width being accessed.
Long	A Long expression that specifies the width to display the links.

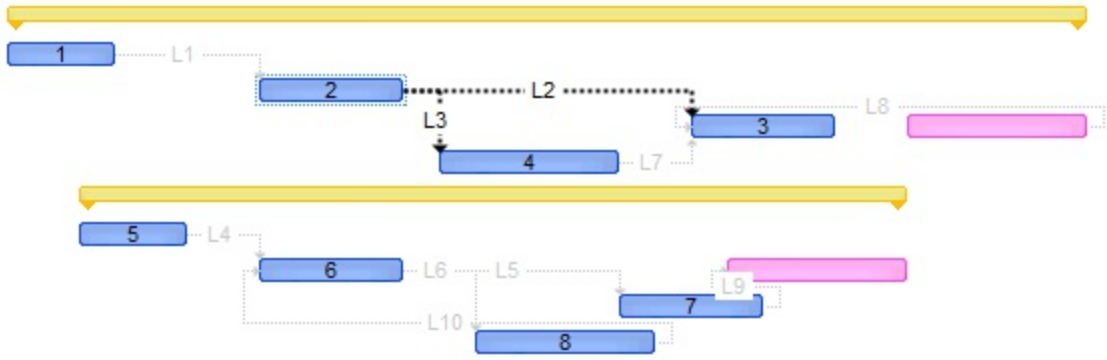
By default the ShowLinksWidth property is 0, which means that it has no effect. The ShowLinksWidth property specifies the width to show the links when the link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. The ShowLinksStyle property specifies the width to show the links when the link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. The ShowLinksColor property specifies the color to display the links when link starts from selected bar, ends on selected bar, or when it is not related to any of selected bars. *You can hide the links if the ShowLinksColor property is the same as the chart's background color being specified by the BackColor property.*

The ShowLinks property specifies whether the chart's links are visible or hidden. Use the LinkColor property to specify the color for all links in the chart area. Use the LinkStyle property to specify the style for all links in the chart area. Use the LinkWidth property to specify the width of the pen, in pixels, to draw the links between bars. Use the AddLink method to add programmatically a link a bar with another.

The following screen shows the links when no bar is selected:



The following screen shows the links when the bar 2 is selected:



The following VB sample makes the links that starts from selected bar being wider, while the rest being transparent:

```
With G2antt1
   .BeginUpdate
   .Columns.Add "Tasks"
   With .Chart
      .FirstVisibleDate = #9/20/2006#
      .PaneWidth(0) = 64
      .ShowLinksWidth(exShowLinksStartFrom) = 2
      .ShowLinksStyle(exShowLinksStartFrom) = exLinkDot
      .ShowLinksColor(exShowLinksEndTo) = RGB(200,200,200)
      .ShowLinksColor(exShowUnselectedLinks) = RGB(200,200,200)
   End With
   With .Items
      h1 = .AddItem("Task 1")
      .AddBar h1,"Task",#9/21/2006#,#9/23/2006#
      h2 = .AddItem("Task 2")
      .AddBar h2,"Task",#9/25/2006#,#9/27/2006#
      .ItemBar(h2,"",exBarSelected) = True
      .AddLink "L1",h1,"",h2,""
      .Link("L1",exLinkText) = "L1"
      h3 = .AddItem("Task 3")
      .AddBar h3,"Task",#9/29/2006#,#10/2/2006#
      .AddLink "L2",h2,"",h3,""
      .Link("L2",exLinkText) = "L2"
   End With
   .EndUpdate
End With
```

The following VB.NET sample makes the links that starts from selected bar being wider, while the rest being transparent:

```
Dim h1,h2,h3
With AxG2antt1
   .BeginUpdate
   .Columns.Add "Tasks"
   With .Chart
      .FirstVisibleDate = #9/20/2006#
      .PaneWidth(0) = 64
      .ShowLinksWidth(EXG2ANTTLib.ShowLinksEnum.exShowLinksStartFrom) = 2
      .ShowLinksStyle(EXG2ANTTLib.ShowLinksEnum.exShowLinksStartFrom) =
EXG2ANTTLib.LinkStyleEnum.exLinkDot
      .ShowLinksColor(EXG2ANTTLib.ShowLinksEnum.exShowLinksEndTo) = 13158600
      .ShowLinksColor(EXG2ANTTLib.ShowLinksEnum.exShowUnselectedLinks) =
13158600
   End With
   With .Items
      h1 = .AddItem("Task 1")
      .AddBar h1,"Task",#9/21/2006#,#9/23/2006#
      h2 = .AddItem("Task 2")
      .AddBar h2,"Task",#9/25/2006#,#9/27/2006#
      .ItemBar(h2,"",EXG2ANTTLib.ItemBarPropertyEnum.exBarSelected) = True
      .AddLink "L1",h1,"",h2,""
      .Link("L1",EXG2ANTTLib.LinkPropertyEnum.exLinkText) = "L1"
      h3 = .AddItem("Task 3")
      .AddBar h3,"Task",#9/29/2006#,#10/2/2006#
      .AddLink "L2",h2,"",h3,""
      .Link("L2",EXG2ANTTLib.LinkPropertyEnum.exLinkText) = "L2"
   End With
   .EndUpdate
End With
```

The following C++ sample makes the links that starts from selected bar being wider, while the rest being transparent:

```
/*
   Copy and paste the following directives to your header file as
```

```
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
spG2antt1->GetColumns()->Add(L"Tasks");
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutFirstVisibleDate("9/20/2006");
   var_Chart->PutPaneWidth(0,64);
   var_Chart->PutShowLinksWidth(EXG2ANTTLib::exShowLinksStartFrom,2);
   var_Chart-
>PutShowLinksStyle(EXG2ANTTLib::exShowLinksStartFrom,EXG2ANTTLib::exLinkDot);
   var_Chart->PutShowLinksColor(EXG2ANTTLib::exShowLinksEndTo,RGB(200,200,200));
   var_Chart-
>PutShowLinksColor(EXG2ANTTLib::exShowUnselectedLinks,RGB(200,200,200));
EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems();
   long h1 = var_Items->AddItem("Task 1");
   var_Items->AddBar(h1,"Task","9/21/2006","9/23/2006",vtMissing,vtMissing);
   long h2 = var_Items->AddItem("Task 2");
   var_Items->AddBar(h2,"Task","9/25/2006","9/27/2006",vtMissing,vtMissing);
   var_Items->PutItemBar(h2,"",EXG2ANTTLib::exBarSelected,VARIANT_TRUE);
   var_Items->AddLink("L1",h1,"",h2,"");
   var_Items->PutLink("L1",EXG2ANTTLib::exLinkText,"L1");
   long h3 = var_Items->AddItem("Task 3");
   var_Items->AddBar(h3,"Task","9/29/2006","10/2/2006",vtMissing,vtMissing);
   var_Items->AddLink("L2",h2,"",h3,"");
   var_Items->PutLink("L2",EXG2ANTTLib::exLinkText,"L2");
spG2antt1->EndUpdate();
```

The following C# sample makes the links that starts from selected bar being wider, while the rest being transparent:

```
axG2antt1.BeginUpdate();
axG2antt1.Columns.Add("Tasks");
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
```

```
   var_Chart.FirstVisibleDate = "9/20/2006";
   var_Chart.set_PaneWidth(0 != 0,64);

var_Chart.set_ShowLinksWidth(EXG2ANTTLib.ShowLinksEnum.exShowLinksStartFrom,2);

var_Chart.set_ShowLinksStyle(EXG2ANTTLib.ShowLinksEnum.exShowLinksStartFrom,EXG2


var_Chart.set_ShowLinksColor(EXG2ANTTLib.ShowLinksEnum.exShowLinksEndTo,1315860


var_Chart.set_ShowLinksColor(EXG2ANTTLib.ShowLinksEnum.exShowUnselectedLinks,131

EXG2ANTTLib.Items var_Items = axG2antt1.Items;
   int h1 = var_Items.AddItem("Task 1");
   var_Items.AddBar(h1,"Task","9/21/2006","9/23/2006",null,null);
   int h2 = var_Items.AddItem("Task 2");
   var_Items.AddBar(h2,"Task","9/25/2006","9/27/2006",null,null);
   var_Items.set_ItemBar(h2,"",EXG2ANTTLib.ItemBarPropertyEnum.exBarSelected,true);
   var_Items.AddLink("L1",h1,"",h2,"");
   var_Items.set_Link("L1",EXG2ANTTLib.LinkPropertyEnum.exLinkText,"L1");
   int h3 = var_Items.AddItem("Task 3");
   var_Items.AddBar(h3,"Task","9/29/2006","10/2/2006",null,null);
   var_Items.AddLink("L2",h2,"",h3,"");
   var_Items.set_Link("L2",EXG2ANTTLib.LinkPropertyEnum.exLinkText,"L2");
axG2antt1.EndUpdate();
```

The following VFP sample makes the links that starts from selected bar being wider, while the rest being transparent:

```
with thisform.G2antt1
   .BeginUpdate
   .Columns.Add("Tasks")
   with .Chart
      .FirstVisibleDate = {^2006-9-20}
      .PaneWidth(0) = 64
      .ShowLinksWidth(1) = 2
      .ShowLinksStyle(1) = 2
```

```
    .ShowLinksColor(2) = RGB(200,200,200)
    .ShowLinksColor(4) = RGB(200,200,200)
  endwith
  with .Items
    h1 = .AddItem("Task 1")
    .AddBar(h1,"Task",{^2006-9-21},{^2006-9-23})
    h2 = .AddItem("Task 2")
    .AddBar(h2,"Task",{^2006-9-25},{^2006-9-27})
    .DefaultItem = h2
    .ItemBar(0,"",257) = .T.
    .AddLink("L1",h1,"",h2,"")
    .Link("L1",12) = "L1"
    h3 = .AddItem("Task 3")
    .AddBar(h3,"Task",{^2006-9-29},{^2006-10-2})
    .AddLink("L2",h2,"",h3,"")
    .Link("L2",12) = "L2"
  endwith
  .EndUpdate
endwith
```

The following Delphi sample makes the links that starts from selected bar being wider, while the rest being transparent:

```
with AxG2antt1 do
begin
  BeginUpdate();
  Columns.Add('Tasks');
  with Chart do
  begin
    FirstVisibleDate := '9/20/2006';
    PaneWidth[0 <> 0] := 64;
    ShowLinksWidth[EXG2ANTTLib.ShowLinksEnum.exShowLinksStartFrom] := 2;
    ShowLinksStyle[EXG2ANTTLib.ShowLinksEnum.exShowLinksStartFrom] :=
EXG2ANTTLib.LinkStyleEnum.exLinkDot;
    ShowLinksColor[EXG2ANTTLib.ShowLinksEnum.exShowLinksEndTo] := 13158600;
    ShowLinksColor[EXG2ANTTLib.ShowLinksEnum.exShowUnselectedLinks] :=
13158600;
  end;
```

```
  with Items do
  begin
    h1 := AddItem('Task 1');
    AddBar(h1,'Task','9/21/2006','9/23/2006',Nil,Nil);
    h2 := AddItem('Task 2');
    AddBar(h2,'Task','9/25/2006','9/27/2006',Nil,Nil);
    ItemBar[h2,'',EXG2ANTTLib.ItemBarPropertyEnum.exBarSelected] := TObject(True);
    AddLink('L1',h1,'',h2,'');
    Link['L1',EXG2ANTTLib.LinkPropertyEnum.exLinkText] := 'L1';
    h3 := AddItem('Task 3');
    AddBar(h3,'Task','9/29/2006','10/2/2006',Nil,Nil);
    AddLink('L2',h2,'',h3,'');
    Link['L2',EXG2ANTTLib.LinkPropertyEnum.exLinkText] := 'L2';
  end;
  EndUpdate();
end
```

property Chart.ShowNonworkingDates as Boolean

Shows or hides nonworking dates.

Type	Description
Boolean	A boolean expression that indicates whether the chart marks the nonworking days.

Use the ShowNonworkingDates property to stop highlighting the nonworking dates. The NonworkingDays property specifies the days being marked as nonworking in a week. Use the AddNonworkingDate method to add custom dates as being nonworking days. Use the IsNonworkingDate property to specify whether the date is a nonworking day. Use the NonworkingDaysPattern property to specify the pattern being used to fill non-working days. The NonworkingDaysColor property specifies the color being used to fill the non-working days. Use the ClearNonworkingDates method to remove all nonworking dates. Use the ItemNonworkingUnits property to specify different non-working zones for different items. Use the UnitWidthNonworking property to specify a different width for non-working units in the base level. The ShowNonworkingUnits property specifies whether the nonworking units are shown or hidden. Use the FormatLabel property to specify the format of the chart's level (header).

Generally, you can use the following functions to specify non-working parts in the chart:

- The NonworkingDays property specifies the days being marked as nonworking in a week. Use the AddNonworkingDate method to add custom dates as being nonworking days. Use the NonworkingDaysPattern property to specify the pattern being used to fill non-working days. The NonworkingDaysColor property specifies the color being used to fill the non-working days. Use the ClearNonworkingDates method to remove all nonworking dates. Use the IsDateVisible property to specify whether a date fits the chart's area.
- The NonworkingHours property indicates the non-working hours with in a day. The non-working hours are shown using the NonworkingHoursPattern and the NonworkingHoursColor which defines the pattern and the color, when the base level of the chart displays hours, if the ShowNonworkingUnits property is True (by default).
- The ItemNonworkingUnits property specifies different non-working zones for different items. If the Item parameter indicates a valid handle, the IsNonworkingDate property queries the non-working expression for the item if the giving Date parameter is being non-working or working unit.

Generally, you can use the following attributes for a bar (ItemBar property) to work with non-working part of the bars:

- exBarWorkingCount attribute specifies the working count in days for the giving bar. For instance, if the exBarWorkingCount is 1 indicates a full day, or 24 working hours, while

if it is 1/24 it indicates one working hour.

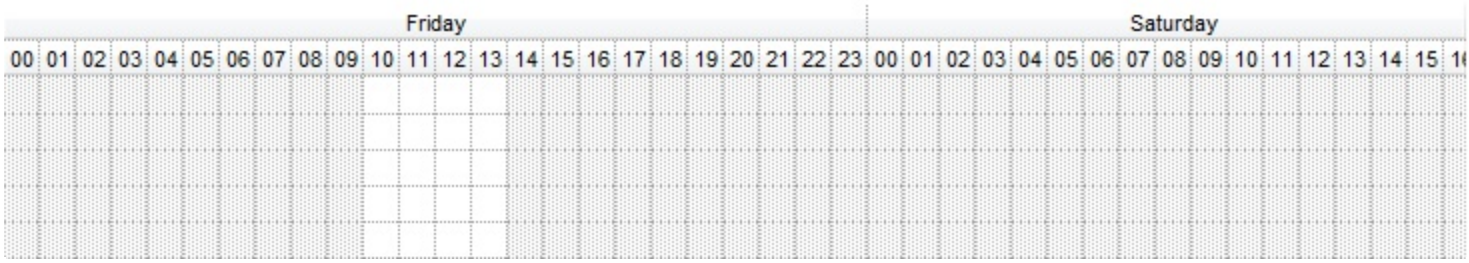
- exBarNonWorkingCount attributes specifies the working count in days for the giving bar. For instance, if the exBarNonWorkingCount is 1 indicates a full day, or 24 non-working hours, while if it is 1/24 it indicates one non-working hour.
- exBarWorkingUnits attribute retrieves a safe array of pair (start-end) that indicates the working parts of the bar. You can use the exBarWorkingUnitsAsString attribute to display the working parts of the bar as a string. The /NET assembly provides the *public virtual DateTime[] get_BarWorkingUnits(int Item, object Key)* method that returns the array of 2 DateTime objects that specifies the working parts of the bar.
- exBarWorkingUnitsAsString attribute retrieves the working part of the bar as a string, in other words it is similar with the exBarWorkingUnits excepts that it returns a string that shows the working parts of the bar.
- exBarNonWorkingUnits attribute retrieves a safe array of pair (start-end) that indicates the non-working parts of the bar. You can use the exBarNonWorkingUnitsAsString attribute to display the non-working parts of the bar as a string. The /NET assembly provides the *public virtual DateTime[] get_BarNonWorkingUnits(int Item, object Key)* method that returns the array of 2 DateTime objects that specifies the working parts of the bar.
- exBarNonWorkingUnitsAsString attribute retrieves the non-working part of the bar as a string, in other words it is similar with the exBarNonWorkingUnits excepts that it returns a string that shows the non-working parts of the bar.

The following screen shots shows the chart based on the value for the properties:

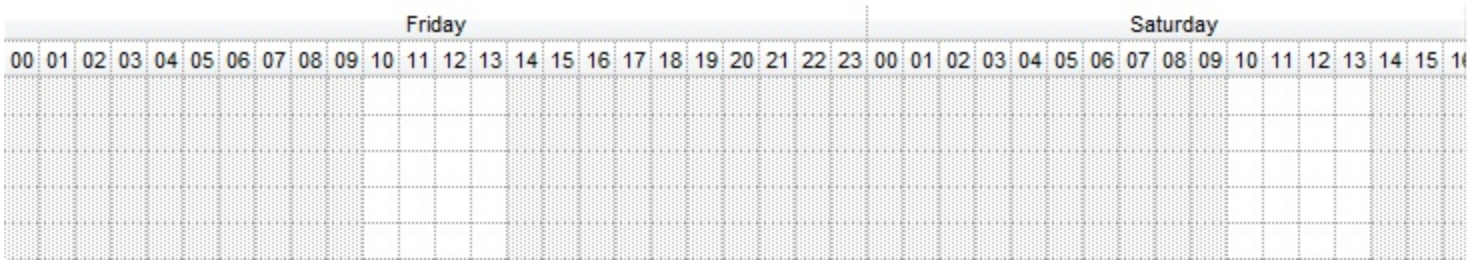
- ShowNonworkingDates property indicates whether the non-working (weekend) days are shown or hidden
- ShowNonworkingUnits property indicates whether the charts lists the non-working units, hours or days
- UnitWidthNonworking property specifies the width for non-working units, hours or days

Use the AdjustLevelsToBase property to align the levels to base level. The chart's NonworkingDays property is 65 (indicates that saturday and sunday are non-working days) while the NonworkingHours property is 16761855 (indicates working hour from 10AM to 2 PM).

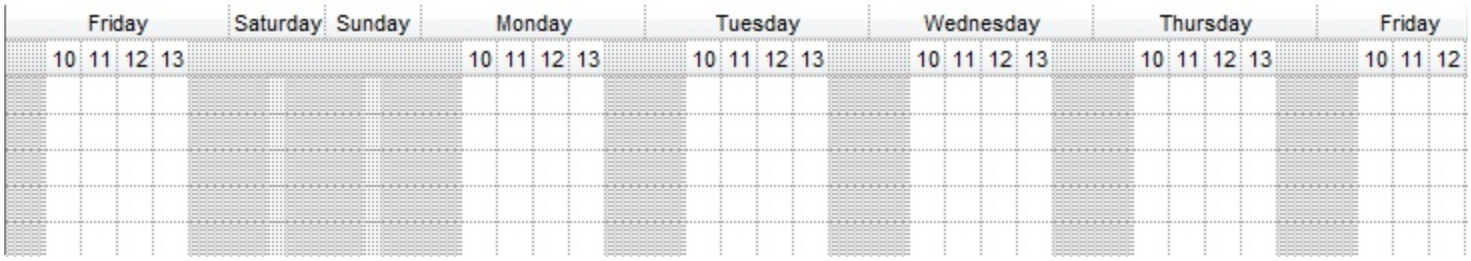
- ShowNonworkingDates = True, UnitWidthNonworking = 0, ShowNonworkingUnits = True (by default)



- ShowNonworkingDates = False, UnitWidthNonworking = 0, ShowNonworkingUnits = True

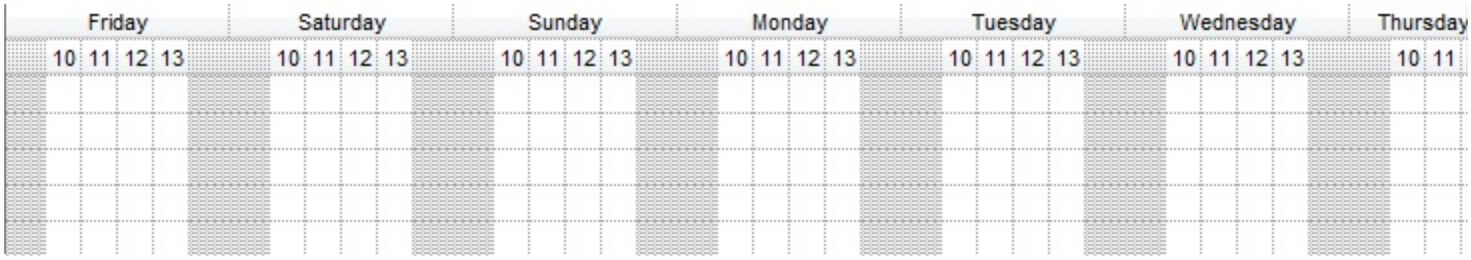


- ShowNonworkingDates = True, UnitWidthNonworking = 2, ShowNonworkingUnits = True



- ShowNonworkingDates = True, UnitWidthNonworking = -18, ShowNonworkingUnits = True

- ShowNonworkingDates = False, UnitWidthNonworking = 2, ShowNonworkingUnits = True



- ShowNonworkingDates = False, UnitWidthNonworking = -18, ShowNonworkingUnits = True

- ShowNonworkingDates = True, UnitWidthNonworking = 0, ShowNonworkingUnits = False

- ShowNonworkingDates = True, UnitWidthNonworking = 2, ShowNonworkingUnits = False

- ShowNonworkingDates = True, UnitWidthNonworking = -18, ShowNonworkingUnits = False

- ShowNonworkingDates = False, UnitWidthNonworking = 0, ShowNonworkingUnits = False

property Chart.ShowNonworkingHours as Boolean

Shows or hides nonworking hours.

Type	Description
Boolean	A boolean expression that indicates whether the chart marks the nonworking hours.

The ShowNonworkingHours property specifies whether the non-working hours are shown on the chart. While the Chart.ShowNonworkingHours property is True and Chart.ShowNonworkingUnits property is True, the non-working hours are visible on the chart only if the Chart.NonworkingHours property is set, Chart.UnitScale is exDay (and Level.Count property is 1), exHour, exMinute and exSecond. The NonworkingHours property indicates the non-working hours with in a day. The non-working hours are shown using the NonworkingHoursPattern and the NonworkingHoursColor which defines the pattern and the color, when the base level of the chart displays hours, if the ShowNonworkingUnits property is True (by default).

The following screen shot shows the control's chart when ShowNonworkingHours and ShowNonworkingUnits properties are False:

The following screen shot shows the control's chart when ShowNonworkingHours and ShowNonworkingUnits properties are True (by default):

property Chart.ShowNonworkingUnits as Boolean

Retrieves or sets a value that indicates whether the non-working units are visible or hidden.

Type	Description
Boolean	A Boolean expression that specifies whether the non-working units (hours or days) are visible or hidden.

By default, the ShowNonworkingUnits property is True. In other words, by default the control displays the non-working units. Use the ShowNonworkingDates property to specify whether the the days are shown or hidden while the ShowNonworkingUnits property is False. Use the ShowNonworkingHours property to specify whether the the hours are shown or hidden while the ShowNonworkingUnits property is False. Use the UnitWidthNonworking property to specify a different width for non-working units in the base level. Use the NonworkingHours property to specify the non-working hours in your chart. Use the NonworkingDays property to specify the non-working days. Use the ShowNonworkingUnits property to display ONLY working units. For instance, you can display for each day the hours from 08:00 AM to 04:00PM, as the other hours (non working hours) are not displayed in the chart. Use the FormatLabel property to specify the format of the chart's level (header). The ItemNonworkingUnits property specifies different non-working zones for different items. The ShowNonworkingUnits property has no effect if the NonworkingHours and NonWorkingsDays properties are 0.

If ShowNonworkingUnits property is True the:

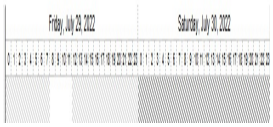
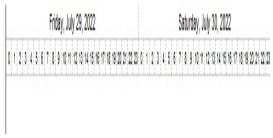
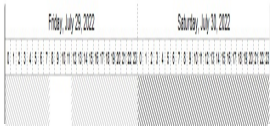
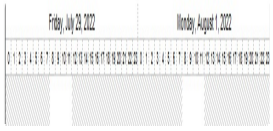
- ShowNonworkingDates property specifies whether the non-working days pattern is displayed or hidden
- ShowNonworkingHours property specifies whether the non-working hours pattern is displayed or hidden

If ShowNonworkingUnits property is False the:

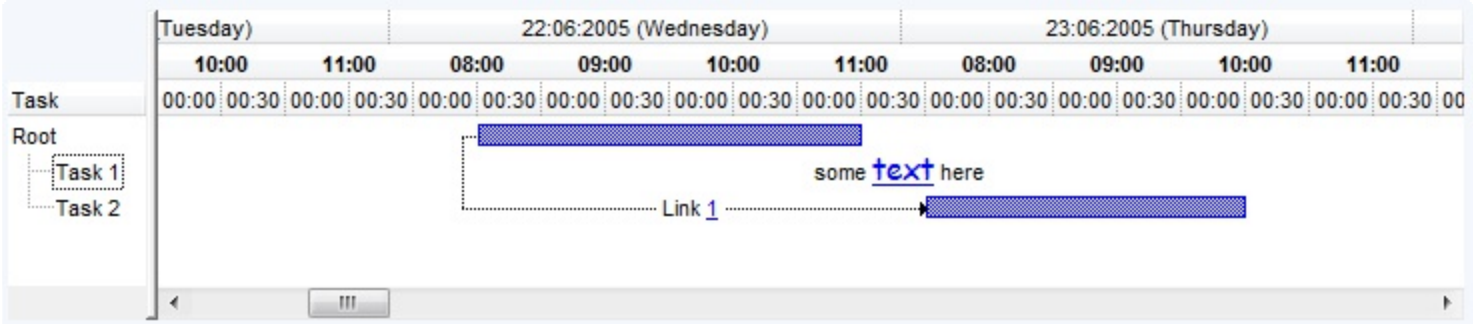
- ShowNonworkingDates property specifies whether the non-working days is shown(visible) or hidden
- ShowNonworkingHours property specifies whether the non-working hours is shown(visible) or hidden

The following tables shows the control while **ShowNonworkingUnits / ShowNonworkingDates / ShowNonworkingHours** properties have different values (**x** indicate true):

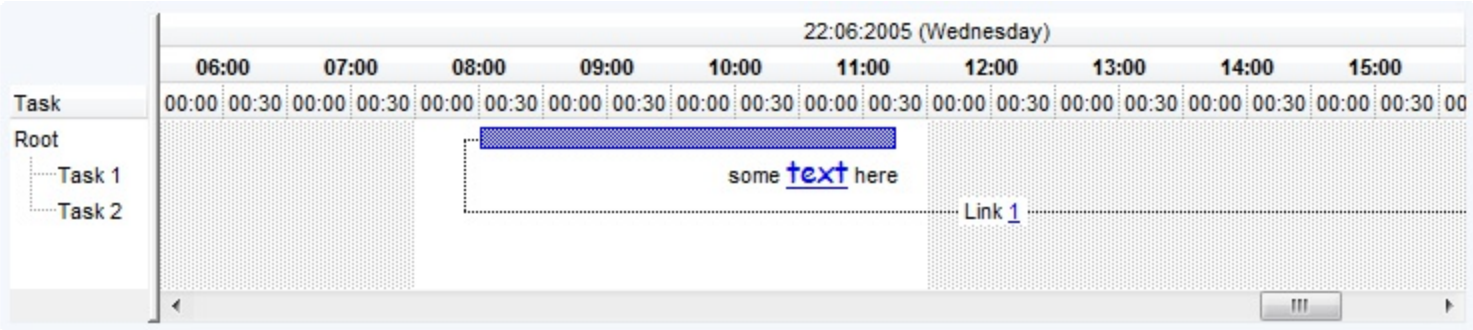
ShowNonworkingUnits	ShowNonworkingDates	ShowNonworkingHours

x	x	x	
x	x		
x		x	
x			
	x	x	
	x		
		x	

The following screen shot shows ONLY working hours from 08:00 AM to 12:00 PM (**ShowNonworkingUnits property is False**) :



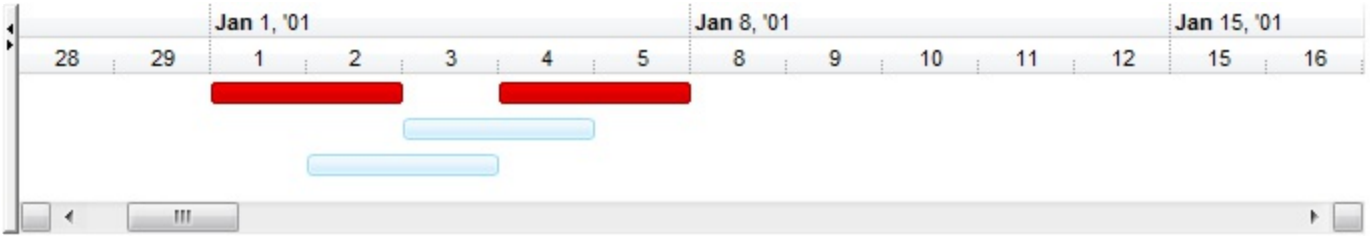
The following screen shot shows with a different pattern the non-working hours (**ShowNonworkingUnits property is True**) :



1. The following screen shot shows the non-working units when the ShowNonworkingUnits property is True and UnitWidthNonworking property is 0 (by default):

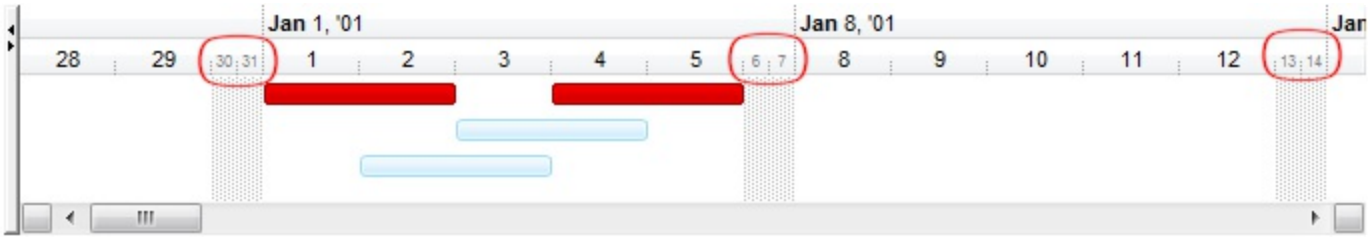
The days 30, 31, 6, 7, ... are shown using the same width, when the ChartWidthNonworking property is 0

2. The following screen shot hides the non-working units when the ShowNonworkingUnits property is False and ShowNonworkingDates property is False:



The days 30, 31, 6, 7, 13, 14, ... are not shown if the ShowNonworkingUnits property is False.

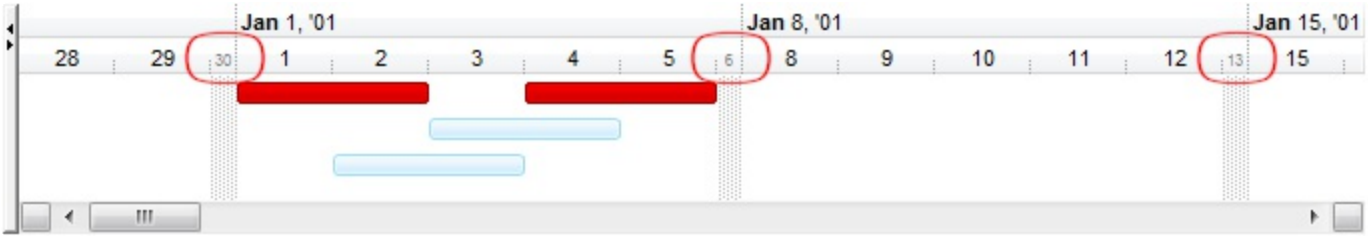
3. The following screen shot shows the non-working units when the ShowNonworkingUnits property is True and UnitWidthNonworking property is 12 (positive value):



The days 30, 31, 6, 7, 13, 14, ... are shown using the a different width, when the

ChartWidthNonworking property is positive.

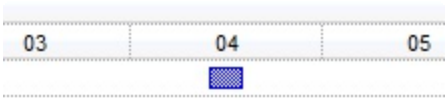
4. The following screen shot shows the non-working units when the ShowNonworkingUnits property is True and UnitWidthNonworking property is -12 (negative value):



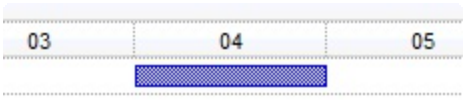
The days 30, 31, 6, 7, 13, 14, ... are shown as a single non-working unit, when the ChartWidthNonworking property is negative.

5. If the chart displays days and use the NonworkingHours property to specify the non-working hours, but want to display the bars using the entire space of the day you need to specify the ShowNonworkingUnits property on False, the Unit property on exHour, Count property on 24 (so actually it you simulate an entire day)

By default a bar that starts on #1/4/2002 10:00# and ends on #1/4/2002 14:00# looks as follows:



instead if the Unit property on exHour, Count property on 24 for the level that displays days and specify the ShowNonworkingUnits property on False we get this:



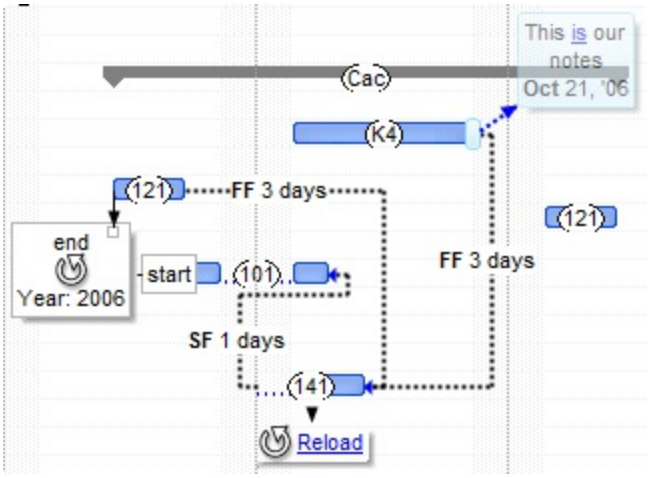
property Chart.ShowNotes as Boolean

Specifies whether all notes or boxes are shown or hidden.

Type	Description
Boolean	A Boolean expression that specifies whether the chart shows or hides the notes associated with DATEs ot BARs.

By default, the ShowNotes property is True. Use the ShowNotes property to show or hide the notes in the chart area. The Clear method removes all notes in the control. Use the Remove method to remove a specific Note in the collection. Use the Add method to add new notes to the chart area. Use the NoteFromPoint property to access the note from the cursor. Use the Visible property to show or hide a specific note, or use the PartVisible property to specify whether the start or ending part of the note is visible or hidden. The ShowLink property specifies whether the link between parts of the notes is visible or hidden. The ClipTo property specifies the limits of the notes within the chart.

The following screen shot shows notes associated with a BAR:



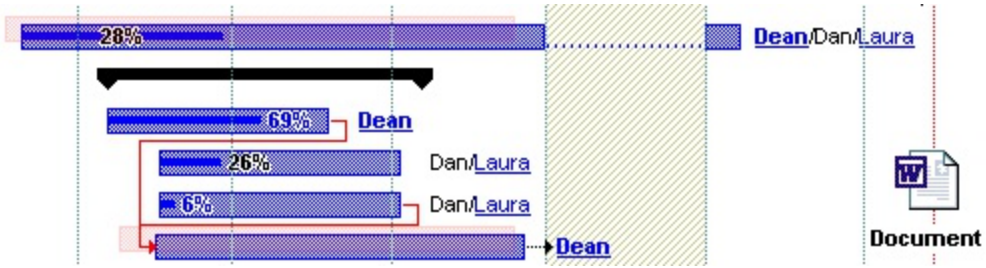
property Chart.ShowTransparentBars as Long

Gets or sets a value that indicates percent of the transparency to display the bars.

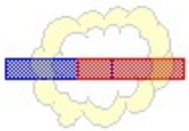
Type	Description
Long	A Long expression, from 0 to 100, that indicates the percent of transparency that's used to paint the bars. 0 means opaque, 100 means hidden, or 100% transparent. 50 means semi-transparent.

By default, the ShowTransparentBars property is 0, which means that the bars are opaque. Use the ShowTransparentBars property to draw all bars using a semi-transparent color. Use the ShowTransparentBars property to draw the intersection of bars using a semi-transparent color.

The following screen shot shows only few items that are shown using a semi-transparent color (the bars in red). Use the ItemBar(exBarTransparent) property to specify the percent of the transparency to display a specified bar. Use the ItemBar(exBarOffset) property to specify the the vertical offset to show the bar.



The following screen shot shows two bars when the ShowTransparentBars property is 0:



The following screen shot shows two bars when the ShowTransparentBars property is 60, which means 60% transparent:

property Chart.SplitPaneWidth as String

Specifies the width of split panels, separated by comma.

Type	Description
String	A String expression that specifies the width (in pixels) of split panels, separated by comma.

By default, the SplitPaneWidth property is empty. The SplitPaneWidth property specifies the width of split panels, separated by comma. The AllowSplitPane property specifies whether the chart panel supports splitting. Once the AllowSplitPane property is set, the user can click the lower-right split bar, and drag to a new position to add a new split to the current chart. The Background(exCSplitBar) property specifies the visual appearance of the chart's split bar. The exDisableSplitPane flag of OnResizeControl property specifies whether the user can drag the split bar at runtime. The ChartStartChanging(exSplitPaneChange) / ChartEndChanging(exSplitPaneChange) events notify that the user splits/resizes the chart's panel into multiple-views.

method Chart.StartBlockUndoRedo ()

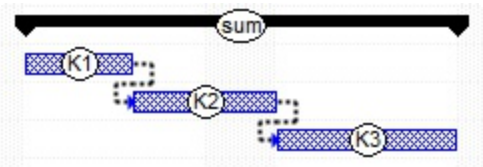
Starts recording the UI operations as a block of undo/redo operations.

Type	Description

The StartBlockUndoRedo method starts recording the UI operations as a block on undo/redo operations. The method has effect only if the AllowUndoRedo property is True. The EndBlockUndoRedo method collects all undo/redo operations since StartBlockUndoRedo method was called and add them to the undo/redo queue as a block. This way the next call on a Undo operation, the entire block is restored, so all UI operations are restored. For instance, if you have a procedure that moves several bars, and want all of them being grouped, you can use StartBlockUndoRedo to start recording the operations as a block, and call the EndBlockUndoRedo when procedure ends, so next call of an undo operation the bars are restored to their original position. The EndBlockUndoRedo method must be called the same number of times as the StartBlockUndoRedo method was called. For instance, if you have called the StartBlockUndoRedo twice the EndBlockUndoRedo method must be called twice too, and the collected operations are added to the chart's queue of undo/redo operations at the end.

The chart fires the ChartStartChanging event when the user starts an UI operation, for instance, moving a bar. The ChartEndChanging event notifies your application once the user operation on the chart ends. By default, each undo/redo operation is added sequentially as they occur. You can call the StartBlockUndoRedo method during the ChartStartChanging event so all operations that are about to begin will be as a block when calling the EndBlockUndoRedo during the ChartEndChanging event. For instance, if a bar is related to multiple bars using grouping options, so if a bar is moved other bars must be moved, the undo/redo operations are added sequentially as they appear. So calling the Undo action will restore moving a bar once at the time. Using the StartBlockUndoRedo/EndBlockUndoRedo methods you can control the block of undo/redo operations being grouped in a block, so next time the Undo/Redo operation is performed, the entire block of operations is performed or restored at once. For instance, the SchedulePDM method performs multiple operations during bars, so all of them are grouped as a block.

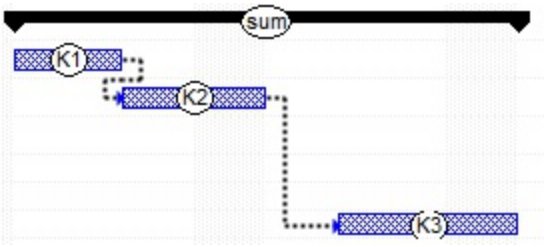
For instance, we we have the following chart:



In this case the, the K1, K2 and K3 bars are grouped, so moving any bar will result in moving relative bars.

The following screen shot shows the chart after moving the bar K3 to a new position as

well as a to a new parent,



so the undo/redo queue looks like:

StartBlock MoveBar;1;sum
MoveBar;2;K4
MoveBar;3;K3
EndBlock
ParentChangeBar;2;K3

In this case, we need to press twice the CTRL + Z to restore back the chart as it was before moving the bar K3.

Instead if we are using the StartBlockUndoRedo and EndBlockUndoRedo methods as follow:

```
Private Sub G2antt1_ChartStartChanging(ByVal Operation As
EXG2ANTTLibCtl.BarOperationEnum)
   G2antt1.Chart.StartBlockUndoRedo
End Sub

Private Sub G2antt1_ChartEndChanging(ByVal Operation As
EXG2ANTTLibCtl.BarOperationEnum)
   G2antt1.Chart.EndBlockUndoRedo
End Sub
```

We have the undo/redo queue as follows (if we perform the same operation):

StartBlock
MoveBar;1;sum
MoveBar;2;K4
MoveBar;3;K3
ParentChangeBar;2;K3
EndBlock

In this case, we need to press only once the CTRL + Z to restore back the chart as it was before moving the bar K3.

property Chart.StartPrintDate as Variant

Retrieves or sets a value that indicates the printing start date.

Type	Description
Variant	A DATE expression that specifies the ending date to print the chart. The get method always retrieves a DATE expression. When calling the set method of the StartPrintDate property, it can be a string, a DATE or any other expression that can be converted to a date.

The StartPrintDate property indicates the date the chart starts, when:

- printing the control's content using the exprint component
- coping the control's content using the CopyTo method (since 22.0.1.5)

By default, the StartPrintDate property computes the required start date so the entire chart is displayed, *if the StartPrintDate was not specified before*. For instance, if you set the StartPrintDate property on "Jan 1 2001", the StartPrintDate property returns the "Jan 1 2001" date and does not compute the required start date. If you have specified a value for the StartPrintDate but you still need to get the required start date being computed, set the StartPrintDate property on 0, and calling the next method get of StartPrintDate property computes the required start date to print the chart. Use the EndPrintDate property to specify the end date to print the chart. Use the CountVisibleUnits property to count the number of units within the specified range. Use the FirstVisibleDate property to specify the first visible date of the chart when displaying on the screen.

property Chart.TimeZoneFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Variant

Retrieves the time-zone from the cursor.

Type	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in client coordinates.
Variant	A String expression that indicates the key of the date-time zone from the cursor, or empty value if there is no date-time zone.

The TimeZoneFromPoint property retrieves the key of the time-zone from the cursor. A zone can be used to highlight a range of dates, specifying the start and end zone. The TimeZoneInfo property retrieves information about the time-zone giving its key. The DateFromPoint property gets the date from point. Use the MarkTimeZone method to highlight different time-zones. Use the RemoveTimeZone method to delete the time zone being added previously using the MarkTimeZone method. The MarkTodayColor property specifies the color to mark the today date. Use the SelectDate property to select a date by clicking the chart's header. Use the MarkNowColor property to show a vertical bar that indicates the current date-time in the control's chart, from seconds to seconds, minutes, and so on. **If the X parameter is -1 and Y parameter is -1 the TimeZoneFromPoint property determines the key of the date-time zone from the cursor**.

property Chart.TimeZoneInfo (Key as Variant) as Variant

Retrieves information about the time-zone giving its key.

Type	Description
Key as Variant	A String expression that specifies the key of the date-time zone to access the information
Variant	A safe array of 5 elements that indicates the Key, Start, End, Color and Options parameters being using at the time MarkTimeZone method is invoked.

The TimeZoneInfo property can be used to access the information about a date-time zone. A zone can be used to highlight a range of dates, specifying the start and end zone. Use the MarkTimeZone method to highlight different time-zones. Use the RemoveTimeZone method to delete the time zone being added previously using the MarkTimeZone method. The TimeZoneFromPoint property retrieves the key of the time-zone from the cursor.

The TimeZoneInfo property retrieves a vector of 5 fields as follows:

- Key - A String expression that specifies a key to identify the zone.
- Start - A DATE expression that specifies the starting point for the zone.
- End - A DATE expression that specifies the ending point for the zone.
- Color - A Color expression that specifies the date-time zone to be highlighted.
- Options - A String expression that specifies options to mark the zone.

For instance, the following VB sample displays the stating point of the date-time zone from the cursor:

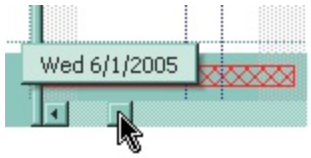
```
Private Sub G2antt1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Dim k As Variant
   k = G2antt1.Chart.TimeZoneFromPoint(-1, -1)
   If (Not IsEmpty(k)) Then
      Debug.Print "Start at: " & G2antt1.Chart.TimeZoneInfo(k)(1)
   End If
End Sub
```

property Chart.ToolTip as String

Retrieves or sets a value that indicates the format of the tooltip being shown while the user scrolls the chart.

Type	Description
String	A String expression that includes the format of the tooltip.

By default, the ToolTip property is " <%ddd%> <%m%>/<%d%>/<%yyyy%> ". The ToolTip property specifies the tooltip that shows up when the user scrolls the chart. The Tooltip message shows the value for the chart's FirstVisibleDate property. If the ToolTip property is empty, the control doesn't show up the tooltip when the user scrolls the chart instead it does directly the scrolling. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. The ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipWidth property to specify the width of the tooltip window. Use the FormatDate property to format a date. Use the MonthNames property to specify the name of the months in the year. The WeekDays property retrieves or sets a value that indicates the list of names for each week day, separated by space. Use the Zoom method to zoom the chart to a specified interval of dates. Use the AMPM property to specify the name of the AM and PM indicators. The Label property specifies a predefined label for a specified unit. Use the ScrollBar property to show or hide the chart's scroll bar. Use the ItemBar(exBarToolTip) property to assign a tooltip to a bar. Use the OverviewToolTip property retrieves or sets a value that indicates the format of the tooltip being shown while the cursor hovers the chart's overview area.



The ToolTip property supports the following built-in tags:

- <%**d**%> - Day of the month in one or two numeric digits, as needed (1 to 31).
- <%**dd**%> - Day of the month in two numeric digits (01 to 31).
- <%**d1**%> - First letter of the weekday (S to S). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**d2**%> - First two letters of the weekday (Su to Sa). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**d3**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays

property to specify the name of the days in the week)
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**ddd**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays property to specify the name of the days in the week). You can use the <%**loc_ddd**%> that indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%**loc_ddd**%> - Indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%**dddd**%> - Full name of the weekday (Sunday to Saturday). (Use the WeekDays property to specify the name of the days in the week). You can use the <%**loc_dddd**%> that indicates day of week as its full name using the current user regional and language settings.
- <%**loc_dddd**%> - Indicates day of week as its full name using the current user regional and language settings.
- <%**i**%> - Displays the number instead the date. For instance, you can display numbers as 1000, 1001, 1002, 1003, instead dates. (the valid range is from -647,434 to 2,958,465)
- <%**w**%> - Day of the week (1 to 7).
- <%**ww**%> - Week of the year (1 to 53).
- <%**m**%> - Month of the year in one or two numeric digits, as needed (1 to 12).
- <%**mr**%> - Month of the year in Roman numerals, as needed (I to XII).
- <%**mm**%> - Month of the year in two numeric digits (01 to 12).
- <%**m1**%> - First letter of the month (J to D). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**m2**%> - First two letters of the month (Ja to De). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**m3**%> - First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**mmm**%> - First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmm**%> that indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%**loc_mmm**%> - Indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%**mmmm**%> - Full name of the month (January to December). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmmm**%> that indicates month as its full name using the current user regional

and language settings.

- <%**loc_mmmm**%> - Indicates month as its full name using the current user regional and language settings.
- <%**q**%> - Date displayed as the quarter of the year (1 to 4).
- <%**y**%> - Number of the day of the year (1 to 366).
- <%**yy**%> - Last two digits of the year (01 to 99).
- <%**yyyy**%> - Full year (0100 to 9999).
- <%**hy**%> - Date displayed as the half of the year (1 to 2).
- <%**loc_gg**%> - Indicates period/era using the current user regional and language settings.
- <%**loc_sdate**%> - Indicates the date in the short format using the current user regional and language settings.
- <%**loc_ldate**%> - Indicates the date in the long format using the current user regional and language settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user regional and language settings (/).
- <%**h**%> - Hour in one or two digits, as needed (0 to 23).
- <%**hh**%> - Hour in two digits (00 to 23).
- <%**h12**%> - Hour in 12-hour time format, in one or two digits - [0(12),11]
- <%**hh12**%> - hour in 12-hour time format, in two digits - [00(12),11]
- <%**n**%> - Minute in one or two digits, as needed (0 to 59).
- <%**nn**%> - Minute in two digits (00 to 59).
- <%**s**%> - Second in one or two digits, as needed (0 to 59).
- <%**ss**%> - Second in two digits (00 to 59).
- <%**AM/PM**%> - Twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate. (Use the AMPM property to specify the name of the AM and PM indicators). You can use the <%**loc_AM/PM**%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%**loc_A/P**%> that indicates the one character time marker such as A or P using the current user regional and language settings
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user regional and language settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user regional and language settings.
- <%**loc_time**%> - Indicates the time using the current user regional and language settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%**loc_tsep**%> - indicates the time separator using the current user regional and language settings (:)
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.

- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

The following tags are displayed based on the user's Regional and Language Options:

- <%**loc_sdate**%> - Indicates the date in the short format using the current user settings.
- <%**loc_ldate**%> - Indicates the date in the long format using the current user settings.
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**loc_ddd**%> - Indicates day of week as a three-letters abbreviation using the current user settings.
- <%**loc_dddd**%> - Indicates day of week as its full name using the current user settings.
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**loc_mmm**%> - Indicates month as a three-letters abbreviation using the current user settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user settings.
- <%**loc_gg**%> - Indicates period/era using the current user settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user settings.
- <%**loc_time**%> - Indicates the time using the current user settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user settings.
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user settings.
- <%**loc_tsep**%> - Indicates the time separator using the current user settings

- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

method Chart.Undo ()

Performs the last Undo operation.

Type	Description

Call the Undo method to Undo the last chart operation. The Undo method have effect only if the AllowUndoRedo property is True. The CTRL+Z performs the last undo operation, while the CTRL+Y redoes the next action in the chart's Redo queue. The Redo redoes the next action in the chart's redo queue. The CanUndo property retrieves a value that indicates whether the chart may perform the last Undo operation. The CanRedo property retrieves a value that specifies whether the chart can execute the next operation in the chart's Redo queue. The ChartStartChanging(exUndo/exRedo) / ChartEndChanging(exUndo/exRedo) event notifies your application whenever an Undo/Redo operation is performed.

The records of the Undo/Redo queue may contain actions in the following format:

- "**AddBar**;ITEMINDEX;KEY", indicates that a new bar has been created
- "**RemoveBar**;ITEMINDEX;KEY", indicates that a bar has been removed
- "**MoveBar**;ITEMINDEX;KEY", indicates that a bar has been moved or resized
- "**PercentChange**;ITEMINDEX;KEY", indicates that the bar's percent has been changed
- "**UpdateBar**;ITEMINDEX;KEY", indicates that one or more properties of the bar has been updated (ItemBar property, this operation can be added only using the StartUpdateBar / EndUpdateBar methods)
- "**ParentChangeBar**;ITEMINDEX;KEY", indicates that the bar's parent has been changed
- "**GroupBars**;ITEMINDEXA;KEYA;STARTA;ITEMINDEXB;KEYB;STARTB", specifies that two bars has been grouped
- "**UngroupBars**;ITEMINDEXA;KEYA;STARTA;ITEMINDEXB;KEYB;STARTB", specifies that two bars has been ungrouped
- "**DefineSummaryBars**;SUMMARYITEMINDEX;SUMMARYKEY;ITEMINDEX;KEY", indicates that a bar has been defined as a child of a summary bar
- "**UndefineSummaryBars**;SUMMARYITEMINDEX;SUMMARYKEY;ITEMINDEX;KEY", indicates that a bar has been removed from the summary bar's children
- "**AddLink**;KEY", indicates that a new link has been created
- "**RemoveLink**;KEY", indicates that a link has been removed
- "**UpdateLink**;KEY", specifies that one of more properties of the link has been updated (Link property, this operation can be added only using the StartUpdateLink / EndUpdateLink methods)

The records of the Undo/Redo queue may contain actions in the following format (available starting from 23.0):

- "**AddItem**;ITEMINDEX", indicates that a new item has been created
- "**RemoveItem**;ITEMINDEX", indicates that an item has been removed
- "**ChangeItemPos**;ITEMINDEX", indicates that an item changes its position or / and parent
- "**ChangeCellValue**;ITEMINDEX;CELLINDEX", indicates that the cell's value has been changed
- "**ChangeCellState**;ITEMINDEX;CELLINDEX", indicates that the cell's state has been changed

Also, the Undo/Redo queue may include:

- "**StartBlock**", specifies that a block of operations begins
- "**EndBlock**", specifies that a block of operations ends

The UndoListAction property lists the Undo actions that can be performed in the chart. Use the UndoRemoveAction method to remove the last actions from the undo queue.

property Chart.UndoListAction ([Action as Variant], [Count as Variant]) as String

Lists the Undo actions that can be performed in the chart.

Type	Description
Action as Variant	[optional] A long expression that specifies the action being listed. If missing or -1, all actions are listed. The Action parameter can be one of the following: • exChartUndoRedoAddBar(0) ~ "**AddBar**;ITEMINDEX;KEY", indicates that a new bar has been created • exChartUndoRedoRemoveBar(1) ~ "**RemoveBar**;ITEMINDEX;KEY", indicates that a bar has been removed • exChartUndoRedoMoveBar(2) ~ "**MoveBar**;ITEMINDEX;KEY", indicates that a bar has been moved or resized • exChartUndoRedoPercentChange(3) ~ "**PercentChange**;ITEMINDEX;KEY", indicates that the bar's percent has been changed • exChartUndoRedoUpdateBar(4) ~ "**UpdateBar**;ITEMINDEX;KEY", indicates that one or more properties of the bar has been updated (ItemBar property, this operation can be added only using the StartUpdateBar / EndUpdateBar methods) • exChartUndoRedoParentChangeBar(5) ~ "**ParentChangeBar**;ITEMINDEX;KEY", indicates that the bar's parent has been changed • exChartUndoRedoGroupBars(6) ~ "**GroupBars**;ITEMINDEXA;KEYA;STARTA;ITEMINDE specifies that two bars has been grouped • exChartUndoRedoUngroupBars(7) ~ "**UngroupBars**;ITEMINDEXA;KEYA;STARTA;ITEMIND specifies that two bars has been ungrouped • exChartUndoRedoDefineSummaryBars(8) ~ "**DefineSummaryBars**;SUMMARYITEMINDEX;SUMM indicates that a bar has been defined as a child of a summary bar • exChartUndoRedoUndefineSummaryBars(9) ~ "**UndefineSummaryBars**;SUMMARYITEMINDEX;SUM

	indicates that a bar has been removed from the summary bar's children • exChartUndoRedoAddLink(10) ~ "**AddLink**;KEY", indicates that a new link has been created • exChartUndoRedoRemoveLink(11) ~ "**RemoveLink**;KEY", indicates that a link has been removed • exChartUndoRedoUpdateLink(12) ~ "**UpdateLink**;KEY", specifies that one of more properties of the link has been updated (Link property, this operation can be added only using the StartUpdateLink / EndUpdateLink methods) • exListUndoRedoAddItem(13) ~ "**AddItem**;ITEMINDEX", indicates that a new item has been created • exListUndoRedoRemoveItem(14) ~ "**RemoveItem**;ITEMINDEX", indicates that an item has been removed • exListUndoRedoChangeItemPos(15) ~ "**ChangeItemPos**;ITEMINDEX", indicates that an item changes its position or / and parent • exListUndoRedoChangeCellValue(16) ~ "**ChangeCellValue**;ITEMINDEX;CELLINDEX", indicates that the cell's value has been changed • exListUndoRedoChangeCellState(17) ~ "**ChangeCellState**;ITEMINDEX;CELLINDEX", indicates that the cell's state has been changed For instance, UndoListAction(0) shows only AddBar actions in the undo stack.
Count as Variant	[optional] A long expression that indicates the number of actions being listed. If missing or -1, all actions are listed. For instance, UndoListAction(0,1) shows only the last AddBar action being added to the undo stack
String	A String expression that lists the Undo actions that may be performed.

Use the UndoListAction property to show the list of actions that the user may perform by doing Undo operations. The ChartStartChanging(exUndo/exRedo) / ChartEndChanging(exUndo/exRedo) event notifies your application whenever an Undo/Redo operation is performed. For instance, the ChartEndChanging(exUndoRedoUpdate) notifies whether a new operation is added/removed from the undo/redo queue. Use the

UndoRemoveAction method to remove the last actions from the undo queue. The RedoListAction property lists the Redo actions that can be performed in the chart. The CanUndo property specifies whether an undo operation can be performed if CTRL+Z key is pressed.

The records of the Undo/Redo queue may contain actions in the following format:

- "**AddBar**;ITEMINDEX;KEY", indicates that a new bar has been created
- "**RemoveBar**;ITEMINDEX;KEY", indicates that a bar has been removed
- "**MoveBar**;ITEMINDEX;KEY", indicates that a bar has been moved or resized
- "**PercentChange**;ITEMINDEX;KEY", indicates that the bar's percent has been changed
- "**UpdateBar**;ITEMINDEX;KEY", indicates that one or more properties of the bar has been updated (ItemBar property, this operation can be added only using the StartUpdateBar / EndUpdateBar methods)
- "**ParentChangeBar**;ITEMINDEX;KEY", indicates that the bar's parent has been changed
- "**GroupBars**;ITEMINDEXA;KEYA;STARTA;ITEMINDEXB;KEYB;STARTB", specifies that two bars has been grouped
- "**UngroupBars**;ITEMINDEXA;KEYA;STARTA;ITEMINDEXB;KEYB;STARTB", specifies that two bars has been ungrouped
- "**DefineSummaryBars**;SUMMARYITEMINDEX;SUMMARYKEY;ITEMINDEX;KEY", indicates that a bar has been defined as a child of a summary bar
- "**UndefineSummaryBars**;SUMMARYITEMINDEX;SUMMARYKEY;ITEMINDEX;KEY", indicates that a bar has been removed from the summary bar's children
- "**AddLink**;KEY", indicates that a new link has been created
- "**RemoveLink**;KEY", indicates that a link has been removed
- "**UpdateLink**;KEY", specifies that one of more properties of the link has been updated (Link property, this operation can be added only using the StartUpdateLink / EndUpdateLink methods)

The records of the Undo/Redo queue may contain actions in the following format (available starting from 23.0):

- "**AddItem**;ITEMINDEX", indicates that a new item has been created
- "**RemoveItem**;ITEMINDEX", indicates that an item has been removed
- "**ChangeItemPos**;ITEMINDEX", indicates that an item changes its position or / and parent
- "**ChangeCellValue**;ITEMINDEX;CELLINDEX", indicates that the cell's value has been changed
- "**ChangeCellState**;ITEMINDEX;CELLINDEX", indicates that the cell's state has been changed

Also, the Undo/Redo queue may include:

- "**StartBlock**", specifies that a block of operations begins
- "**EndBlock**", specifies that a block of operations ends

Each action is on a single line, and each field is separated by ; character. The lines are separated by "\r\n" characters (vbCrLf in VB).

For instance,

- An AddLink action in the Undo stack means that an Undo operation will perform a RemoveLink action.
- A RemoveLink action in the Undo stack means that an Undo operation will perform an AddLink action.
- An AddLink action in the Redo stack means that a Redo operation will perform an AddLink action.
- A RemoveLink action in the Redo stack means that a Redo operation will perform a RemoveLink action.

Here's a sample format of the UndoListAction property may get:

```
StartBlock
   MoveBar;1;E
   MoveBar;2;E
   MoveBar;3;
   MoveBar;4;
EndBlock
MoveBar;1;
DefineSummaryBars;4;;1;E
AddBar;1;E
DefineSummaryBars;4;;2;E
AddBar;2;E
DefineSummaryBars;4;;3;
AddBar;3;
AddBar;4;
DefineSummaryBars;1;;4;E
GroupBars;3;E;0;4;E;-1
GroupBars;3;E;0;4;E;-1
AddLink;L2
AddBar;4;E
GroupBars;2;;0;3;E;-1
```

```
GroupBars;2;;0;3;E;-1
AddLink;L1
DefineSummaryBars;1;;3;E
AddBar;3;E
DefineSummaryBars;1;;2;
AddBar;2;
AddBar;1;
```

The following VB sample splits the UndoListAction value and adds each action to a listbox control:

```
List1.Clear
Dim s() As String
s = Split(G2antt1.Chart.UndoListAction, vbCrLf)
For i = LBound(s) To UBound(s)
        List1.AddItem s(i)
Next
```

property Chart.UndoRedoQueueLength as Long

Gets or sets the maximum number of Undo/Redo actions that may be stored to the chart's queue.

Type	Description
Long	A Long expression that specifies the length of the Undo/Redo queue. If -1, the queue is unlimited, 0 allows no entries in the Undo/Redo queue.

By default, the UndoRedoQueueLength property is -1. Use the UndoRedoQueueLength property to specify the number of entries that Undo/Redo queue may store. For instance, if the UndoRedoQueueLength property is 1, the control retains only the last chart operation. Changing the UndoRedoQueueLength property may change the current Undo/Redo queue based on the new length. The length being specified, does not affect the blocks in the queue. A block may hold multiple Undo/Redo actions. Use the GroupUndoRedoActions method to group two or more entries in the Undo/Redo queue in a single block, so when a next Undo/Redo operation is performed, multiple actions may occur. For instance, moving several bars in the same time (multiple bars selection) is already recorded as a single block.

method Chart.UndoRemoveAction ([Action as Variant], [Count as Variant])

Removes the last the undo actions that can be performed in the chart.

Type	Description
Action as Variant	[optional] A long expression that specifies the action being removed. If missing or -1, all actions are removed from the undo queue. The Action parameter can be one of the following: • exChartUndoRedoAddBar(0) ~ "**AddBar**;ITEMINDEX;KEY", indicates that a new bar has been created • exChartUndoRedoRemoveBar(1) ~ "**RemoveBar**;ITEMINDEX;KEY", indicates that a bar has been removed • exChartUndoRedoMoveBar(2) ~ "**MoveBar**;ITEMINDEX;KEY", indicates that a bar has been moved or resized • exChartUndoRedoPercentChange(3) ~ "**PercentChange**;ITEMINDEX;KEY", indicates that the bar's percent has been changed • exChartUndoRedoUpdateBar(4) ~ "**UpdateBar**;ITEMINDEX;KEY", indicates that one or more properties of the bar has been updated (ItemBar property, this operation can be added only using the StartUpdateBar / EndUpdateBar methods) • exChartUndoRedoParentChangeBar(5) ~ "**ParentChangeBar**;ITEMINDEX;KEY", indicates that the bar's parent has been changed • exChartUndoRedoGroupBars(6) ~ "**GroupBars**;ITEMINDEXA;KEYA;STARTA;ITEMINDE specifies that two bars has been grouped • exChartUndoRedoUngroupBars(7) ~ "**UngroupBars**;ITEMINDEXA;KEYA;STARTA;ITEMIND specifies that two bars has been ungrouped • exChartUndoRedoDefineSummaryBars(8) ~ "**DefineSummaryBars**;SUMMARYITEMINDEX;SUMM indicates that a bar has been defined as a child of a summary bar • exChartUndoRedoUndefineSummaryBars(9) ~

"**UndefineSummaryBars**;SUMMARYITEMINDEX;SUM indicates that a bar has been removed from the summary bar's children

- exChartUndoRedoAddLink(10) ~ "**AddLink**;KEY", indicates that a new link has been created
- exChartUndoRedoRemoveLink(11) ~ "**RemoveLink**;KEY", indicates that a link has been removed
- exChartUndoRedoUpdateLink(12) ~ "**UpdateLink**;KEY", specifies that one of more properties of the link has been updated (Link property, this operation can be added only using the StartUpdateLink / EndUpdateLink methods)
- exListUndoRedoAddItem(13) ~ "**AddItem**;ITEMINDEX", indicates that a new item has been created
- exListUndoRedoRemoveItem(14) ~ "**RemoveItem**;ITEMINDEX", indicates that an item has been removed
- exListUndoRedoChangeItemPos(15) ~ "**ChangeItemPos**;ITEMINDEX", indicates that an item changes its position or / and parent
- exListUndoRedoChangeCellValue(16) ~ "**ChangeCellValue**;ITEMINDEX;CELLINDEX", indicates that the cell's value has been changed
- exListUndoRedoChangeCellState(17) ~ "**ChangeCellState**;ITEMINDEX;CELLINDEX", indicates that the cell's state has been changed

For instance, UndoRemoveAction(0) removes only AddBar actions in the undo stack.

Count as Variant	[optional] A long expression that indicates the number of actions to be removed. If missing or -1, all actions are removed. For instance, UndoRemoveAction(0,1) removes only the last AddBar action from the undo stack

Use the UndoRemoveAction method to remove the last action from the undo queue. Use the UndoRemoveAction() (with no parameters) to remove all undo actions. The UndoListAction property retrieves the list of actions that an undo operation can perform. The RedoRemoveAction method removes the first action to be performed if the Redo method is invoked. For instance, let's say that during the AddLink event, you decide that the link being added should be removed, and so, you do not need to record the last RemoveLink action.

In other words, you need to call the UndoRemoveAction(exChartUndoRedoRemoveLink, 1) removes the last RemoveLink action from the undo stack.

The records of the Undo/Redo queue may contain actions in the following format:

- "**AddBar**;ITEMINDEX;KEY", indicates that a new bar has been created
- "**RemoveBar**;ITEMINDEX;KEY", indicates that a bar has been removed
- "**MoveBar**;ITEMINDEX;KEY", indicates that a bar has been moved or resized
- "**PercentChange**;ITEMINDEX;KEY", indicates that the bar's percent has been changed
- "**UpdateBar**;ITEMINDEX;KEY", indicates that one or more properties of the bar has been updated (ItemBar property, this operation can be added only using the StartUpdateBar / EndUpdateBar methods)
- "**ParentChangeBar**;ITEMINDEX;KEY", indicates that the bar's parent has been changed
- "**GroupBars**;ITEMINDEXA;KEYA;STARTA;ITEMINDEXB;KEYB;STARTB", specifies that two bars has been grouped
- "**UngroupBars**;ITEMINDEXA;KEYA;STARTA;ITEMINDEXB;KEYB;STARTB", specifies that two bars has been ungrouped
- "**DefineSummaryBars**;SUMMARYITEMINDEX;SUMMARYKEY;ITEMINDEX;KEY", indicates that a bar has been defined as a child of a summary bar
- "**UndefineSummaryBars**;SUMMARYITEMINDEX;SUMMARYKEY;ITEMINDEX;KEY", indicates that a bar has been removed from the summary bar's children
- "**AddLink**;KEY", indicates that a new link has been created
- "**RemoveLink**;KEY", indicates that a link has been removed
- "**UpdateLink**;KEY", specifies that one of more properties of the link has been updated (Link property, this operation can be added only using the StartUpdateLink / EndUpdateLink methods)

The records of the Undo/Redo queue may contain actions in the following format (available starting from 23.0):

- "**AddItem**;ITEMINDEX", indicates that a new item has been created
- "**RemoveItem**;ITEMINDEX", indicates that an item has been removed
- "**ChangeItemPos**;ITEMINDEX", indicates that an item changes its position or / and parent
- "**ChangeCellValue**;ITEMINDEX;CELLINDEX", indicates that the cell's value has been changed
- "**ChangeCellState**;ITEMINDEX;CELLINDEX", indicates that the cell's state has been changed

Also, the Undo/Redo queue may include:

- "**StartBlock**", specifies that a block of operations begins
- "**EndBlock**", specifies that a block of operations ends

The following samples shows preventing adding the AddLink and RemoveLink undo operations when AddLink event occurs.

VBA (MS Access, Excell...)

```
' AddLink event - Occurs when the user links two bars using the mouse.
Private Sub G2antt1_AddLink(ByVal LinkKey As String)
   With G2antt1
      .Items.RemoveLink LinkKey
      .Chart.UndoRemoveAction 7,1
      .Chart.UndoRemoveAction 8,1
      Debug.Print(.Chart.UndoListAction())
   End With
End Sub

' ChartEndChanging event - Occurs after the chart has been changed.
Private Sub G2antt1_ChartEndChanging(ByVal Operation As Long)
   With G2antt1
      Debug.Print(.Chart.UndoListAction())
   End With
End Sub

With G2antt1
   .BeginUpdate
   .Columns.Add "Tasks"
   With .Chart
      .AllowUndoRedo = True
      .FirstVisibleDate = #6/20/2005#
      .AllowLinkBars = True
      .LevelCount = 2
      .PaneWidth(0) = 48
   End With
   With .Items
      .AddBar .AddItem("Task 1"),"Task",#6/21/2005#,#6/25/2005#,""
      .AddBar .AddItem("Task 2"),"Task",#6/28/2005#,#7/2/2005#,""
```

```
    End With
    .EndUpdate
End With
```

VB6

```
' AddLink event - Occurs when the user links two bars using the mouse.
Private Sub G2antt1_AddLink(ByVal LinkKey As String)
    With G2antt1
        .Items.RemoveLink LinkKey
        .Chart.UndoRemoveAction 7,1
        .Chart.UndoRemoveAction 8,1
        Debug.Print(.Chart.UndoListAction())
    End With
End Sub

' ChartEndChanging event - Occurs after the chart has been changed.
Private Sub G2antt1_ChartEndChanging(ByVal Operation As
EXG2ANTTLibCtl.BarOperationEnum)
    With G2antt1
        Debug.Print(.Chart.UndoListAction())
    End With
End Sub

With G2antt1
    .BeginUpdate
    .Columns.Add "Tasks"
    With .Chart
        .AllowUndoRedo = True
        .FirstVisibleDate = #6/20/2005#
        .AllowLinkBars = True
        .LevelCount = 2
        .PaneWidth(0) = 48
    End With
    With .Items
        .AddBar .AddItem("Task 1"),"Task",#6/21/2005#,#6/25/2005#,""
        .AddBar .AddItem("Task 2"),"Task",#6/28/2005#,#7/2/2005#,""
```

```
    End With
    .EndUpdate
End With
```

VB.NET

```
' AddLink event - Occurs when the user links two bars using the mouse.
Private Sub Exg2antt1_AddLink(ByVal sender As System.Object,ByVal LinkKey As String)
Handles Exg2antt1.AddLink
    With Exg2antt1
        .Items.RemoveLink(LinkKey)
        .Chart.UndoRemoveAction(7,1)
        .Chart.UndoRemoveAction(8,1)
        Debug.Print(.Chart.get_UndoListAction())
    End With
End Sub

' ChartEndChanging event - Occurs after the chart has been changed.
Private Sub Exg2antt1_ChartEndChanging(ByVal sender As System.Object,ByVal Operation
As exontrol.EXG2ANTTLib.BarOperationEnum) Handles Exg2antt1.ChartEndChanging
    With Exg2antt1
        Debug.Print(.Chart.get_UndoListAction())
    End With
End Sub

With Exg2antt1
    .BeginUpdate()
    .Columns.Add("Tasks")
    With .Chart
        .AllowUndoRedo = True
        .FirstVisibleDate = #6/20/2005#
        .AllowLinkBars = True
        .LevelCount = 2
        .set_PaneWidth(False,48)
    End With
    With .Items
        .AddBar(.AddItem("Task 1"),"Task",#6/21/2005#,#6/25/2005#,"")
```

```
    .AddBar(.AddItem("Task 2"),"Task",#6/28/2005#,#7/2/2005#,"")
  End With
  .EndUpdate()
End With
```

VB.NET for /COM

```
' AddLink event - Occurs when the user links two bars using the mouse.
Private Sub AxG2antt1_AddLink(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_AddLinkEvent) Handles AxG2antt1.AddLink
  With AxG2antt1
    .Items.RemoveLink(e.linkKey)
    .Chart.UndoRemoveAction(7,1)
    .Chart.UndoRemoveAction(8,1)
    Debug.Print(.Chart.UndoListAction())
  End With
End Sub

' ChartEndChanging event - Occurs after the chart has been changed.
Private Sub AxG2antt1_ChartEndChanging(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_ChartEndChangingEvent) Handles
AxG2antt1.ChartEndChanging
  With AxG2antt1
    Debug.Print(.Chart.UndoListAction())
  End With
End Sub

With AxG2antt1
  .BeginUpdate()
  .Columns.Add("Tasks")
  With .Chart
    .AllowUndoRedo = True
    .FirstVisibleDate = #6/20/2005#
    .AllowLinkBars = True
    .LevelCount = 2
    .PaneWidth(False) = 48
  End With
```

```
    With .Items
        .AddBar(.AddItem("Task 1"),"Task",#6/21/2005#,#6/25/2005#,"")
        .AddBar(.AddItem("Task 2"),"Task",#6/28/2005#,#7/2/2005#,"")
    End With
    .EndUpdate()
End With
```

C++

```
// AddLink event - Occurs when the user links two bars using the mouse.
void OnAddLinkG2antt1(LPCTSTR LinkKey)
{
    /*
        Copy and paste the following directives to your header file as
        it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'
        #import <ExG2antt.dll>
        using namespace EXG2ANTTLib;
    */
    EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
    spG2antt1->GetItems()->RemoveLink(LinkKey);
    spG2antt1->GetChart()->UndoRemoveAction(long(7),long(1));
    spG2antt1->GetChart()->UndoRemoveAction(long(8),long(1));
    OutputDebugStringW(spG2antt1->GetChart()-
>GetUndoListAction(vtMissing,vtMissing));
}

// ChartEndChanging event - Occurs after the chart has been changed.
void OnChartEndChangingG2antt1(long Operation)
{
    EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
    OutputDebugStringW(spG2antt1->GetChart()-
>GetUndoListAction(vtMissing,vtMissing));
}

EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
```

```
>GetControlUnknown();
spG2antt1->BeginUpdate();
spG2antt1->GetColumns()->Add(L"Tasks");
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutAllowUndoRedo(VARIANT_TRUE);
   var_Chart->PutFirstVisibleDate("6/20/2005");
   var_Chart->PutAllowLinkBars(VARIANT_TRUE);
   var_Chart->PutLevelCount(2);
   var_Chart->PutPaneWidth(VARIANT_FALSE,48);
EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems();
   var_Items->AddBar(var_Items->AddItem("Task
1"),"Task","6/21/2005","6/25/2005","",vtMissing);
   var_Items->AddBar(var_Items->AddItem("Task
2"),"Task","6/28/2005","7/2/2005","",vtMissing);
spG2antt1->EndUpdate();
```

C#

```
// AddLink event - Occurs when the user links two bars using the mouse.
private void exg2antt1_AddLink(object sender,string LinkKey)
{
   exg2antt1.Items.RemoveLink(LinkKey);
   exg2antt1.Chart.UndoRemoveAction(7,1);
   exg2antt1.Chart.UndoRemoveAction(8,1);
   System.Diagnostics.Debug.Print(exg2antt1.Chart.get_UndoListAction(null,null));
}
//this.exg2antt1.AddLink += new
exontrol.EXG2ANTTLib.exg2antt.AddLinkEventHandler(this.exg2antt1_AddLink);

// ChartEndChanging event - Occurs after the chart has been changed.
private void exg2antt1_ChartEndChanging(object
sender,exontrol.EXG2ANTTLib.BarOperationEnum Operation)
{
   System.Diagnostics.Debug.Print(exg2antt1.Chart.get_UndoListAction(null,null));
}
//this.exg2antt1.ChartEndChanging += new
exontrol.EXG2ANTTLib.exg2antt.ChartEndChangingEventHandler(this.exg2antt1_Ch
```

```
exg2antt1.BeginUpdate();
exg2antt1.Columns.Add("Tasks");
exontrol.EXG2ANTTLib.Chart var_Chart = exg2antt1.Chart;
   var_Chart.AllowUndoRedo = true;
   var_Chart.FirstVisibleDate = Convert.ToDateTime("6/20/2005");
   var_Chart.AllowLinkBars = true;
   var_Chart.LevelCount = 2;
   var_Chart.set_PaneWidth(false,48);
exontrol.EXG2ANTTLib.Items var_Items = exg2antt1.Items;
   var_Items.AddBar(var_Items.AddItem("Task
1"),"Task",Convert.ToDateTime("6/21/2005"),Convert.ToDateTime("6/25/2005"),"",null);
   var_Items.AddBar(var_Items.AddItem("Task
2"),"Task",Convert.ToDateTime("6/28/2005"),Convert.ToDateTime("7/2/2005"),"",null);
exg2antt1.EndUpdate();
```

C# for /COM

```
// AddLink event - Occurs when the user links two bars using the mouse.
private void axG2antt1_AddLink(object sender,
AxEXG2ANTTLib._IG2anttEvents_AddLinkEvent e)
{
   axG2antt1.Items.RemoveLink(e.linkKey);
   axG2antt1.Chart.UndoRemoveAction(7,1);
   axG2antt1.Chart.UndoRemoveAction(8,1);
   System.Diagnostics.Debug.Print(axG2antt1.Chart.get_UndoListAction(null,null));
}
//this.axG2antt1.AddLink += new
AxEXG2ANTTLib._IG2anttEvents_AddLinkEventHandler(this.axG2antt1_AddLink);

// ChartEndChanging event - Occurs after the chart has been changed.
private void axG2antt1_ChartEndChanging(object sender,
AxEXG2ANTTLib._IG2anttEvents_ChartEndChangingEvent e)
{
   System.Diagnostics.Debug.Print(axG2antt1.Chart.get_UndoListAction(null,null));
}
```

```
//this.axG2antt1.ChartEndChanging += new
AxEXG2ANTTLib._IG2anttEvents_ChartEndChangingEventHandler(this.axG2antt1_Ch



axG2antt1.BeginUpdate();
axG2antt1.Columns.Add("Tasks");
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.AllowUndoRedo = true;
   var_Chart.FirstVisibleDate = Convert.ToDateTime("6/20/2005");
   var_Chart.AllowLinkBars = true;
   var_Chart.LevelCount = 2;
   var_Chart.set_PaneWidth(false,48);
EXG2ANTTLib.Items var_Items = axG2antt1.Items;
   var_Items.AddBar(var_Items.AddItem("Task
1"),"Task",Convert.ToDateTime("6/21/2005"),Convert.ToDateTime("6/25/2005"),"",null);
   var_Items.AddBar(var_Items.AddItem("Task
2"),"Task",Convert.ToDateTime("6/28/2005"),Convert.ToDateTime("7/2/2005"),"",null);
axG2antt1.EndUpdate();
```

Delphi 8 (.NET only)

```
// AddLink event - Occurs when the user links two bars using the mouse.
procedure TWinForm1.AxG2antt1_AddLink(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_AddLinkEvent);
begin
   with AxG2antt1 do
   begin
      Items.RemoveLink(TObject(e.linkKey));
      Chart.UndoRemoveAction(TObject(7),TObject(1));
      Chart.UndoRemoveAction(TObject(8),TObject(1));
      OutputDebugString(Chart.UndoListAction[Nil,Nil]);
   end
end;

// ChartEndChanging event - Occurs after the chart has been changed.
procedure TWinForm1.AxG2antt1_ChartEndChanging(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_ChartEndChangingEvent);
```

```
begin
   with AxG2antt1 do
   begin
      OutputDebugString(Chart.UndoListAction[Nil,Nil]);
   end
end;

with AxG2antt1 do
begin
   BeginUpdate();
   Columns.Add('Tasks');
   with Chart do
   begin
      AllowUndoRedo := True;
      FirstVisibleDate := '6/20/2005';
      AllowLinkBars := True;
      LevelCount := 2;
      PaneWidth[False] := 48;
   end;
   with Items do
   begin
      AddBar(AddItem('Task 1'),'Task','6/21/2005','6/25/2005','',Nil);
      AddBar(AddItem('Task 2'),'Task','6/28/2005','7/2/2005','',Nil);
   end;
   EndUpdate();
end
```

Delphi (standard)

```
// AddLink event - Occurs when the user links two bars using the mouse.
procedure TForm1.G2antt1AddLink(ASender: TObject; LinkKey : WideString);
begin
   with G2antt1 do
   begin
      Items.RemoveLink(OleVariant(LinkKey));
      Chart.UndoRemoveAction(OleVariant(7),OleVariant(1));
      Chart.UndoRemoveAction(OleVariant(8),OleVariant(1));
```

```
      OutputDebugString(Chart.UndoListAction[Null,Null]);
   end
end;

// ChartEndChanging event - Occurs after the chart has been changed.
procedure TForm1.G2antt1ChartEndChanging(ASender: TObject; Operation :
BarOperationEnum);
begin
   with G2antt1 do
   begin
      OutputDebugString(Chart.UndoListAction[Null,Null]);
   end
end;

with G2antt1 do
begin
   BeginUpdate();
   Columns.Add('Tasks');
   with Chart do
   begin
      AllowUndoRedo := True;
      FirstVisibleDate := '6/20/2005';
      AllowLinkBars := True;
      LevelCount := 2;
      PaneWidth[False] := 48;
   end;
   with Items do
   begin
      AddBar(AddItem('Task 1'),'Task','6/21/2005','6/25/2005','',Null);
      AddBar(AddItem('Task 2'),'Task','6/28/2005','7/2/2005','',Null);
   end;
   EndUpdate();
end
```

VFP

```
*** AddLink event - Occurs when the user links two bars using the mouse. ***
```

```
LPARAMETERS LinkKey
   with thisform.G2antt1
      .Items.RemoveLink(LinkKey)
      .Chart.UndoRemoveAction(7,1)
      .Chart.UndoRemoveAction(8,1)
      DEBUGOUT(.Chart.UndoListAction())
   endwith

*** ChartEndChanging event - Occurs after the chart has been changed. ***
LPARAMETERS Operation
   with thisform.G2antt1
      DEBUGOUT(.Chart.UndoListAction())
   endwith

with thisform.G2antt1
   .BeginUpdate
   .Columns.Add("Tasks")
   with .Chart
      .AllowUndoRedo = .T.
      .FirstVisibleDate = {^2005-6-20}
      .AllowLinkBars = .T.
      .LevelCount = 2
      .PaneWidth(0) = 48
   endwith
   with .Items
      .AddBar(.AddItem("Task 1"),"Task",{^2005-6-21},{^2005-6-25},"")
      .AddBar(.AddItem("Task 2"),"Task",{^2005-6-28},{^2005-7-2},"")
   endwith
   .EndUpdate
endwith
```

property Chart.UnitScale as UnitEnum

Retrieves or sets a value that indicates the base unit being displayed.

Type	Description
UnitEnum	A UnitEnum expression that indicates the minimum time unit being displayed in the level.

Use the UnitScale property to change the scale unit. Use the UnitWidth property to specify the width of the time unit. The UnitScale property changes the Label, Unit and the ToolTip for a level with predefined values defined by the Label and LabelToolTip properties. Use the Label property to specify predefined formats for time units. Use the Label property to assign a different label for a specified level. Use the Unit property to specify the time unit being displayed by the level. If the user changes the Label or Unit property for a level, it is possible that UnitScale property to be changed. Use the Count property to increase the number of units being displayed in the level. Use the Alignment property to align the label in the level. Use the Zoom method to zoom the chart to a specified interval of dates. Use the LevelCount property to specify the number of levels being displayed in the control's header. Use the NextDate property to get the next date. Use the AllowOverviewZoom property to specify whether the control displays the zooming scale on the overview area. Once the user selects a new time scale unit in the overview zoom area, the control fires the OverviewZoom event.

property Chart.UnitWidth as Long

Specifies the width in pixels for the minimal unit.

Type	Description
Long	A Long expression that indicates the width of the time unit, in pixels.

Use the UnitWidth property to specify the width of the time unit. Use the UnitScale property to change the scale unit. Use the PaneWidth property to specify the width of the chart area. Use the Label property to specify the label being displayed in the level. Use the Zoom method to zoom the chart to a specified interval of dates. Use the FirstVisibleDate property to specify the first visible date in the chart. Use the ScrollTo property to ensure that a specified date fits the chart's client area. Use the Alignment property to align the label in the level. Use the Count property to increase the number of units being displayed in the level. Use the UnitWidthNonworking property to specify a different width for non-working units in the base level. Use the ShowNonworkingUnits property to hide the non-working units.

property Chart.UnitWidthNonworking as Long

Specifies the width in pixels for the minimal unit.

Type	Description
Long	A long expression that specifies the width for non-working units. The expression could be positive or negative as explained bellow.

By default, the UnitWidthNonworking property is 0. Use the UnitWidthNonworking property to reduce the amount to display non-working units in the chart. The UnitWidth property specifies the width for the units in the base level. The UnitWidthNonworking property has no effect if it is 0 (by default). Use the ShowNonworkingUnits/ShowNonworkingDates property to hide the non-working units. Use the NonworkingHours property to specify the non-working hours in your chart. Use the NonworkingDays property to specify the non-working days. The FormatLevel property indicates the formula to display the HTML captions in the levels of the chart. The Width property defines the width for a specified time-unit. The UnitWidthNonworking property has no effect if the ShowNonworkingUnits and ShowNonworkingDates properties are False. Use the FormatLabel property to specify the format of the chart's level (header).

The UnitWidthNonworking property has effect only if:

- is not zero.
- the base level displays days or hours only, (UnitScale property is exDay or exHour, and the Count property of the base level is 1)
- ShowNonworkingUnits property is True (by default)
- NonworkingHours property is not zero, if the base level displays hours
- NonworkingDays property is not zero, if the base level displays days

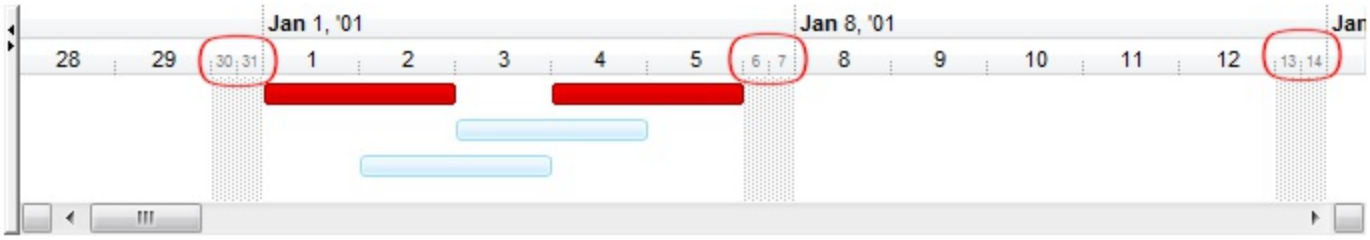
The UnitWidthNonworking property specifies the width for non-working units being displayed in the base level as follow:

1. if 0, it has no effect, so the UnitWidth property specifies the width for all units.
2. if positive, it indicates the width for non-working units.
3. if negative, it indicates that neighbor non-working units are shown as a single non-working unit with a different width (absolute value)

1. The following screen shot shows the non-working units when the UnitWidthNonworking property is 0 (by default):

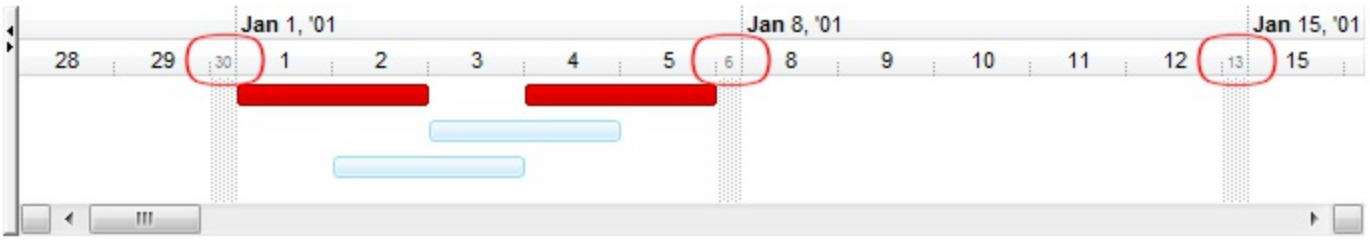
The days 30, 31, 6, 7, ... are shown using the same width, when the UnitWidthNonworking property is 0

2. The following screen shot shows the non-working units when the UnitWidthNonworking property is 12 (positive value):



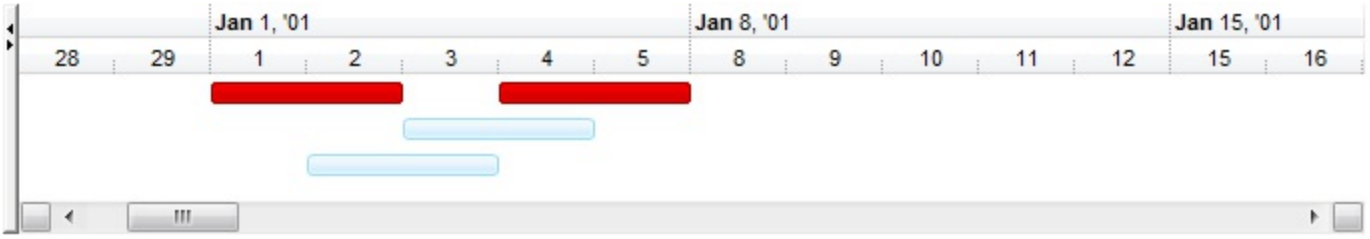
The days 30, 31, 6, 7, 13, 14, ... are shown using the a different width, when the UnitWidthNonworking property is positive.

3. The following screen shot shows the non-working units when the UnitWidthNonworking property is -12 (negative value):



The days 30, 31, 6, 7, 13, 14, ... are shown as a single non-working unit, when the UnitWidthNonworking property is negative.

4. The following screen shot hides the non-working units when the ShowNonworkingUnits property is False:



The days 30, 31, 6, 7, 13, 14, ... are not shown if the ShowNonworkingUnits property is False.

method Chart.UnselectDates ()

Unselects all dates in the chart.

Type	Description

Use the UnselectDates method to unselect all dates in the chart. Use the SelectDate property to select or unselect a new date, or to find if a specified date is selected or it is not selected. The SelectedDates property can be used to retrieve all selected dates, or to select a collection of dates. Use the SelectLevel property to specify the area being highlighted when a date is selected. The user can select dates by clicking the chart's header. You can select multiple dates keeping the CTRL key and clicking a new date. The MarkSelectDateColor property specifies the color being used to highlight the selected dates. If the MarkSelectDateColor property is identical with the BackColor property of the Chart object, the selected dates are not shown.

property Chart.UpdateOnMoving as Boolean

Specifies whether the control moves or resizes all related bars or just the bar being moved or resized.

Type	Description
Boolean	A Boolean expression that specifies whether the control moves/resizes all related bars when the current bar is moving or resizing.

By default, the UpdateOnMoving property is True. You can use this property on False, to prevent moving/updating all related bars while moving/resizing the current bar. For instance, if you are moving a summary bar, all related child bars are moved together, which could be a time consuming, if having thousand of child bars. Using the UpdateOnMoving property on False, the summary bar will be single moving, and once the user releases the mouse, all related bars takes their new positions.

property Chart.WeekDays as String

Retrieves or sets a value that indicates the list of names for each week day, separated by space.

Type	Description
String	A String expression that indicates the name of the days in the week, separated by spaces.

By default, the WeekDays property is "Sunday Monday Tuesday Wednesday Thursday Friday Saturday". The order of week days is Sunday, Monday, and so on. The FormatDate property formats a date. Use the MonthNames property to specify the name of the months in the year. Use the AMPM property to specify the name of the AM and PM indicators. Use the Label property to specify the label being displayed in the level. Use the Label property to specify the predefined format for a level based on the unit time. Use the ToolTip property to specify the tool tip being displayed when the cursor hovers the level. Use the FirstWeekDay property to specify the first day in the week.

The WeekDays property specifies the name of the days in the week for the following built-in tags:

- <%**d1**%> - First letter of the weekday (S to S).
- <%**d2**%> - First two letters of the weekday (Su to Sa).
- <%**d3**%> - First three letters of the weekday (Sun to Sat).
- <%**ddd**%> - First three letters of the weekday (Sun to Sat).
- <%**dddd**%> - Full name of the weekday (Sunday to Saturday).

The following VB sample assigns Romanian name for days in the week:

```
With G2antt1.Chart
   .WeekDays = "Duminica Luni Marti Miercuri Joi Vineri Simbata"
End With
```

The following C++ sample assigns Romanian name for days in the week:

```
m_g2antt.GetChart().SetWeekDays( "Duminica Luni Marti Miercuri Joi Vineri Simbata" );
```

The following VB.NET sample assigns Romanian name for days in the week:

```
With AxG2antt1.Chart
   .WeekDays = "Duminica Luni Marti Miercuri Joi Vineri Simbata"
End With
```

The following C# sample assigns Romanian name for days in the week:

```
axG2antt1.Chart.WeekDays = "Duminica Luni Marti Miercuri Joi Vineri Simbata";
```

The following VFP sample assigns Romanian name for days in the week:

```
With thisform.G2antt1.Chart
    .WeekDays = "Duminica Luni Marti Miercuri Joi Vineri Simbata"
EndWith
```

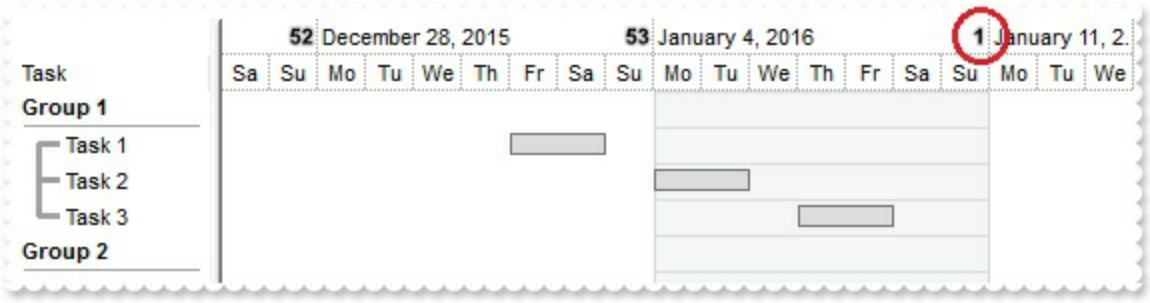
property Chart.WeekNumberAs as WeekNumberAsEnum

Specifies the way the control displays the week number.

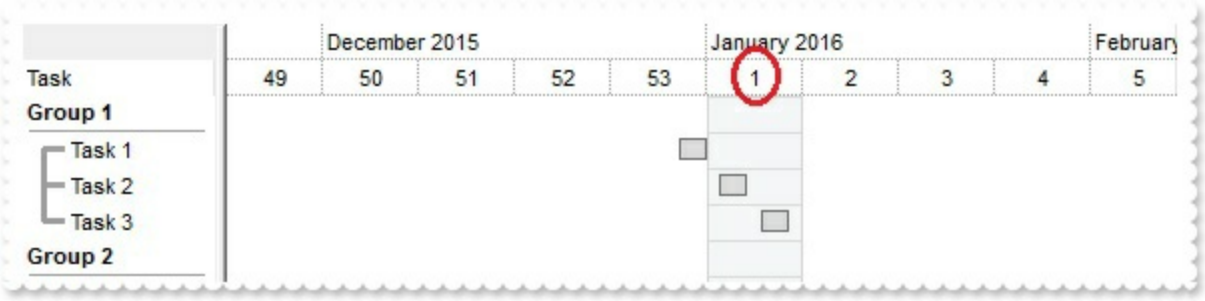
Type	Description
WeekNumberAsEnum	A WeekNumberAsEnum expression that specifies the way the control displays the week number.

By default, the WeekNumberAs property is exSimpleWeekNumber, which indicates the first week starts on January 1st of a given year, week n+1 starts 7 days after week n. The FirstWeekDay property specifies the first day of the week where the week begins. Use WeekDays property to specify the name of the days in the week. Use the MonthNames property to specify the name of the months in the year. Use the AMPM property to specify the name of the AM and PM indicators. The FormatDate property formats a date. The NextDate property computes the next date based on the time unit. Use the FirstVisibleDate property to specify the first visible date in the chart.

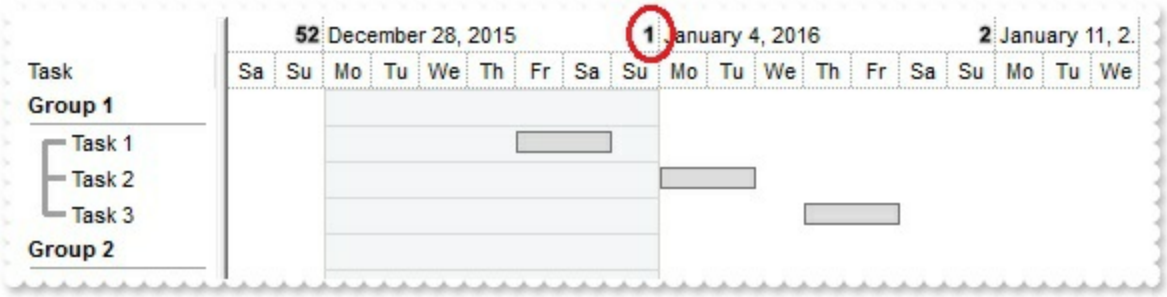
The following screen shot shows the weeks as exISO8601WeekNumber (exDay scale):



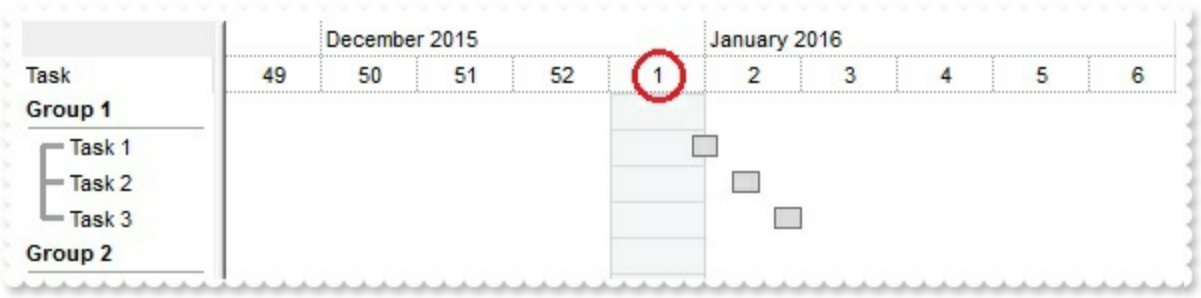
The following screen shot shows the weeks as exISO8601WeekNumber (exWeek scale):



The following screen shot shows the weeks as exSimpleWeekNumber (exDay scale):



The following screen shot shows the weeks as exSimpleWeekNumber (exWeek scale):



method Chart.Zoom (StartDate as Date, EndDate as Date, [ChangeUnitWidth as Variant])

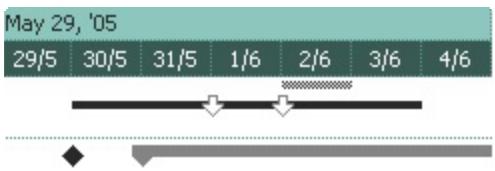
Sets or retrieves the magnification scale of the chart.

Type	Description
StartDate as Date	A Date expression that indicates the start date.
EndDate as Date	A Date expression that indicates the end date.
ChangeUnitWidth as Variant	A Boolean expression that indicates whether the Zoom method may change the UnitWidth property., If missing, the ChangeUnitWidth parameter is True.

The Zoom method zooms the chart to ensure that interval StartDate and EndDate fits the chart's area. The Zoom method may change the Label, Unit, Count and the ToolTip properties for all levels in the chart. If the ChangeUnitWidth parameter is True, the Zoom method changes the UnitWidth property as necessary. Use the LevelCount property to specify the number of levels in the chart. Use the Level property to access the level in the chart area. Use the NextDate property to compute the next date based on a given unit. Use the NonworkingDaysPattern property on hide the nonworking days. Once the user selects a new time scale unit in the overview zoom area, the control fires the OverviewZoom event.

When zooming

- the Label property takes a predefined value that's specified by the Label property of the Chart object. This way you can use the Label property of the Chart object to define the predefined formats for specified units. If the Label property for a specified unit is empty, the unit is ignored when zooming.
- the Unit property is changed accordingly with the Label property. For instance, if the Label property is set to "<%d%>", the Unit property is automatically put on exDay.
- the Count property is changed based on the available units (the Label property is not empty) and how large the interval is.
- the ToolTip property is set on a predefined value that's specified by the LabelToolTip property, accordingly with the Unit property
- If the ChangeUnitWidth parameter is True, the UnitWidth property is changed if required. For instance, if we need to display a single week, that means that the PaneWidth property is divided in 7 pieces, and so the UnitWidth property is the PaneWidth / 7.



The following VB sample zooms the chart to display one week:

```
With G2antt1.Chart
   .Label(exThirdMonth) = ""
   .Label(exDay) = "<%d%>/<%m%>"
   .Zoom .FirstVisibleDate, .NextDate(.FirstVisibleDate, exWeek), True
End With
```

The following C++ sample zooms the chart to display one week:

```
CChart chart = m_g2antt.GetChart();
chart.SetLabel(17 /*exThirdMonth*/, "" );
chart.SetLabel(4096 /*exDay*/, "<%d%>/<%m%>" );
chart.Zoom( V2D( &chart.GetFirstVisibleDate() ), chart.GetNextDate( V2D(
&chart.GetFirstVisibleDate() ), 256, COleVariant( (long)1 ) ), COleVariant( (long)TRUE ) );
```

The following VB.NET sample zooms the chart to display one week:

```
With AxG2antt1.Chart
   .Label(EXG2ANTTLib.UnitEnum.exThirdMonth) = ""
   .Label(EXG2ANTTLib.UnitEnum.exDay) = "<%d%>/<%m%>"
   .Zoom(.FirstVisibleDate, .NextDate(.FirstVisibleDate, EXG2ANTTLib.UnitEnum.exWeek),
True)
End With
```

The following C# sample zooms the chart to display one week:

```
EXG2ANTTLib.Chart chart = axG2antt1.Chart;
chart.set_Label(EXG2ANTTLib.UnitEnum.exThirdMonth, "");
chart.set_Label(EXG2ANTTLib.UnitEnum.exDay, "<%d%>/<%m%>");
chart.Zoom(Convert.ToDateTime( chart.FirstVisibleDate ),
chart.get_NextDate(Convert.ToDateTime(chart.FirstVisibleDate),
EXG2ANTTLib.UnitEnum.exWeek, 1), true);
```

The following VFP sample zooms the chart to display one week:

```
With thisform.G2antt1.Chart
   .Label(17) = "" && exThirdMonth
   .Label(4096) = "<%d%>/<%m%>" && exDay
```

```
    .Zoom(.FirstVisibleDate, .NextDate(.FirstVisibleDate, 256), .t.) && exWeek
EndWith
```

property Chart.ZoomOnFlyCaption as String

Specifies the caption to be shown in the zoom-on-fly panel, when the cursor hovers a bar.

Type	Description
String	A String expression that may contain HTML tags and expressions that indicates the additional information about the bar from the cursor to be shown in the Zoom-OnFly view.

By default, the ZoomOnFlyCaption property is "*<c><%=%C0%>
Start:<%=%1%>
End:<%=%2%>
Duration:<%=((1:=int(0:= (%513))) != 0 ? (=:1 + ' day(s)') : '') + (=:1 ? ' ' : '') + ((1:=int(0:=((=:0 - =:1 + 1/24/60/60/2)*24))) != 0 ? =:1 + ' hour(s)' : '') + (=:1 ? ' ' : '') + ((1:=round((=:0 - =:1)*60)) != 0 ? =:1 + ' min(s)' : '')%>
Working:<%=((1:=int(0:= (%258))) != 0 ? (=:1 + ' day(s)') : '') + (=:1 ? ' ' : '') + ((1:=int(0:=((=:0 - =:1 + 1/24/60/60/2)*24))) != 0 ? =:1 + ' hour(s)' : '') + (=:1 ? ' ' : '') + ((1:=round((=:0 - =:1)*60)) != 0 ? =:1 + ' min(s)' : '')%>*", which specifies that the ZoomOnFly panel displays the start, end, duration and working time of the item. Use the ZoomOnFlyCaption property to customize the additional information or caption to be shown on the bottom part of the Zoom-OnFly view, like shown in the picture bellow. The AllowZoomOnFly property specifies whether the user can display the Zoom-OnFly panel, when the cursor hovers the chart part of the area, and the user presses the CTRL + SHIFT keys combination.

The following screen shot shows the default caption in the Zoom-OnFly view:

By default, the caption of the Zoom-on-fly includes the following information:

- the cell's caption on the first column (*<%=%C0%>*)
- the starting point of the bar from the bar (*<%=%1%>*)
- the ending point of the bar from the bar (*<%=%2%>*)
- the duration or length of the bar as being the difference between start and ending point of the bar (*<%=round(%2-%1) + ' days'%>*)
- the working units as days (*<%=%258%>*)

Let's say that we have the Chart.ZoomOnFlyCaption on "*
<c><%=%C0*

*+ ' / <fgcolor=00FF00>' + %3%></fgcolor>
<solidline><upline> Start:<%=%1%>
End:<%=%2%>
Duration: <%=round(%2-%1) + ' days'%>
Working:<%='' + int(%258) + ' days' + (0:=(%258 - int(%258)) ? (' <fgcolor=FF0000>' + round(24 * =:0) + ' hours') : '') %>*", so it includes the working time in hours, not only on days.

so the caption of the Zoom-on-fly includes the following information:

- the cell's caption on the first column / the caption of the bar from the point (*<%=%C0 + ' / <fgcolor=00FF00>' + %3%>*)
- the starting point of the bar from the bar (*<%=%1%>*)
- the ending point of the bar from the bar (*<%=%2%>*)
- the duration or length of the bar as being the difference between start and ending point of the bar (*<%=round(%2-%1) + ' days'%>*)
- the working units as days and hours. (*<%='' + int(%258) + ' days' + (0:= (%258 - int(%258)) ? (' <fgcolor=FF0000>' + round(24 * =:0) + ' hours') : '') %>*)

and the zoom on fly panel shows as following:

The ZoomOnFlyCaption property supports the following built-in HTML tags:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.

The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using <a ;exp=> or <a ;e64=> anchor tags. The exp/e64 field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption.

- exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
- e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).

- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{subscript}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the

height of the font. For instance the "<**out** 000000><fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor></**sha**>*" gets:

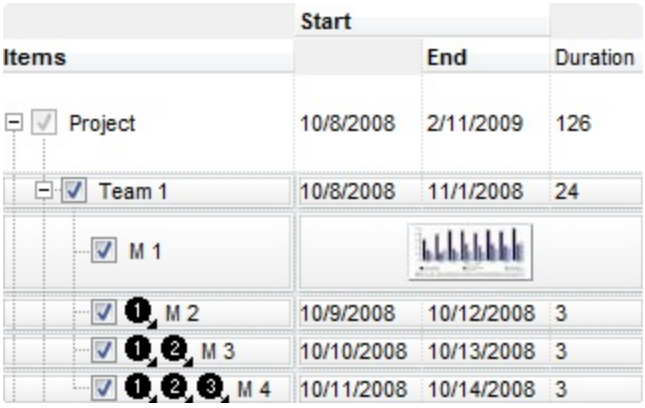
The ZoomOnFlyCaption property supports expressions between **<%=** and **%>**, aka <%=expression%>, where the expression can use predefined operators and functions like explained bellow. Inside the expression, the the **%0, %1, %2, %3**, indicates the bar's property. For instance, the <%=%0%> indicates the ItemBar(exBarName) or the type of the bar such as "Task", "Split", ... , as the exBarName identifier is 0, or <%=%9%> indicates the ItemBar(exBarKey) or the key of the bar (the Key parameter of the AddBar method) , as the exBarKey identifier is 9. Also, the known variables are **%C0, %C1, %C2, ...** which indicates the values in the columns. For instance the expression <%=%9 + '/' + %C0%>, displays the key of the bar, followed by '/' character, and next the value in the first column (0 indicates the index of the column).

This property/method supports predefined constants and operators/functions as described here.

Column object

The ExG2antt component supports multiple columns. The Columns object contains a collection of Column objects. By default, the control doesn't add any default column, so the user has to add at least one column, before inserting any new item and bars. The Column object holds information about a control's column like: Alignment, Caption, Position and so on.

The following screen shot shows the list part of the control, in other words, the part that displays the columns of the control:



The Column object supports the following properties and methods:

Name	Description
Alignment	Specifies the column's alignment.
AllowDragging	Retrieves or sets a value indicating whether the user will be able to drag the column.
AllowGroupBy	Specifies if the column can be grouped by.
AllowSizing	Retrieves or sets a value indicating whether the user will be able to change the width of the visible columns by dragging.
AllowSort	Returns or sets a value that indicates whether the user can sort the column by clicking the column's header.
AutoSearch	Specifies the kind of searching while user types characters within the columns.
AutoWidth	Computes the column's width required to fit the entire column's content.
Caption	Retrieves or sets the text displayed to the column's header.
ComputedField	Retrieves or sets a value that indicates the formula of the computed column.

CustomFilter	Retrieves or sets a value that indicates the list of custom filters.
Data	Associates an extra data to the column.
Def	Retrieves or sets a value that indicates the default value of given properties for all cells in the same column.
DefaultSortOrder	Specifies whether the default sort order is ascending or descending.
DisplayExpandButton	Shows or hides the expanding/collapsing button in the column's header.
DisplayFilterButton	Specifies whether the column's header displays the filter button.
DisplayFilterDate	Specifies whether the drop down filter window displays a date selector to specify the interval dates to filter for.
DisplayFilterPattern	Specifies whether the dropdown filterbar contains a textbox for editing the filter as pattern.
DisplaySortIcon	Retrieves or sets a value indicating whether the sort icon is visible on column's header, while the column is sorted.
Editor	Gets the column's editor object.
Enabled	Returns or sets a value that determines whether a column's header can respond to user-generated events.
ExpandColumns	Specifies the list of columns to be shown when the current column is expanded.
Expanded	Expands or collapses the column.
Filter	Specifies the column's filter when the filter type is exFilter, exPattern, exDate or exNumeric.
FilterBarDropDownWidth	Specifies the width of the drop down filter window proportionally with the width of the column.
FilterList	Specifies whether the drop down filter list includes visible or all items.
FilterOnType	Filters the column as user types characters in the drop down filter window.
FilterType	Specifies the column's filter type.
FireFormatColumn	Retrieves or sets a value that indicates whether the control fires FormatColumn to format the value of a cell hosted by column.
FormatColumn	Specifies the format to display the cells in the column.

FormatLevel	Retrieves or sets a value that indicates the layout of columns being displayed in the column's header.
GroupByFormatCell	Indicates the format of the cell to be displayed when the column gets grouped by.
GroupByTotalField	Indicates the aggregate formula to be displayed when the column gets grouped by.
HeaderAlignment	Specifies the alignment of the column's caption.
HeaderBold	Retrieves or sets a value that indicates whether the column's caption should appear in bold.
HeaderImage	Retrieves or sets a value indicating the index of an Image in the Images collection, which is displayed to the column's header.
HeaderImageAlignment	Retrieves or sets the alignment of the image in the column's header.
HeaderItalic	Retrieves or sets a value that indicates whether the column's caption should appear in italic.
HeaderStrikeOut	Retrieves or sets a value that indicates whether the column's caption should appear in strikeout.
HeaderUnderline	Retrieves or sets a value that indicates whether the column's caption should appear in underline.
HeaderVertical	Specifies whether the column's header is vertically displayed.
HTMLCaption	Retrieves or sets the text in HTML format displayed in the column's header.
Index	Returns a value that represents the index of an object in a collection.
Key	Retrieves or sets a the column's key.
LevelKey	Retrieves or sets a value that indicates the key of the column's level.
MaxWidthAutoResize	Retrieves or sets a value that indicates the maximum column's width when the WidthAutoResize is True.
MinWidthAutoResize	Retrieves or sets a value that indicates the minimum column's width when the WidthAutoResize is True.
PartialCheck	Specifies whether the column supports partial check feature.
	Retrieves or sets a value that indicates the position of the

Position	column in the header bar area.
Selected	Retrieves or sets a value that indicates whether the cell in the column is selected.
ShowFilter	Shows the column's filter window.
SortOrder	Specifies the column's sort order.
SortPosition	Returns or sets a value that indicates the position of the column in the sorting columns collection.
SortType	Returns or sets a value that indicates the way a control sorts the values for a column.
ToolTip	Specifies the column's tooltip description.
Visible	Retrieves or sets a value indicating whether the column is visible or hidden.
Width	Retrieves or sets the column's width.
WidthAutoResize	Retrieves or sets a value that indicates whether the column is automatically resized according to the width of the contents within the column.

property Column.Alignment as AlignmentEnum

Retrieves or sets the alignment of the caption into the column's header.

Type	Description
AlignmentEnum	An AlignmentEnum expression that indicates the alignment of the cells inside the column.

Use the Alignment property to change the column's alignment. Use the HeaderAlignment property to align the column's caption inside the column's header. By default, all columns are aligned to left. If the column displays the hierarchy lines, and if the Alignment property is RightAlignment the hierarchy lines are painted from right to left side. Use the HasLines property to display the control's hierarchy lines. Use the CellHAlignment property to align a particular cell. The RightToLeft property flips the order of the control's elements from right to left.

property Column.AllowDragging as Boolean

Retrieves or sets a value indicating whether the user will be able to drag the column.

Type	Description
Boolean	A boolean expression indicating whether the user will be able to drag the column.

Use the AllowDragging property to forbid user to change the column's position by dragging. If the AllowDragging is false, the column's position cannot be changed by dragging it to another position. Use the AllowSort property to avoid sorting a column when the user clicks the column's header. Use the AllowSizing property to allow user resizes a column at runtime. The HeaderEnabled property enables or disables the control's header (including the control's sort/groupby-bar). If the header is disabled, the user can't resize, sort or drag and drop any column.

property Column.AllowGroupBy as Boolean

Specifies if the column can be grouped by.

Type	Description
Boolean	A Boolean expression that specifies whether the user can drag and drop the column to be grouped by,

By default, the AllowGroupBy property is True. The AllowGroupBy property has effect only if the control's AllowGroupBy property is True. Use the AllowGroupBy property on False, to prevent a specific column to be sorted/grouped by. The same you can achieve if the AllowSort property is False. The SortBarVisible property specifies whether the control's sort bar is visible or hidden. If the control's sort bar is visible, the user can drag and drop columns to it, so the column get sorted and items grouped. The Group/Ungroup method groups or ungroup the control's list. For instance, you can remove the grouping items, by calling the Ungroup method. The GroupByFormatCell property determines the format of the cell to be displayed in the grouping item, when the column gets sorted.

property Column.AllowSizing as Boolean

Retrieves or sets a value indicating whether the user will be able to change the width of the visible columns by dragging.

Type	Description
Boolean	A boolean expression indicating whether the user will be able to change the width of the visible columns by dragging.

Use the AllowSizing property to fix the column's width. Use the ColumnAutoResize property of the control to fit the columns to the control's client area. Use the AllowSort property to avoid sorting a column when the user clicks the column's header. Use the AllowDragging property to forbid user to change the column's position by dragging. Use the Width property to specify the column's width.

property Column.AllowSort as Boolean

Returns or sets a value that indicates whether the user can sort the column by clicking the column's header.

Type	Description
Boolean	A boolean expression that indicates whether the column gets sorted when the user clicks the column's header.

Sorting by a single column in the control is a simple matter of clicking on the column head. Sorting by multiple columns, however, is not so obvious. But it's actually quite easy. First, sort by the first criterion, by clicking on the column head. Then hold the Shift key down as you click on a second heading. Another option is dragging the column's header to the control's sort bar. The SortBarVisible property shows the control's sort bar. Use the AllowSort property to avoid sorting a column when the user clicks the column's header. Use the SortOnClick property to specify the action that control executes when the user clicks the column's head. The control fires the Sort event when the control sorts a column (the user clicks the column's head) or when the sorting position is changed in the control's sort bar. Use the AllowDragging property to specify whether the column's header can be dragged. Use the DefaultSortOrder property to specify the column's default sort order, when the user first clicks the column's header. The HeaderEnabled property enables or disables the control's header (including the control's sort/groupby-bar). If the header is disabled, the user can't resize, sort or drag and drop any column.

property Column.AutoSearch as AutoSearchEnum

Specifies the kind of searching while user types characters within the columns.

Type	Description
AutoSearchEnum	An AutoSearchEnum expression that defines the type of incremental searching.

By default, the AutoSearch property is exStartWith. The AutoSearch property has effect only if the AutoSearch property of the control is True. Use the AutoSearch property to define a 'contains' incremental search. If the AutoSearch property is exContains, the control searches for items that contains the typed characters. The searching column is defined by the SearchColumnIndex property. Use the ExpandOnSearch property to expand items while user types characters in the control.

property Column.AutoWidth as Long

Computes the column's width required to fit the entire column's content.

Type	Description
Long	A long expression that indicates the width of the column to fit the entire column's content.

Use the AutoWidth property to arrange the columns to fit the entire control's content. The AutoWidth property doesn't change the column's width. Use Width property to change the column's width at runtime. Use the ColumnAutoResize property to specify whether the control resizes all visible columns to fit the control's client area.

The following VB function resizes all columns:

```
Private Sub autoSize(ByVal t As EXG2ANTTLibCtl.G2antt)
   t.BeginUpdate
      Dim c As Column
      For Each c In t.Columns
         c.Width = c.AutoWidth
      Next
      t.EndUpdate
      t.Refresh
End Sub
```

The following C++ sample resizes all visible columns:

```
#include "Columns.h"
#include "Column.h"
void autoSize( CG2antt& g2antt )
{
   g2antt.BeginUpdate();
   CColumns columns = g2antt.GetColumns();
   for ( long i = 0;i < columns.GetCount(); i++ )
   {
      CColumn column = columns.GetItem( COleVariant( i ) );
      if ( column.GetVisible() )
         column.SetWidth( column.GetAutoWidth() );
   }
   g2antt.EndUpdate();
```

```
}
```

The following VB.NET sample resizes all visible columns:

```
Private Sub autoSize(ByRef g2antt As AxEXG2ANTTLib.AxG2antt)
   g2antt.BeginUpdate()
   Dim i As Integer
   With g2antt.Columns
      For i = 0 To .Count - 1
         If .Item(i).Visible Then
            .Item(i).Width = .Item(i).AutoWidth
         End If
      Next
   End With
   g2antt.EndUpdate()
End Sub
```

The following C# sample resizes all visible columns:

```
private void autoSize( ref AxEXG2ANTTLib.AxG2antt g2antt )
{
   g2antt.BeginUpdate();
   for ( int i = 0; i < g2antt.Columns.Count - 1; i++ )
      if ( g2antt.Columns[i].Visible)
         g2antt.Columns[i].Width = g2antt.Columns[i].AutoWidth;
   g2antt.EndUpdate();
}
```

The following VFP sample resizes all visible columns:

```
with thisform.G2antt1
   .BeginUpdate()
   for i = 0 to .Columns.Count - 1
      if ( .Columns(i).Visible )
         .Columns(i).Width = .Columns(i).AutoWidth
      endif
   next
   .EndUpdate()
endwith
```

property Column.Caption as String

Retrieves or sets the text displayed to the column's header.

Type	Description
String	A string expression that indicates the column's caption.

Each property of Items object that has an argument ColIndex can use the column's caption to identify a column. Adding two columns with the same caption is accepted and these are differentiated by their indexes. Use the HTLMCaption property to display the column's caption using HTML tags. To hide a column use the Visible property of the Column object. The column's caption is displayed using the following font attributes: HeaderBold, HeaderItalic, HeaderUnderline, HeaderStrikeout. Use the Add method to add new columns and to specify their captions.

property Column.ComputedField as String

Retrieves or sets a value that indicates the formula of the computed column.

Type	Description
String	A String expression that indicates the formula to compute the field/cell. The formula is applied to all cells in the column with the CellValueFormat property on exText (the exText value is by default).

A computed field or cell displays the result of an arithmetic formula that may include operators, variables and constants. By default, the ComputedField property is empty. If the the ComputedField property is empty, the property have no effect. If the ComputedField property is not empty, all cells in the column, that have the CellValueFormat property on exText, uses the same formula to display their content. For instance, you can use the CellValueFormat property on exHTML, for cells in the column, that need to display other things than column's formula, or you can use the CellValueFormat property on exComputedField, to change the formula for a particular cell. Use the FormatColumn property to format the column.

Use the CellValueFormat property to change the type for a particular cell. Use the CellValue property to specify the cell's content. For instance, if the CellValueFormat property is exComputedField, the Caption property indicates the formula to compute the cell's content.

The Def(exCellValueFormat) property is changed to exComputedField, each time the ComputeField property is changed to a not empty value. If the ComputedField property is set to an empty string, the Def(exCellValueFormat) property is set to exText. Call the Refresh method to force refreshing the control.

The expression may be a combination of variables, constants, strings, dates and operators. A string is delimited by ", ` or ' characters, and inside they can have the starting character preceded by \ character, ie "\"This is a quote\"". A date is delimited by # character, ie #1/31/2001 10:00# means the January 31th, 2001, 10:00 AM.

The expression supports cell's identifiers as follows:

- ***%0**, **%1**, **%2**, ... specifies the value of the cell in the column with the index 0, 1 2, ... The CellValue property specifies the cell's value. For instance, "%0 format ``" formats the value on the cell with the index 0, using current regional setting, while "int(%1)" converts the value of the column with the index 1, to integer.*
- ***%C0**, **%C1**, **%C2**, ... specifies the caption of the cell, or the string the cell displays in the column with the index 0, 1 2, ... The CellCaption property specifies the cell's caption. The cell's value may be different than what the cell displays as a string. For instance, let's say a cell display HTML format. The %0 returns the html format*

including the HTML tags, while %C0 returns the cell's content as string without HTML tags. For instance, "upper(%C1)" converts the caption of the cell with the index 1, to upper case, while "%C0 left 2" returns the leftmost two characters on the cell with the index 0.

- ***%CD0**, **%CD1**, **%CD2**, ... specifies the cell's extra data in the column with the index 0, 1 2, ... The CellData property associates any extra/user data to a cell. For instance, "%CD0 = `your user data`" specifies all cells whose CellData property is `your user data`, on the column with the index 0.*
- ***%CS0**, **%CS1**, **%CS2**, ... specifies the cell's state in the column with the index 0, 1 2, ... The CellState property specifies the cell's state, and so it indicates whether the cell is checked or un-checked. For instance, "%CS0" defines all checked items on the column with the index 0, or "not %CS1" defines all un-checked items in the column with the index 1.*

This property/method supports predefined constants and operators/functions as described here.

Samples:

1. "**1**", the cell displays 1
2. "**%0 + %1**", the cell displays the sum between cells in the first and second columns.
3. "**%0 + %1 - %2**", the cell displays the sum between cells in the first and second columns minus the third column.
4. "**(%0 + %1)*0.19**", the cell displays the sum between cells in the first and second columns multiplied with 0.19.
5. "**(%0 + %1 + %2)/3**", the cell displays the arithmetic average for the first three columns.
6. "**%0 + %1 < %2 + %3**", displays 1 if the sum between cells in the first two columns is less than the sum of third and forth columns.
7. "**proper(%0)**'" formats the cells by capitalizing first letter in each word
8. "**currency(%1)**'" displays the second column as currency using the format in the control panel for money
9. "**len(%0) ? currency(dbl(%0)) : ''**" displays the currency only for not empty/blank cells.
10. "**int(date(%1)-date(%2)) + 'D ' + round(24*(date(%1)-date(%2) - floor(date(%1)-date(%2)))) + 'H'**'" displays interval between two dates in days and hours, as xD yH
11. **"2:=((1:=int(0:= date(%1)-date(%0))) = 0 ? '' : str(=:1) + ' day(s)') + (3:=round(24*(=:0-floor(=:0))) ? (len(=:2) ? ' and ' : '') + =:3 + ' hour(s)' : '')"** displays the interval between two dates, as x day(s) [and y hour(s)], where the x indictaes the number of days, and y the number of hours. The hour part is missing, if 0 hours is displayed, or nothing is displayed if dates are identical.

property Column.CustomFilter as String

Retrieves or sets a value that indicates the list of custom filters.

Type	Description
String	A String expression that defines the list of custom filters.

By default, the CustomFilter property is empty. The CustomFilter property has effect only if it is not empty, and the FilterType property is not exImage, exCheck or exNumeric. Use the DisplayFilterPattern property to hide the text box to edit the pattern, in the drop down filter window. The All predefined item and the list of custom filter is displayed in the drop down filter window, if the CustomFilter property is not empty. The Blanks and NonBlanks predefined items are not defined, when custom filter is displayed. Use the Description(exFilterBarAll) property on empty string to hide the All predefined item, in the drop down filter window. Use the DisplayFilterButton property to show the button on the column's header to drop down the filter window. Use the Background property to define the visual appearance for the drop down button.

The CustomFilter property defines the list of custom filters as pairs of (caption,pattern) where the caption is displayed in the drop down filter window, and the pattern is get selected when the user clicks the item in the drop down filter window (the FilterType property is set on exPattern, and the Filter property defines the custom pattern being selected). The caption and the pattern are separated by a "||" string (two vertical bars, character 124). The pattern expression may contains multiple patterns separated by a single "|" character (vertical bar, character 124). A pattern may contain the wild card characters '?' for any single character, '*' for zero or more occurrences of any character, '#' for any digit character. If any of the *, ?, # or | characters are preceded by a \ (escape character) it masks the character itself. If the pattern is not present in the (caption,pattern) pair, the caption is considered as being the pattern too. The pairs in the list of custom patterns are separated by "|||" string (three vertical bars, character 124). So, the syntax of the CustomFilter property should be of: CAPTION [**||** PATTERN [**|** PATTERN]] [**|||** CAPTION [**||** PATTERN [**|** PATTERN]]].

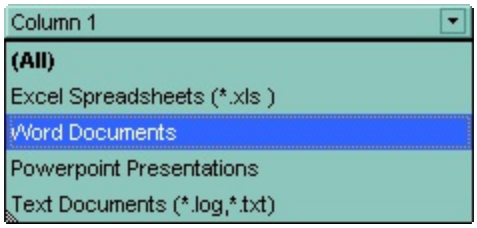
For example, you may have a list of documents and instead of listing the name of each document in the filter drop down list for the names column you may want to list the following:

Excel Spreadsheets
Word Documents
Powerpoint Presentations
Text Documents

And define the filter patterns for each line above as follows:

*.xls
*.doc
*.pps
*.txt, *.log

and so the CustomFilter property should be "**Excel Spreadsheets (*.xls)||*.xls|||Word Documents||*.doc|||Powerpoint Presentations||*.pps|||Text Documents (*.log,*.txt)||*.txt|*.log**". The following screen shot shows this custom filter format:



The FilterBarPromptVisible property specifies whether the filter prompt is visible or hidden. The filter prompt feature allows you to filter the items as you type while the filter bar is visible on the bottom part of the list area. The Filter prompt feature allows at runtime filtering data on hidden columns too.

property Column.Data as Variant

Associates an extra data to the column.

Type	Description
Variant	A Variant expression that indicates the column's extra data.

Use the Data property to assign any extra data to a column. Use the CellData property to assign an extra data to a cell. Use the ItemData property to assign an extra data to an item. Use the SortUserData or SortUserDataString type to sort the column based on the CellData value.

property Column.Def(Property as DefColumnEnum) as Variant

Retrieves or sets a value that indicates the default value of given properties for all cells in the same column.

Type	Description
Property as DefColumnEnum	A DefColumnEnum expression that indicates the property being changed.
Variant	A Variant value that specifies the newly value.

Use the Def property to specify a common value for given properties for all cells in the column. For instance, you can use the Def property to assign check boxes to all cells in the column, without enumerating them.

The following VB sample assigns checkboxes for all cells in the first column:

```
G2antt1.Columns(0).Def(exCellHasCheckBox) = True
```

The following VB sample changes the background color for all cells in the first column:

```
G2antt1.Columns(0).Def(exCellBackColor) = RGB(240, 240, 240)
```

The following C++ sample assigns checkboxes for all cells in the first column:

```
COleVariant vtCheckBox( VARIANT_TRUE );
m_g2antt.GetColumns().GetItem( COleVariant( (long) 0 ) ).SetDef( /*exCellHasCheckBox*/
0, vtCheckBox );
```

The following C++ sample changes the background color for all cells in the first column:

```
COleVariant vtBackColor( (long)RGB(240, 240, 240) );
m_g2antt.GetColumns().GetItem( COleVariant( (long) 0 ) ).SetDef( /*exCellBackColor*/ 4,
vtBackColor );
```

The following VB.NET sample assigns checkboxes for all cells in the first column:

```
With AxG2antt1.Columns(0)
   .Def(EXG2ANTTLib.DefColumnEnum.exCellHasCheckBox) = True
End With
```

The following VB.NET sample changes the background color for all cells in the first column:

```
With AxG2antt1.Columns(0)
   .Def(EXG2ANTTLib.DefColumnEnum.exCellBackColor) = ToUInt32(Color.WhiteSmoke)
End With
```

where the ToUInt32 function converts a Color expression to OLE_COLOR,

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
   Dim i As Long
   i = c.R
   i = i + 256 * c.G
   i = i + 256 * 256 * c.B
   ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample assigns checkboxes for all cells in the first column:

```
axG2antt1.Columns[0].set_Def( EXG2ANTTLib.DefColumnEnum.exCellHasCheckBox, true );
```

The following C# sample changes the background color for all cells in the first column:

```
axG2antt1.Columns[0].set_Def(EXG2ANTTLib.DefColumnEnum.exCellBackColor,
ToUInt32(Color.WhiteSmoke));
```

where the ToUInt32 function converts a Color expression to OLE_COLOR,

```
private UInt32 ToUInt32(Color c)
{
   long i;
   i = c.R;
   i = i + 256 * c.G;
   i = i + 256 * 256 * c.B;
   return Convert.ToUInt32(i);
}
```

The following VFP sample assigns checkboxes for all cells in the first column:

```
with thisform.G2antt1.Columns(0)
   .Def( 0 ) = .t.
endwith
```

The following VFP sample changes the background color for all cells in the first column:

```
with thisform.G2antt1.Columns(0)
    .Def( 4 ) = RGB(240,240,240)
endwith
```

property Column.DefaultSortOrder as Boolean

Specifies whether the default sort order is ascending or descending.

Type	Description
Boolean	A boolean expression that specifies whether the default sort order is ascending or descending. True means ascending, False means descending.

By default, the DefaultSortOrder property is False. Use the SortOnClick property to specify the operation that control should execute when the user clicks the column's header. Use the DefaultSortOrder to specify how the column is sorted at the first click on its header. Use the SortOrder property to sort a column. Use the SingleSort property to allow sorting by multiple columns.

property Column.DisplayExpandButton as Boolean

Shows or hides the expanding/collapsing button in the column's header.

Type	Description
Boolean	A Boolean expression that specifies whether the +/- expanding/collapsing button is shown in the column's header.

By default, the DisplayExpandButton property is True. The DisplayExpandButton property indicates whether the +/- expanding/collapsing button is shown in the column's header. Use the Expanded property to programmatically expand/collapse the columns. For instance, the Expanded property on False, collapse the column, while the Expanded property on True, expands the columns indicated by the ExpandColumns property. The ExpandColumns property specifies the columns to be shown/hidden when a column is expanded or collapsed.

property Column.DisplayFilterButton as Boolean

Shows or hides the column's filter bar button.

Type	Description
Boolean	A boolean expression that indicates whether the column's filter bar button is visible or hidden.

The column's filter button is displayed on the column's caption. The DisplayFilterPattern property determines whether the column's filter window includes the pattern field. *Use the FilterOnType property to enable the Filter-On-Type feature, that allows you to filter the control's data based on the characters you type.*

Use the DisplayFilterDate property to include a date selector to the column's drop down filter window. Use the FilterBarDropDownHeight to specify the height of the drop down filter window. Use the FilterBarDropDownWidth property to specify the width of the drop down filter window. Use the FilterBarHeight property to specify the height of the filter bar header. Use the FilterList property to specify the list of items being included in the column's drop down filter list. Use the FilterInclude property to specify whether the child items should be included to the list when the user applies the filter. Use the FilterCriteria property to filter items using the AND, OR and NOT operators. Use the CustomFilter property to define you custom filters. Use the ShowFilter method to show programmatically the column's drop down filter window.

The FilterBarPromptVisible property specifies whether the filter prompt is visible or hidden. The filter prompt feature allows you to filter the items as you type while the filter bar is visible on the bottom part of the list area. The Filter prompt feature allows at runtime filtering data on hidden columns too.

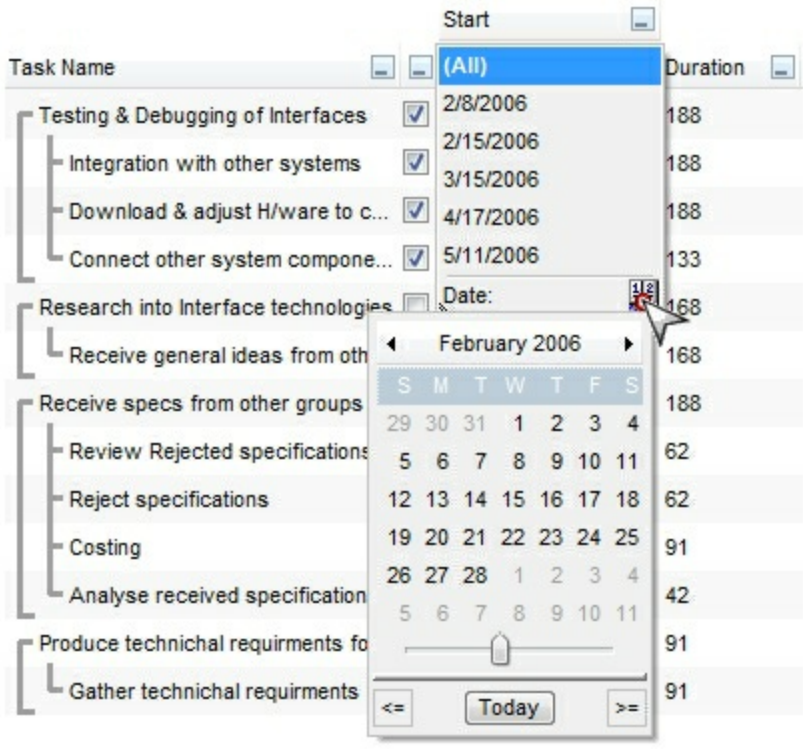
property Column.DisplayFilterDate as Boolean

Specifies whether the drop down filter window displays a date selector to specify the interval dates to filter for.

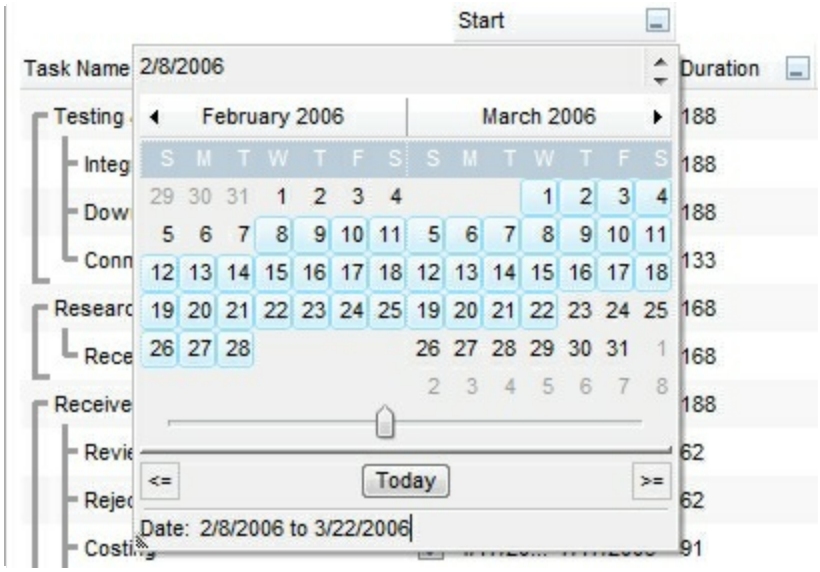
Type	Description
Boolean	A boolean expression that indicates whether the drop down filter window displays a date selector to filter items into a given interval.

By default, the DisplayFilterDate property is False. Use the DisplayFilterDate property to filter items that match a given interval of dates. Use the Description property to customize the strings being displayed on the drop down filter window. If the Date field in the filter drop down window is not empty, the FilterType property of the Column object is set on exDate, and the Filter property of the Column object points to the interval of dates being used when filtering. The FilterList property indicates the items to be include din the drop down filter's list among other filtering options.

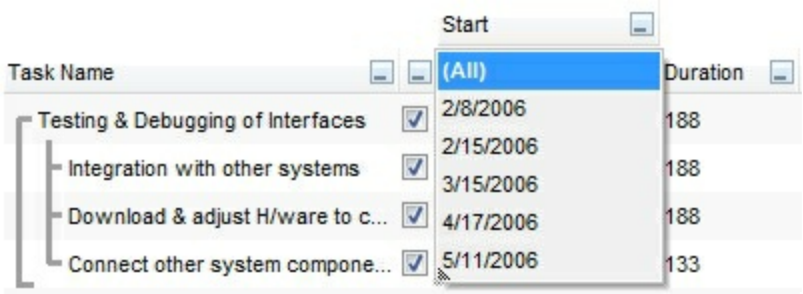
- The "Date:" filed is shown if the DisplayFilterPattern property is **True** and the DisplayFilterDate property is **True**. Use The FilterList property on exNoItems, so no items from the column are being included in the filter's list.



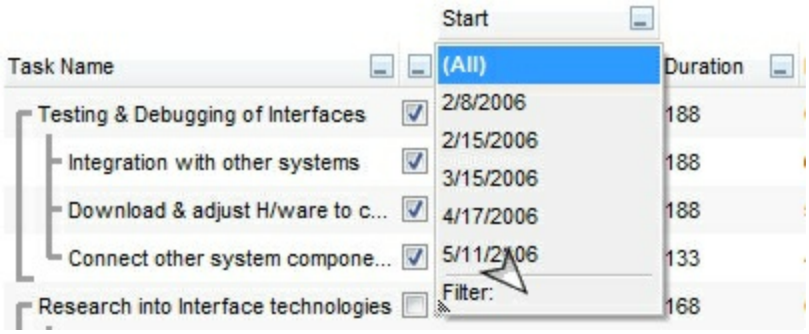
- The calendar control is shown if the DisplayFilterPattern property is **False** and the DisplayFilterDate property is **True**. Use The FilterList property on exNoItems, so no items from the column are being included in the filter's list.

- 

- No date or pattern field is shown if the DisplayFilterPattern property is **False** and the DisplayFilterDate property is **False**.

- 

- The "Filter For:" filed is shown if the DisplayFilterPattern property is **True** and the DisplayFilterDate property is **False**.

- 

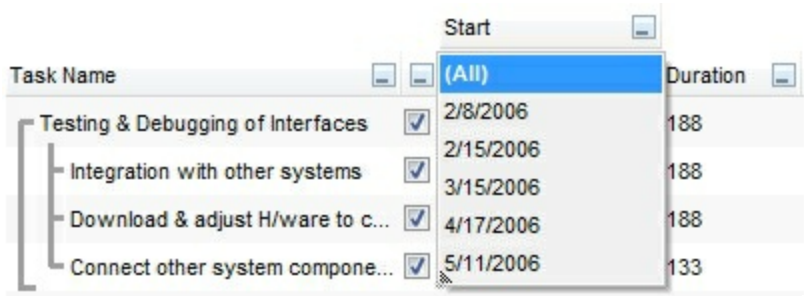
property Column.DisplayFilterPattern as Boolean

Specifies whether the dropdown filter bar contains a textbox for editing the filter as pattern.

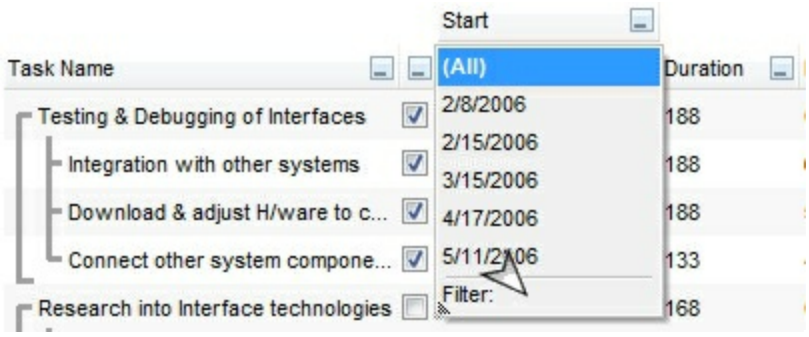
Type	Description
Boolean	A boolean expression that indicates whether the pattern field is visible or hidden.

By default, the DisplayFilterPattern property is True. The DisplayFilterPattern property specifies whether the pattern or date field is shown in the drop down filter window. The DisplayFilterDate property indicates whether the date or calendar control is shown in the drop down filter window. The FilterList property indicates the items to be include din the drop down filter's list among other filtering options.

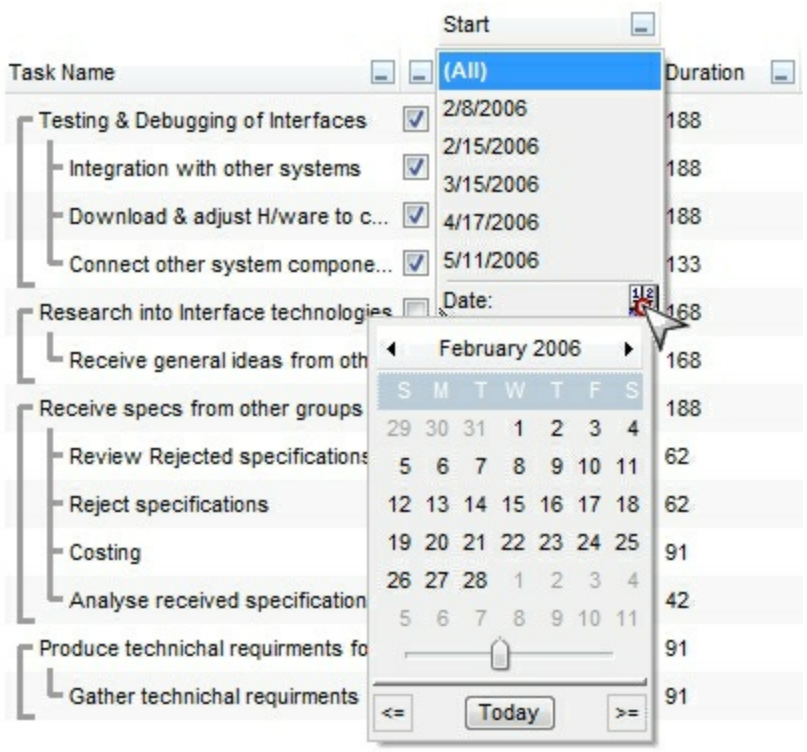
- No date or pattern field is shown if the DisplayFilterPattern property is **False** and the DisplayFilterDate property is **False**.



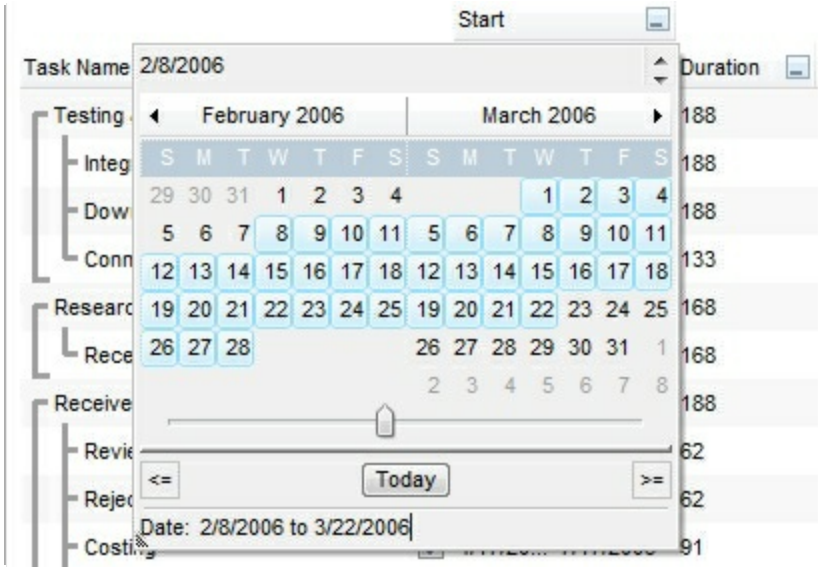
- The "Filter For:" filed is shown if the DisplayFilterPattern property is **True** and the DisplayFilterDate property is **False**.



- The "Date:" filed is shown if the DisplayFilterPattern property is **True** and the DisplayFilterDate property is **True**. Use The FilterList property on exNoItems, so no items from the column are being included in the filter's list.



- The Calendar control is shown if the DisplayFilterPattern property is **False** and the DisplayFilterDate property is **True**. Use The FilterList property on exNoItems, so no items from the column are being included in the filter's list.



Use the DisplayFilterButton property to show the column's filter button. Use the CustomFilter property to define you custom filters. The "Filter For" (pattern) field in the drop down filter window is always shown if the FilterOnType property is True, no matter of the DisplayFilterPattern property.

The drop down filter window displays the "Filter For" field if the DisplayFilterPattern property is True, and the DisplayFilterDate property is False. If the drop down filter window displays "Filter For" field, and user types the filter inside, the FilterType property of the Column is set to exPattern, and Filter property of the Column object specifies the filter being typed.

The FilterBarPromptVisible property specifies whether the filter prompt is visible or hidden. The filter prompt feature allows you to filter the items as you type while the filter bar is visible on the bottom part of the list area. The Filter prompt feature allows at runtime filtering data on hidden columns too.

property Column.DisplaySortIcon as Boolean

Retrieves or sets a value indicating whether the sort icon is visible on column's header, while the column is sorted.

Type	Description
Boolean	A boolean expression indicating whether the sort icon is visible on column's header, while the column is sorted.

Use the DisplaySortIcon property to hide the sort icon. Use the SortChildren property of the Items object to sort a column. Use the SortOrder property to sort a column. Use the SingleSort property to allow multiple sort columns.

property Column.Editor as Editor

Gets the column's editor object.

Type	Description
Editor	An Editor object that is associated to the column.

Use the Editor object to assign the same type of editor to all cells in the column. The Editor objects holds information about editing cells in the column. Use the EditType property to specify the column's edit type. Use the CellEditor property to assign a particular editor to a cell. Use the CellEditorVisible property to hide the cell's editor. Use the CellValue property to assign a value to a cell.

The following VB sample assigns a date editor to the first column:

```
With G2antt1.Columns(0).Editor
   .EditType = DateType
End With
```

The following C++ sample assigns a date editor to the first column:

```
#include "Column.h"
#include "Columns.h"
CColumn column = m_g2antt.GetColumns().GetItem( COleVariant( long(0) ) );
CEditor editor = column.GetEditor();
editor.SetEditType( 7/*DateType*/ );
```

The following VB.NET sample assigns a date editor to the first column:

```
With AxG2antt1.Columns(0).Editor
   .EditType = EXG2ANTTLib.EditTypeEnum.DateType
End With
```

The following C# sample assigns a date editor to the first column:

```
EXG2ANTTLib.Editor editor = axG2antt1.Columns[0].Editor;
editor.EditType = EXG2ANTTLib.EditTypeEnum.DateType;
```

The following VFP sample assigns a date editor to the first column:

```
with thisform.G2antt1.Columns.Item(0).Editor
```

```
    .EditType = 7 && DateType
endwith
```

property Column.Enabled as Boolean

Returns or sets a value that determines whether a column's header can respond to user-generated events.

Type	Description
Boolean	A boolean expression that determines whether a column's header can respond to user-generated events.

If the Enabled property is False, then all cells in the column are disabled, no matter if the CellEnabled property is True. Use the Enabled property to disable the control. The HeaderEnabled property enables or disables the control's header (including the control's sort/groupby-bar). If the header is disabled, the user can't resize, sort or drag and drop any column.

property Column.ExpandColumns as String

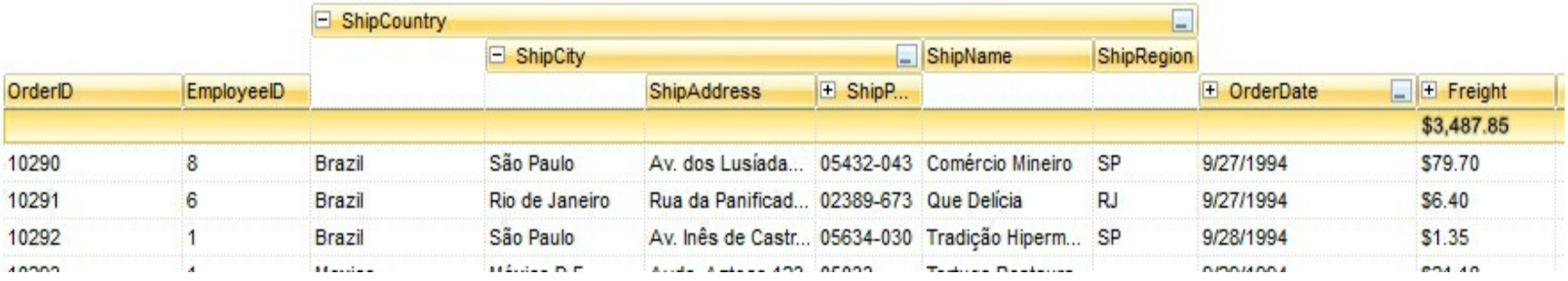
Specifies the list of columns to be shown when the current column is expanded.

Type	Description
String	A String expression that specifies the columns to be expanded/collapsed by current column. The expression contains the index of the columns to be shown or hidden, separated by comma. The list can includes the index of the current column, and so the column is always visible no matter if the column is expanded or collapsed.

By default, the ExpandColumns property is "". The ExpandColumns property specifies the columns to be shown/hidden when a column is expanded or collapsed. The ExpandColumns property can include the index of the current column, which indicates that the column is visible no matter if the column is expanded or collapsed. In other words, the Expanded/ExpandColumns properties provides expandable header. The Index property specifies the index of the column. The Expanded property specifies whether a column is expanded or collapsed. The DisplayExpandButton property indicates whether the +/- expanding/collapsing button is shown in the column's header. The HasButtons property specifies how the +/- buttons are shown.

The following screen shot shows the control's header when all columns are collapsed:

The following screen shot shows the control's header with columns expanded/collapsed :



property Column.Expanded as Boolean

Expands or collapses the column.

Type	Description
Boolean	A Boolean expression that specifies whether the column is expanded / collapsed.

By default, the Expanded property is True. Use the Expanded property to programmatically expand/collapse the columns. For instance, the Expanded property on False, collapse the column, while the Expanded property on True, expands the columns indicated by the ExpandColumns property. The ExpandColumns property specifies the columns to be shown/hidden when a column is expanded or collapsed. The DisplayExpandButton property indicates whether the +/- expanding/collapsing button is shown in the column's header.

property Column.Filter as String

Specifies the column's filter when the filter type is exFilter, exPattern, exDate, exNumeric, exCheck or exImage.

Type	Description
String	A string expression that specifies the column's filter.

- If the FilterType property is **exFilter** the Filter property indicates the list of values being included when filtering. The values are separated by '|' character. For instance if the Filter property is "CellA|CellB" the control includes only the items that have captions like: "CellA" or "CellB".

- If the FilterType is **exPattern** the Filter property defines the list of patterns used in filtering. The list of patterns is separated by the '|' character. A pattern filter may contain the wild card characters like '?' for any single character, '*' for zero or more occurrences of any character, '#' for any digit character. The '|' character separates the options in the pattern. For instance: '1*|2*' specifies all items that start with '1' or '2'.

- If the FilterType property is **exDate**, the Filter property should be of "*[dateFrom] to [dateTo]*" format, and it indicates that only items between a specified range of dates will be included. If the dateFrom value is missing, the control includes only the items before the dateTo date, if the dateTo value is missing, the control includes the items after the dateFrom date. If both dates (dateFrom and dateTo) are present, the control includes the items between this interval of dates. For instance, the "2/13/2004 to" includes all items after 2/13/2004 inclusive, or "2/13/2004 to Feb 14 2005" includes all items between 2/13/2004 and 2/14/2004.

- If the FilterType property is **exNumeric**, the Filter property may include operators like <, <=, =, <>, >= or > and numbers to define rules to include numbers in the control's list. The Filter property should be of the following format "*operator number [operator number ...]*". For instance, the "> 10" indicates all numbers greater than 10. The "<>10 <> 20" filter indicates all numbers except 10 and 20. The "> 10 < 100" filter indicates all numbers greater than 10 and less than 100. The ">= 10 <= 100 <> 50" filter includes all numbers from 10 to 100 excepts 50. The "10" filter includes only 10 in the list. The "=10 =20" includes no items in the list because after control filters only 10 items, the second rule specifies only 20, and so we have no items. The Filter property may include unlimited rules. A rule is composed by an operator and a number. The rules are separated by space characters. The CustomFilter property has no effect of the FilterType property is exNumeric.

- If the FilterType property is **exCheck** the Filter property may include "0" for unchecked

items, and "1" for checked items. The CellState property specifies the state of the cell's checkbox. If the Filter property is empty, the filter is not applied to the column, when ApplyFilter method is called. The CustomFilter property has no effect of the FilterType property is exCheck.

- If the FilterType property is **exImage** the Filter property indicates the list of icons (index of the icon being displayed) being filtered. The values are separated by '|' character. The CellImage property indicates the index of the icon being displayed in the cell. For instance, the '1|2' indicates that the filter includes the cells that display first or the second icon (with the index 1 or 2). The drop down filter window displays the (All) item and the list of icons being displayed in the column. The CustomFilter property has no effect of the FilterType property is exImage.

The Filter property has no effect if the FilterType property is one of the followings: **exAll, exBlanks** and **exNonBlanks**

The ApplyFilter method should be called to update the control's content after changing the Filter or FilterType property. The ClearFilter method clears the Filter and the FilterType properties. Use the FilterCriteria property to filter items using the AND, OR and NOT operators. Use the CustomFilter property to define you custom filters. Use the ShowFilter method to show programmatically the column's drop down filter window.

The FilterBarPromptVisible property specifies whether the filter prompt is visible or hidden. The filter prompt feature allows you to filter the items as you type while the filter bar is visible on the bottom part of the list area. The Filter prompt feature allows at runtime filtering data on hidden columns too.

property Column.FilterBarDropDownWidth as Double

Specifies the width of the drop down filter window proportionally with the width of the column.

Type	Description
Double	A double expression that indicates the width of the drop down filter window proportionally with the width of the column. If the FilterBarDropDownWidth expression is negative, the absolute value indicates the width of the drop down filter window in pixels. Else, the value indicates how many times the width of the column is multiply to get the width of the drop down filter window.

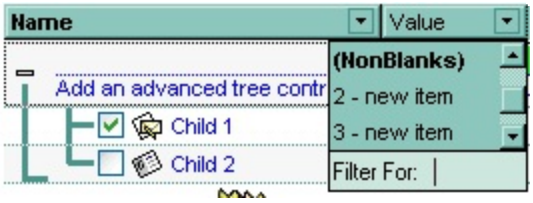
By default, the FilterBarDropDownWidth property is 1, and so, the width of the drop down filter window coincides with the width of the column. Use the Width property to specify the width of the column. Use FilterBarDropDownHeight property to specify the height of the drop down filter window. Use the FilterBarHeight property to specify the height of the control's filter bar. Use the DisplayFilterButton property to display a filter button to the column's caption. Use the Description property to define predefined strings in the filter bar. Use the FilterInclude property to specify whether the child items should be included to the list when the user applies the filter. Use the ShowFilter method to show programmatically the column's drop down filter window.

The following VB sample specifies that the width of the drop down filter window is double of the column's width:

```
With G2antt1.Columns(0)
   .FilterBarDropDownWidth = 2
End With
```

The following VB sample specifies that the width of the drop down filter window is 150 pixels:

```
With G2antt1.Columns(0)
   .FilterBarDropDownWidth = -150
End With
```



property Column.FilterList as FilterListEnum

Specifies whether the drop down filter list includes visible or all items.

Type	Description
FilterListEnum	A FilterListEnum expression that indicates the items being included in the drop down filter list.

By default, the FilterList property is exAllItems. Use the FilterList property to specify the items being included in the column's drop down filter list. Use the DisplayFilterButton property to display the column's filter bar button. The DisplayFilterDate property specifies whether the drop down filter window displays a date selector to specify the interval dates to filter for. Use the FilterCriteria property to filter items using the AND, OR and NOT operators. Use the exSortItemsAsc flag to sort ascending the values in the drop down filter list. For instance, the **exAllItems OR exSortItemsAsc** specifies that the drop down filter window lists all items in ascending order. Add the exIncludeInnerCells flag if you require adding the inner cells value to the drop down filter window.

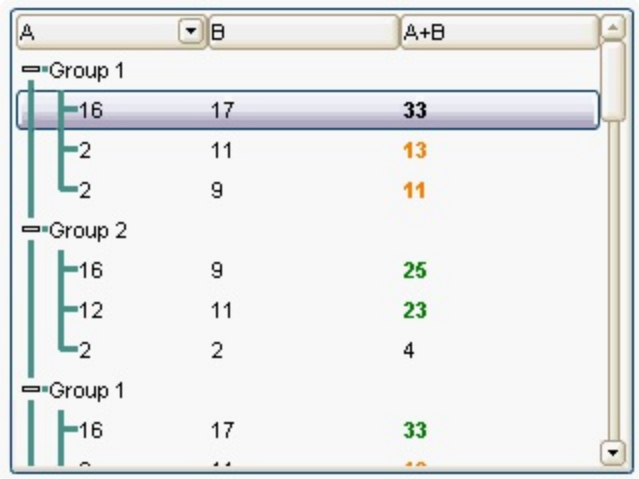
property Column.FilterOnType as Boolean

Filters the column as user types characters in the drop down filter window.

Type	Description
Boolean	A Boolean expression that specifies whether the column gets filtered as the user types characters in the drop down filter window.

By default, the FilterOnType property is False. The Filter-On-Type feature allows you to filter the control's data based on the typed characters. Use the DisplayFilterButton property to add a drop down filter button to the column's header. The Filter-On-Type feature works like follows: User clicks the column's drop down filter button, so the drop down filter window is shown. Use starts type characters, and the control filters the column based on the typed characters as it includes all items that starts with typed characters, if the AutoSearch property is exStartWith, or include in the filter list only the items that contains the typed characters, if the AutoSearch property is exContains. Click the X button on the filterbar, and so the control removes the filter, and so all data is displayed. Once, the FilterOnType property is set on True, the column's FilterType property is changed to exPattern, and the the Filter property indicates the typed string. Use the FilterCriteria property to specify the expression being used to filter the control's data when multiple columns are implied in the filter. Use the Description property to customize the text being displayed in the drop down filter window. Use the FilterHeight property to specify the height of the control's filterbar that's displayed on the bottom side of the control, once a filter is applied. The "Filter For" (pattern) field in the drop down filter window is always shown if the FilterOnType property is True, no matter of the DisplayFilterPattern property. The control fires the FilterChanging event just before applying the filter, and FilterChange once the list gets filtered.

The following screen shot shows how the data gets filtered when the user types characters in the Filter-On-Type columns:



Steps:

- The user clicks the drop down filter window, in the column A
- The "Filter For:" field is shown, and it waits for the user to start type characters.
- As user types characters, the column gets filtered the items.

property Column.FilterType as FilterTypeEnum

Specifies the column's filter type.

Type	Description
FilterTypeEnum	A FilterTypeEnum expression that indicates the filter's type.

The FilterType property defines the filter's type. By default, the FilterType is exAll. No filter is applied if the FilterType is exAll. The Filter property defines the column's filter. Use the DisplayFilterButton property to display the column's filter button. Use the FilterCriteria property to filter items using the AND, OR and NOT operators.

The ApplyFilter method should be called to update the control's content after changing the Filter or FilterType property. The ClearFilter method clears the Filter and the FilterType properties.

The FilterBarPromptVisible property specifies whether the filter prompt is visible or hidden. The filter prompt feature allows you to filter the items as you type while the filter bar is visible on the bottom part of the list area. The Filter prompt feature allows at runtime filtering data on hidden columns too.

property Column.FireFormatColumn as Boolean

Retrieves or sets a value that indicates whether the control fires FormatColumn to format the caption of a cell hosted by column.

Type	Description
Boolean	A boolean expression that indicates whether the control fires the FireFormatColumn event for the cells in the column.

By default, the FireFormatColumn property is False. The FormatColumn event is fired only if the FireFormatColumn property of the Column object is True. The FormatColumn event lets the user to provide the cell's caption before it is displayed on the control's list. For instance, the FormatColumn event is useful when the column cells contains prices (numbers), and you want to display that column formatted as currency, like $50 instead 50. Also, it is useful to use the FormatColumn event when displaying computed cells. *Newer versions of the component provides the FormatColumn property that helps formatting a cell using the several predefined functions without using the control's event FormatColumn.*

The CellValue property of the cell is being shown as:

- formatted using the FormatCell property, if it is valid
- formatted using the FormatColumn property, if it is valid

In other words, all cells applies the format of the FormatColumn property, excepts the cells with the FormatCell property being set. If the cell belongs to a column with the FireFormatColumn property on True, the Value parameter of the FormatColumn event shows the newly caption for the cell to be shown.

The FormatColumn event is fired before displaying a cell, so you can handle the FormatColumn to display anything on the cell at runtime. This way you can display the row position, you can display the value using the currency format, and so on. The FireFormatColumn property allows the control to fire the FormatColumn event for the column. The Position property specifies the position of the column.

- If your chart does *not* display a tree or a hierarchy this property is ok to be used with FormatColumn event to display the position

The following VB sample handles the FormatColumn event to display the row position:

```
Private Sub G2antt1_FormatColumn(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal
ColIndex As Long, Value As Variant)
    Value = G2antt1.Items.ItemPosition(Item)
```

```
End Sub
```

- If your chart displays a tree or a hierarchy the position of the item must be determined relative to the FirstVisibleItem as shown in the following VB sample:

```
Private Sub G2antt1_FormatColumn(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal
ColIndex As Long, Value As Variant)
   Value = G2antt1.ScrollPos(True) + RelPos(Item)
End Sub

Private Function RelPos(ByVal hVisible As Long) As Long
   With G2antt1.Items
      Dim h As Long, i As Long, n As Long
      i = 0
      n = .VisibleCount + 1
      h = .FirstVisibleItem
      While (i <= n) And h <> 0 And h <> hVisible
         i = i + 1
         h = .NextVisibleItem(h)
      Wend
      RelPos = i
   End With
End Function
```

property Column.FormatColumn as String

Specifies the format to display the cells in the column.

Type	Description
String	A string expression that defines the format to display the cell, including HTML formatting, if the cell supports it.

By default, the FormatColumn property is empty. The cells in the column use the provided format only if is valid (not empty, and syntactically correct), to display data in the column. The FormatColumn property provides a format to display all cells in the column using a predefined format. The expression may be a combination of variables, constants, strings, dates and operators, and value. The *value* operator gives the value to be formatted. A string is delimited by ", ` or ' characters, and inside they can have the starting character preceded by \ character, ie "\"This is a quote\"". A date is delimited by # character, ie #1/31/2001 10:00# means the January 31th, 2001, 10:00 AM. The cell's HTML format is applied only if the CellValueFormat or Def(exCellValueFormat) is exHTML. If valid, the FormatColumn is applied to all cells for which the CellCaptionFormat property is not exComputedField. This way you can specify which cells use or not the FormatColumn property. The FormatColumn and FormatCell properties support auto-numbering functions like explained bellow The ComputedField property indicates the formula of the computed column.

The CellValue property of the cell is being shown as:

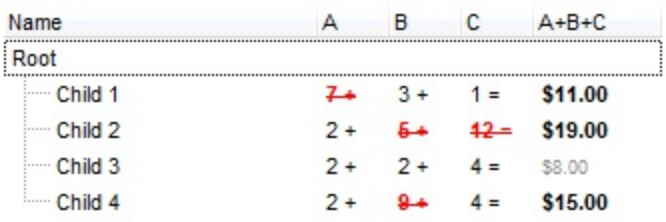
- formatted using the FormatCell property, if it is valid
- formatted using the FormatColumn property, if it is valid

In other words, all cells applies the format of the FormatColumn property, excepts the cells with the FormatCell property being set. If the cell belongs to a column with the FireFormatColumn property on True, the Value parameter of the FormatColumn event shows the newly caption for the cell to be shown.

For instance:

- the "*currency(value)*" displays the column using the current format for the currency ie, 1000 gets displayed as $1,000.00
- the "*longdate(date(value))*" converts the value to a date and gets the long format to display the date in the column, ie #1/1/2001# displays instead Monday, January 01, 2001
- the "*'' + ((0:=proper(value)) left 1) + '' + (=:0 mid 2)*" converts the name to proper, so the first letter is capitalized, bolds the first character, and let unchanged the rest, ie a "mihai filimon" gets displayed "**M**ihai Filimon".
- the "*len(value) ? ((0:=dbl(value)) < 10 ? '<fgcolor=808080><font ;7>' : '') +*

currency(=:0)" displays the cells that contains not empty daya, the value in currency format, with a different font and color for values less than 10, and bolded for those that are greater than 10, as can see in the following screen shot in the column (A+B+C):



The **value** keyword in the FormatColumn property indicates the value to be formatted.

The expression supports cell's identifiers as follows:

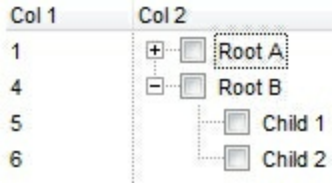
- ***%0**, **%1**, **%2**, ... specifies the value of the cell in the column with the index 0, 1 2, ... The CellValue property specifies the cell's value. For instance, "%0 format ``" formats the value on the cell with the index 0, using current regional setting, while "int(%1)" converts the value of the column with the index 1, to integer.*
- ***%C0**, **%C1**, **%C2**, ... specifies the caption of the cell, or the string the cell displays in the column with the index 0, 1 2, ... The CellCaption property specifies the cell's caption. The cell's value may be different than what the cell displays as a string. For instance, let's say a cell display HTML format. The %0 returns the html format including the HTML tags, while %C0 returns the cell's content as string without HTML tags. For instance, "upper(%C1)" converts the caption of the cell with the index 1, to upper case, while "%C0 left 2" returns the leftmost two characters on the cell with the index 0.*
- ***%CD0**, **%CD1**, **%CD2**, ... specifies the cell's extra data in the column with the index 0, 1 2, ... The CellData property associates any extra/user data to a cell. For instance, "%CD0 = `your user data`" specifies all cells whose CellData property is `your user data`, on the column with the index 0.*
- ***%CS0**, **%CS1**, **%CS2**, ... specifies the cell's state in the column with the index 0, 1 2, ... The CellState property specifies the cell's state, and so it indicates whether the cell is checked or un-checked. For instance, "%CS0" defines all checked items on the column with the index 0, or "not %CS1" defines all un-checked items in the column with the index 1.*

The expression predefined operators for auto-numbering are:

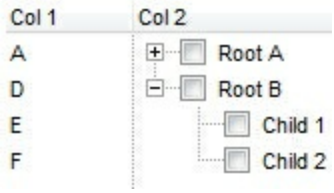
- number **index** 'format', indicates the index of the item. The first added item has the index 0, the second added item has the index 1, and so on. The index of the item remains the same even if the order of the items is changed by sorting. For instance, 1 index '' gets the index of the item starting from 1 while 100 index '' gets the index of the item starting from 100. The number indicates the starting index, while the format is a

set of characters to be used for specifying the index. If the format is missing, the index of the item is formatted as numbers. For instance: 1 index 'A-Z' gets the index as A, B, C... Z, BA, BB, ... BZ, CA, ... The 1 index 'abc' gives the index as: a,b,c,ba,bb,bc,ca,cb,cc,.... You can use other number formatting function to format the returned value. For instance "1 index " format '0||2|:'" gets the numbers grouped by 2 digits and separated by : character.

In the following screen shot the FormatColumn("Col 1") = "1 index ""

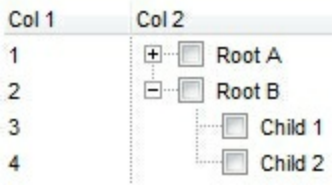


In the following screen shot the FormatColumn("Col 1") = "1 index 'A-Z'"

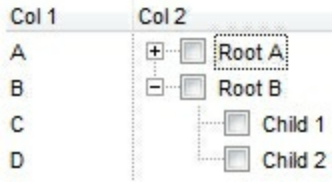


- number **apos** 'format' indicates the absolute position of the item. The first displayed item has the absolute position 0 (scrolling position on top), the next visible item is 1, and so on. The number indicates the starting position, while the format is a set of characters to be used for specifying the position. For instance, 1 apos " gets the absolute position of the item starting from 1, while 100 apos " gets the position of the item starting from 100. If the format is missing, the absolute position of the item is formatted as numbers.

  In the following screen shot the FormatColumn("Col 1") = "1 apos ""

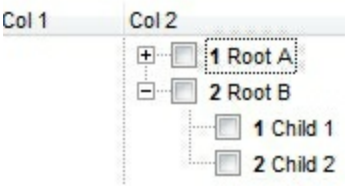


  In the following screen shot the FormatColumn("Col 1") = "1 apos 'A-Z'"

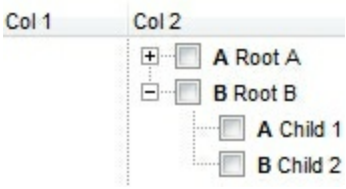


- number **pos** 'format' indicates the relative position of the item. The relative position is the position of the visible child item in the parent children collection. The number indicates the starting position, while the format is a set of characters to be used for specifying the position. For instance, 1 pos '' gets the relative position of the item starting from 1, while 100 pos '' gets the relative position of the item starting from 100. If the format is missing, the relative position of the item is formatted as numbers. *The difference between pos and opos can be seen while filtering the items in the control. For instance, if no filter is applied to the control, the pos and opos gets the same result. Instead, if the filter is applied, the opos gets the position of the item in the list of unfiltered items, while the pos gets the position of the item in the filtered list.*

  In the following screen shot the FormatColumn("Col 2") = "'' + 1 pos '' + ' ' + value"



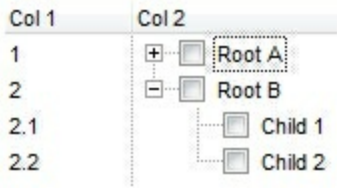
  In the following screen shot the FormatColumn("Col 2") = "'' + 1 pos 'A-Z' + ' ' + value"



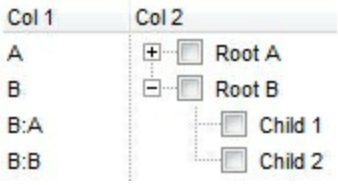
- number **opos** 'format' indicates the relative old position of the item. The relative old position is the position of the child item in the parent children collection. The number indicates the starting position, while the format is a set of characters to be used for specifying the position.For instance, 1 pos '' gets the relative position of the item starting from 1, while 100 pos '' gets the relative position of the item starting from 100. If the format is missing, the relative position of the item is formatted as numbers. *The difference between pos and opos can be seen while filtering the items in the control. For instance, if no filter is applied to the control, the pos and opos gets the same result. Instead, if the filter is applied, the opos gets the position of the item in the list of unfiltered items, while the pos gets the position of the item in the filtered list.*
- number **rpos** 'format' indicates the relative recursive position of the item. The recursive position indicates the position of the parent items too. The relative position is the position of the visible child item in the parent children collection. The number indicates the starting position, while the format is of the following type "delimiter|format|format|...". If the format is missing, the delimiter is . character, and the positions are formatted as numbers. The format is applied consecutively to each

parent item, from root to item itself.

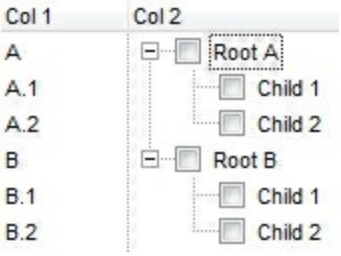
In the following screen shot the FormatColumn("Col 1") = "1 rpos ""



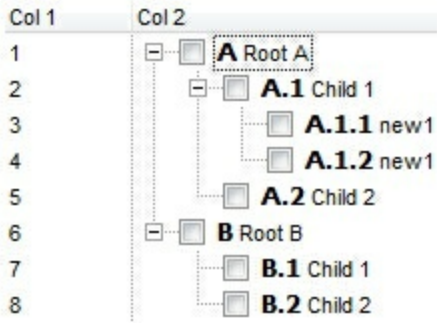
In the following screen shot the FormatColumn("Col 1") = "1 rpos ':|A-Z'"



In the following screen shot the FormatColumn("Col 1") = "1 rpos '.|A-Z|'"



In the following screen shot the FormatColumn("Col 1") = "1 apos "" and FormatColumn("Col 2") = "'' + 1 rpos '.|A-Z|' + ' ' + value"



- number **rindex** 'format', number **rapos** 'format' and number **ropos** 'format' are working similar with number **rpos** 'format', excepts that they gives the index, absolute position, or the old child position.

This property/method supports predefined constants and operators/functions as described here.

The following **VB** sample shows how can I display the column using currency:

```
With G2antt1
   .Columns.Add("Currency").FormatColumn = "currency(dbl(value))"
   With .Items
      .AddItem "1.23"
      .AddItem "2.34"
      .AddItem "0"
      .AddItem 5
      .AddItem "10000.99"
   End With
End With
```

The following **VB.NET** sample shows how can I display the column using currency:

```
With AxG2antt1
   .Columns.Add("Currency").FormatColumn = "currency(dbl(value))"
   With .Items
      .AddItem "1.23"
      .AddItem "2.34"
      .AddItem "0"
      .AddItem 5
      .AddItem "10000.99"
   End With
End With
```

The following **C++** sample shows how can I display the column using currency:

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import "C:\\Windows\\System32\\ExG2antt.dll"
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
((EXG2ANTTLib::IColumnPtr)(spG2antt1->GetColumns()->Add(L"Currency")))-
>PutFormatColumn(L"currency(dbl(value))");
EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems();
```

```
    var_Items->AddItem("1.23");
    var_Items->AddItem("2.34");
    var_Items->AddItem("0");
    var_Items->AddItem(long(5));
    var_Items->AddItem("10000.99");
```

The following **C#** sample shows how can I display the column using currency:

```
(axG2antt1.Columns.Add("Currency") as EXG2ANTTLib.Column).FormatColumn =
"currency(dbl(value))";
EXG2ANTTLib.Items var_Items = axG2antt1.Items;
    var_Items.AddItem("1.23");
    var_Items.AddItem("2.34");
    var_Items.AddItem("0");
    var_Items.AddItem(5);
    var_Items.AddItem("10000.99");
```

The following **VFP** sample shows how can I display the column using currency:

```
with thisform.G2antt1
    .Columns.Add("Currency").FormatColumn = "currency(dbl(value))"
    with .Items
        .AddItem("1.23")
        .AddItem("2.34")
        .AddItem("0")
        .AddItem(5)
        .AddItem("10000.99")
    endwith
endwith
```

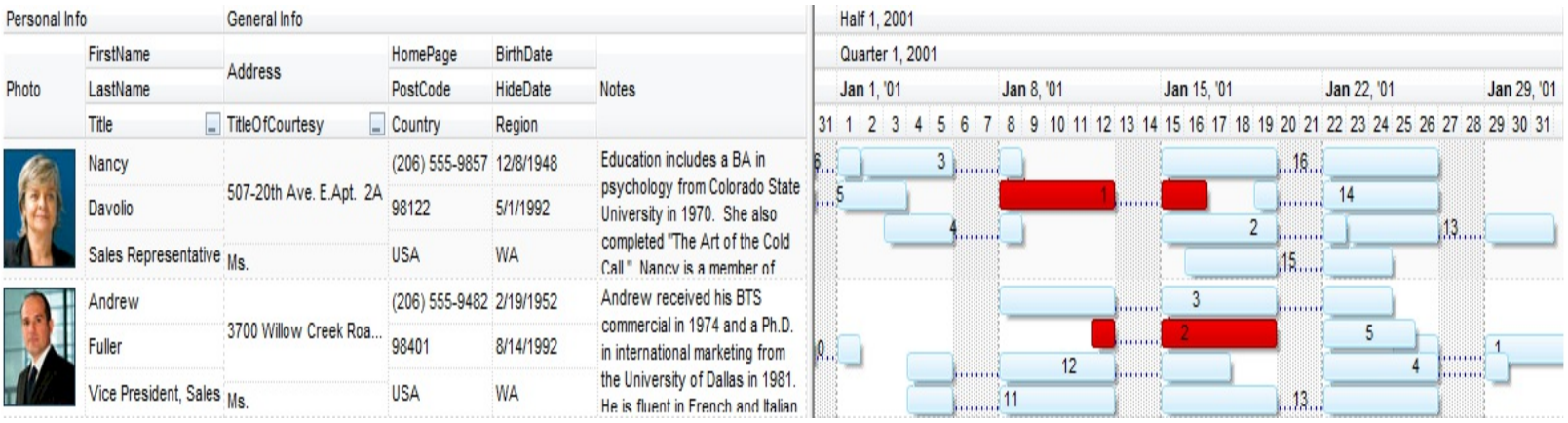
property Column.FormatLevel as String

Retrieves or sets a value that indicates the layout of columns being displayed in the column's header.

Type	Description
String	A string expression that indicates a CRD string that layouts the column's header. The Index elements in the CRD strings indicates the index of the column being displayed. The Caption elements in the CRD string support built-in HTML format.

By default, the FormatLevel property is empty. The FormatLevel property indicates the layout of the column in the control's header bar. Use the HeaderVisible property to show or hide the control's header bar. Use the HeaderHeight property to specify the height of the level in the control's header bar. Use the FormatLevel property to display multiple levels in the column's header. Use the LevelKey property to display neighbor columns on multiple levels. If the FormatLevel property is empty, the control displays the Caption or the HTMLCaption of the column. If the FormatLevel property is not empty it indicates the layout of the column being displayed. For instance, the FormatLevel = "1,2" indicates that the column's header is horizontally divided such as the left part displays the caption of the first column, and the right part displays the caption of the second column. Use the Visible property to specify whether a column is visible or hidden. Use the Add method to add new columns to the control. Use the DataSource property to bound the control to a recordset. Use the Def(exCellFormatLevel) property to specify the layout for all cells in the same column. Use the CellFormatLevel property to indicate the layout for a specific cell.

The following screen shot shows the control/list's header on 4 different levels, the same as displayed in the chart's area:



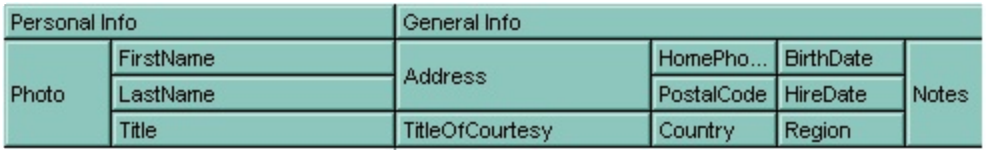
The following VB sample arranges the columns as in the above screen shot (the sample hides the columns and add instead two new columns { Personal Info, General Info }, where

the layout is displayed.

```
With G2antt1
   .BeginUpdate
   Dim c As EXG2ANTTLibCtl.Column
   For Each c In .Columns
      c.Visible = False
   Next
   With .Columns.Add("Personal Info")
      .AllowSort = False
      .AllowDragging = False
      .Width = 196
      .FormatLevel = "18;17/(14:54,(2/1/3))"
   End With
   With .Columns.Add("General Info")
      .AllowSort = False
      .AllowDragging = False
      .Width = 382
      .FormatLevel = "18;18/((7/18;4):128,((((12/10/11),(5/6/9)),15)))"
   End With
   .EndUpdate
End With
```

Before running the sample the control's header bar looks like follows:

After running the sample the control's header bar looks like follows:



The following C++ sample arranges the columns as in the above screen shot (the sample hides the columns and add instead two new columns { Personal Info, General Info }, where the layout is displayed.

```
m_g2antt.BeginUpdate();
CColumns cols = m_g2antt.GetColumns();
long nCount = cols.GetCount();
```

```
for ( long i = 0; i < nCount; i++ )
   cols.GetItem( COleVariant(i) ).SetVisible( FALSE );

CColumn col1( V_DISPATCH( &cols.Add( "Personal Info" ) ) );
col1.SetAllowSort( FALSE );
col1.SetAllowDragging( FALSE );
col1.SetWidth( 196 );
col1.SetFormatLevel( "18;18/(15:54,(2/1/4))" );
CColumn col2( V_DISPATCH( &cols.Add( "General Info" ) ) );
col2.SetAllowSort( FALSE );
col2.SetAllowDragging( FALSE );
col2.SetWidth( 512 );
col2.SetFormatLevel( "18;19/((8/18;5):128,((((13/11/12),(6/7/10)),16)))" );
m_g2antt.EndUpdate();
```

The following VB.NET sample arranges the columns as in the above screen shot (the sample hides the columns and add instead two new columns { Personal Info, General Info }, where the layout is displayed.

```
With AxG2antt1
   .BeginUpdate()
   Dim c As EXG2ANTTLib.Column
   For Each c In .Columns
      c.Visible = False
   Next
   With .Columns.Add("Personal Info")
      .AllowSort = False
      .AllowDragging = False
      .Width = 196
      .FormatLevel = "18;18/(15:54,(2/1/4))"
      .Def(EXG2ANTTLib.DefColumnEnum.exCellFormatLevel) = "15:54,(2/1/4)"
   End With
   With .Columns.Add("General Info")
      .AllowSort = False
      .AllowDragging = False
      .Width = 512
      .FormatLevel = "18;19/((8/18;5):128,((((13/11/12),(6/7/10)),16)))"
      .Def(EXG2ANTTLib.DefColumnEnum.exCellFormatLevel) = "(8/18;5):128,((((13/11/12),
```

```
(6/7/10)),16))"
   End With
   .EndUpdate()
End With
```

The following C# sample arranges the columns as in the above screen shot (the sample hides the columns and add instead two new columns { Personal Info, General Info }, where the layout is displayed.

```
axG2antt1.BeginUpdate();
foreach( EXG2ANTTLib.Column c in axG2antt1.Columns)
   c.Visible = false;
EXG2ANTTLib.Column c1 = axG2antt1.Columns.Add("Personal Info") as
EXG2ANTTLib.Column;
c1.AllowSort = false;
c1.AllowDragging = false;
c1.Width = 196;
c1.FormatLevel = "18;18/(15:54,(2/1/4))";
c1.set_Def(EXG2ANTTLib.DefColumnEnum.exCellFormatLevel,"15:54,(2/1/4)");

EXG2ANTTLib.Column c2 = axG2antt1.Columns.Add("General Info") as
EXG2ANTTLib.Column;
c2.AllowSort = false;
c2.AllowDragging = false;
c2.Width = 512;
c2.FormatLevel = "18;19/((8/18;5):128,((((13/11/12),(6/7/10)),16)))";
c2.set_Def(EXG2ANTTLib.DefColumnEnum.exCellFormatLevel,"(8/18;5):128,((((13/11/12),
(6/7/10)),16))");
axG2antt1.EndUpdate();
```

The following VFP sample arranges the columns as in the above screen shot (the sample hides the columns and add instead two new columns { Personal Info, General Info }, where the layout is displayed.

```
with thisform.G2antt1
   .BeginUpdate()
   with .Columns
      for i = 0 to .Count - 1
      .Item(i).Visible = .f.
```

```
    next
    with .Add("Personal Info")
        .AllowSort = .f.
        .AllowDragging = .f.
        .Width = 196
        .FormatLevel = "18;18/(15:54,(2/1/4))"
        .Def(32) = "15:54,(2/1/4)"
    endwith
    with .Add("General Info")
        .AllowSort = .f.
        .AllowDragging = .f.
        .Width = 512
        .FormatLevel = "18;19/((8/18;5):128,((((13/11/12),(6/7/10)),16)))"
        .Def(32) = "(8/18;5):128,((((13/11/12),(6/7/10)),16))"
    endwith
    endwith
    .EndUpdate()
endwith
```

property Column.GroupByFormatCell as String

Indicates the format of the cell to be displayed when the column gets grouped by.

Type	Description
String	A String expression that may specify HTML format, **<caption>** and **value** keywords as explained bellow.

By default, the GroupByFormatCell property is "'<caption> (' + value + ')'", which indicates that the grouping label is shown in bold, followed by the computed value of the GroupByTotalField property. The GroupByFormatCell property determines the format of the cell to be displayed in the grouping item, when the column gets sorted. The GroupByTotalField property determines the formula to be applied to the column when it gets grouped. *When the control is performing a group-by operation, the Items.FormatCell(Item,Items.GroupItem(Item)) property is set on GroupByFormatCell property, where the Item is the handle of the item being added during grouping or the Item parameter of the AddGroupItem event.*

In conclusion,

- *the **<caption>** keyword in the GroupByFormatCell property is replaced with the grouping label/value, and the result expression is passed to the FormatCell property.*
- *the **value** keyword indicates the computed value of the GroupByTotalField property.*

For instance:

- the *"'<caption> (' + currency(value) + ')'"* displays the grouping label, and the aggregate field as a currency, as specified in the regional settings.
- the *"'<caption> (' + currency(value) + `, inc. VAT ` + currency(1.19*value) + ')'"* displays the grouping label, and the aggregate field, including a computed field (VAT) as a currency, as specified in the regional settings.
- the *"'<caption> <fgcolor=808080>(Total ' + (value format ``) + ') </fgcolor>'"* displays the grouping label, and the aggregate field as a current number format, as specified in the regional settings, with a different font and foreground color.

The **value** keyword in the GroupByFormatCell property indicates the value to be formatted (as a result of the GroupByTotalField property):

The expression supports cell's identifiers as follows:

- ***%0**, **%1**, **%2**, ... specifies the value of the cell in the column with the index 0, 1 2, ... The CellValue property specifies the cell's value. For instance, "%0 format ``" formats the value on the cell with the index 0, using current regional setting, while "int(%1)"*

converts the value of the column with the index 1, to integer.

- ***%C0**, **%C1**, **%C2**, ... specifies the caption of the cell, or the string the cell displays in the column with the index 0, 1 2, ... The CellCaption property specifies the cell's caption. The cell's value may be different than what the cell displays as a string. For instance, let's say a cell display HTML format. The %0 returns the html format including the HTML tags, while %C0 returns the cell's content as string without HTML tags. For instance, "upper(%C1)" converts the caption of the cell with the index 1, to upper case, while "%C0 left 2" returns the leftmost two characters on the cell with the index 0.*
- ***%CD0**, **%CD1**, **%CD2**, ... specifies the cell's extra data in the column with the index 0, 1 2, ... The CellData property associates any extra/user data to a cell. For instance, "%CD0 = `your user data`" specifies all cells whose CellData property is `your user data`, on the column with the index 0.*
- ***%CS0**, **%CS1**, **%CS2**, ... specifies the cell's state in the column with the index 0, 1 2, ... The CellState property specifies the cell's state, and so it indicates whether the cell is checked or un-checked. For instance, "%CS0" defines all checked items on the column with the index 0, or "not %CS1" defines all un-checked items in the column with the index 1.*

This property/method supports predefined constants and operators/functions as described here.

property Column.GroupByTotalField as String

Indicates the aggregate formula to be displayed when the column gets grouped by.

Type	Description
String	A String expression that indicates the formula to be displayed on the grouping caption.

By default, the GroupByTotalField property is "count(current,rec,1)", which count recursively leaf items (implies recursively leaf items) of the grouping item. At runtime, the computed value of this formula is replaced in the HTML format being specified by the GroupByFormatCell property, for the **value** keyword. *When the control is performing a group-by operation, the Items.CellValue(Item,Items.GroupItem(Item)) property is set on GroupByTotalField property, and the Items.CellValueFormat(Item,Items.GroupItem(Item)) is exHTML + exTotalField (5), where the Item is the handle of the item being added during grouping or the Item parameter of the AddGroupItem event.* The GroupByTotalField property determines the formula to be applied to the column when it gets grouped. The GroupByFormatCell property determines the format of the cell to be displayed in the grouping item, when the column gets sorted.

For instance

- "*count(current,dir,1)*" counts the number of child items (not implies recursively child items).
- "*count(current,all,1)*" counts the number of all child items (implies recursively child items).
- "*sum(parent,dir,%1=0?0:1)*" counts the not-zero values in the second column (%1)
- "*sum(parent,dir,%1 + %2)*" indicates the sum of all cells in the second (%1) and third (%2) column that are directly descendent from the parent item.
- "*sum(all,rec,%1 + %2)*" sums all leaf cells in the second (%1) and third (%2) columns.

The syntax for the GroupByTotalField property property should be:
aggregate(**list**,**direction**,**formula**) where:

aggregate must be one of the following:

- *sum* - calculates the sum of values.
- *min* - retrieves the minimum value.
- *max* - retrieves the maximum value.
- *count* - counts the number of items.
- *avg* - calculates the average of values.

list must be one of the following:

- a *long* expression that specifies the index of the item being referred.
- a predefined string expression as follows:
  - *all* - indicates all items, so the formula is being applied to all items. The direction has no effect.
  - *current* - the current item.
  - *parent* - the parent item.
  - *root* - the root item.

direction must be one of the following:

- *dir* - collects the direct descendents.
- *rec* - collects the leaf descendents (leaf items). A leaf item is an item with no child items.
- *all* - collects all descendents.

Currently, the following items are excluded by aggregate functions:

- *not-sortable items*. The SortableItem property specifies whether the item can be sorted (a sortable item can change its position after sorting, while a not-sortable item keeps its position after sorting.
- *not-selectable items*. The SelectableItem property specifies whether the user can selects/focus the specified item.
- *divider items*. The ItemDivider property specifies whether the item displays a single cell, instead displaying whole cells.

In conclusion, aggregate functions counts ONLY items that are:

- *sortable*, SortableItem is True, by default.
- *selectable*, SelectableItem is True, by default.
- *not divider*, ItemDivider is -1, by default.

Shortly, by setting to a different value to any of these properties, makes the item to be ignored by the aggregate functions.

For instance

- count(current,dir,1) counts the number of child items (not implies recursively child items).
- count(current,all,1) counts the number of all child items (implies recursively child items).
- count(current,rec,1) counts the number of leaf items (implies recursively leaf items).
- count(current,rec,1) counts the number of leaf items (a leaf item is an item with no child items).
- sum(parent,dir,%1=0?0:1) counts the not-zero values in the second column (%1)

- sum(parent,dir,%1 + %2) indicates the sum of all cells in the second (%1) and third (%2) column that are directly descendent from the parent item.
- sum(all,rec,%1 + %2) sums all leaf cells in the second (%1) and third (%2) columns.

property Column.HeaderAlignment as AlignmentEnum

Specifies the alignment of the column's caption.

Type	Description
AlignmentEnum	An AlignmentEnum expression that specifies the alignment of the column's caption.

Use the HeaderAlignment property to align the column's caption inside the column's header. Use the Alignment property to align the cells into a column. Use the HeaderImageAlignment property to align the column's icon inside the column's header. Use the CellHAlignment property to align a cell. The RightToLeft property flips the order of the control's elements from right to left.

property Column.HeaderBold as Boolean

Retrieves or sets a value that indicates whether the column's caption should appear in bold.

Type	Description
Boolean	A boolean expression that indicates whether the column's caption should appear in bold.

The HeaderBold property specifies whether the column's caption should appear in bold. Use the CellBold or ItemBold properties to specify whether the cell or item should appear in bold. Use the HTMLCaption property to specify portions of the caption using different colors, fonts. Use the HeaderItalic, HeaderUnderline or HeaderStrikeOut property to specify different font attributes when displaying the column's caption.

property Column.HeaderImage as Long

Retrieves or sets a value indicating the index of an Image in the Images collection, which is displayed to the column's header.

Type	Description
Long	A long expression that indicates the index of image in the column's header. The last 7 bits in the high significant byte of the long expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part.

Use the HeaderImage property to assign an icon to the column's header. Use the HeaderImageAlignment property to align the column's icon inside the column's header.

property Column.HeaderImageAlignment as AlignmentEnum

Retrieves or sets the alignment of the image into the column's header.

Type	Description
AlignmentEnum	An AlignmentEnum expression that indicates the alignment of the image in the column's header.

By default, the image is left aligned. Use the HeaderImageAlignment property to aligns the icon in the column's header. Use the HeaderImage property to attach an icon to the column's header. The RightToLeft property flips the order of the control's elements from right to left.

property Column.HeaderItalic as Boolean

Retrieves or sets the Italic property of the Font object that it is used to paint the column's caption.

Type	Description
Boolean	A boolean expression that indicates whether the column's caption should appear in italic.

Use the HeaderItalic property to specify whether the column's caption should appear in italic. Use the CellItalic or ItemItalic properties to specify whether the the cell or the item should appear in italic. Use the HeaderBold, HeaderUnderline or HeaderStrikeOut property to specify different font attributes when displaying the column's caption.

property Column.HeaderStrikeOut as Boolean

Retrieves or sets a value that indicates whether the column's caption should appear in strikeout.

Type	Description
Boolean	A boolean expression that indicates whether the column's caption should appear in strikeout.

Use the HeaderStrikeOut property to specify whether the column's caption should appear in strikeout. Use the CellStrikeOut or ItemStrikeOut properties to specify whether the cell or the item should appear in strikeout. Use the HeaderItalic, HeaderUnderline or HeaderBold property to specify different font attributes when displaying the column's caption.

property Column.HeaderUnderline as Boolean

Retrieves or sets a value that indicates whether the column's caption should appear in underline.

Type	Description
Boolean	A boolean expression that indicates whether the column's caption should appear in underline.

Use the HeaderUnderline property to specify whether the column's caption should appear in underline. Use the CellUnderline or ItemUnderline properties to specify whether the cell or the item should appear in underline. Use the HeaderItalic, HeaderBold or HeaderStrikeOut property to specify different font attributes when displaying the column's caption.

property Column.HeaderVertical as Boolean

Specifies whether the column's header is vertically displayed.

Type	Description
Boolean	A boolean expression that indicates whether the column's caption is vertically printed.

Use the HeaderVertical property to display vertically the column's caption. Use the HeaderAlignment property to align the caption in the column's header. Use the Caption property to assign a caption to a column. Use the HTMLCaption property to specify an HTML caption to a column. Use the HeaderImage property to assign an icon to a column.

property Column.HTMLCaption as String

Retrieves or sets the text in HTML format displayed in the column's header.

Type	Description
String	A string expression that indicates the column's caption using built-in HTML tags.

If the HTMLCaption property is empty, the Caption property is displayed in the column's header. If the HTMLCaption property is not empty, the control uses it when displaying the column's header. Use the HeaderHeight property to change the height of the control's header bar. The list of built-in HTML tags supported are here.

property Column.Index as Long

Returns a value that represents the index of an object in a collection.

Type	Description
Long	A long expression that represents the index of an object in a collection.

Use the Position property to change the column's position. The Columns collection is zero based, so the Index property starts at 0. The last added column has the Index set to Columns.Count - 1. When a column is removed from the collection, the control updates all indexes. Use the Visible property to hide a column. Use the Columns property to access column from it's index.

property Column.Key as String

Retrieves or sets the column's key.

Type	Description
String	A string expression that defines the column's key

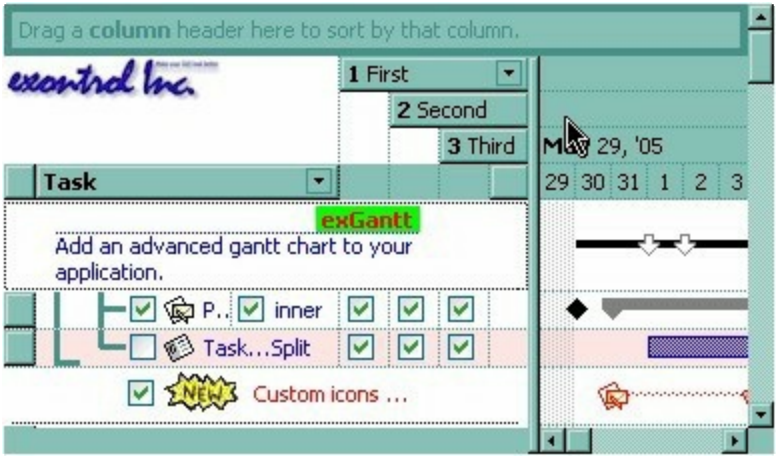
The column's key defines a column when using the Item property. Use the Index or the Key property to identify a column, when using the Columns property.

property Column.LevelKey as Variant

Retrieves or sets a value that indicates the key of the column's level.

Type	Description
Variant	A Variant expression that indicates the key of the column's level.

By default, the LevelKey is empty. The control's header displays multiple levels if there are two or more neighbor columns with the same non empty level key. The HeaderHeight property specifies the height of one level when multiple levels header is on. Use the BackColorLevelHeader property to specify the control's level header area. Use the PictureLevelHeader property to assign a picture on the control's header. The BackColorHeader property specifies the background color for column's captions. Use the LevelCount property to specify the number of levels being displayed in the chart's header.



property Column.MaxWidthAutoResize as Long

Retrieves or sets a value that indicates the maximum column's width when the WidthAutoResize is True.

Type	Description
Long	A long expression that indicates the maximum column's width when the WidthAutoResize is True.

Use the MaxWidthAutoResize property to set the maximum column's width while the WidthAutoResize property is True. If the MaxWidthAutoResize property is less than zero, there is no maximum value for the column's width. By default, the MaxWidthAutoResize property is -1. Use the ColumnAutoResize property to specify whether the control resizes the visible columns so they fit the control's client area.

property Column.MinWidthAutoResize as Long

Retrieves or sets a value that indicates the minimum column's width when the WidthAutoResize is True.

Type	Description
Long	A long expression that indicates the minimum column's width when the WidthAutoResize is True.

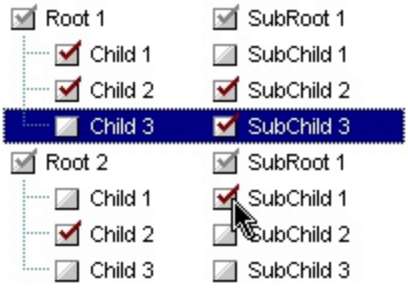
Use the MinWidthAutoResize property to set the minimum column's width while the WidthAutoResize property is True. Use the Width property to specify the column's width. Use the ColumnAutoResize property to specify whether the control resizes the visible columns so they fit the control's client area.

property Column.PartialCheck as Boolean

Specifies whether the column supports partial check feature.

Type	Description
Boolean	A boolean expression that indicates whether the control supports the partial check feature,

The PartialCheck property specifies that the column supports partial check feature. By default, the PartialCheck property is False. Use the CellHasCheckBox property to associate a check box to a cell. Use the Def property to assign a cell box for the entire column. Use the CellState property to determine the cell's state. If the PartialCheck property is True, the CellState property has three states: 0 - Unchecked, 1 - Checked and 2 - Partial Checked. Use the CheckImage property to define the icons for each state. The control supports partial check feature for any column that your control contains. Use the Add method to add new columns to the control.



property Column.Position as Long

Retrieves or sets a value that indicates the position of the column in the header bar area.

Type	Description
Long	A long expression that indicates the position of the column in the header bar area.

The column's index is not the same with the column's position. The Index property of Column cannot be changed by the user. Use the Position property to change the column's position. The EnsureVisibleColumn method ensures that a given column fits the control's client area. Use the SortPosition property to change the position of the column in the control's sort bar. Use the Visible property to hide a column. Use the Width property to specify the column's width. The RightToLeft property flips the order of the control's elements from right to left.

property Column.Selected as Boolean

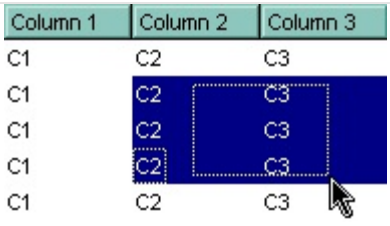
Retrieves or sets a value that indicates whether the cell in the column is selected.

Type	Description
Boolean	A boolean expression that specifies whether the cell in the column is selected.

Use the Selected property to determine the cells being selected, when FullRowSelect property is exRectSel. Use the SelectItem property to programmatically selects an item. Use the SingleSel property to allow multiple items or cells in the selection. The control fires the SelectionChanged event when user changes the selection.

The following VB sample copies the selected cells to the clipboard, if the FullRowSelect property is exRectSel:

```
Private Sub G2antt1_SelectionChanged()
  Dim strData As String
  With G2antt1
    Dim i As Long, h As HITEM
    For i = 0 To .Items.SelectCount - 1
      h = .Items.SelectedItem(i)
      Dim c As Column
      For Each c In .Columns
        If (c.Selected) Then
          strData = strData + .Items.CellCaption(h, c.Index) + vbTab
        End If
      Next
      strData = strData + vbCrLf
    Next
  End With
  Clipboard.Clear
  Clipboard.SetText strData
End Sub
```



The following C++ sample copies the selected cells to the clipboard, if the FullRowSelect property is exRectSel:

```
#include "Column.h"
#include "Columns.h"
#include "Items.h"
void OnSelectionChangedG2antt1()
{
   CString strData;
   CColumns cols = m_g2antt.GetColumns();
   CItems items = m_g2antt.GetItems();
   for ( long i = 0; i < items.GetSelectCount(); i++ )
   {
      COleVariant vtItem( items.GetSelectedItem( i ) );
      for ( long j = 0; j < cols.GetCount(); j++ )
      {
         COleVariant vtColumn( j );
         if ( cols.GetItem( vtColumn ).GetSelected() )
            strData += items.GetCellCaption(vtItem, vtColumn ) + "\t";
      }
      strData += "\r\n";
   }
   if ( OpenClipboard() )
   {
      EmptyClipboard();
      HGLOBAL hGlobal = GlobalAlloc( GMEM_MOVEABLE | GMEM_DDESHARE,
strData.GetLength() );
      CopyMemory( GlobalLock( hGlobal ), strData.operator LPCTSTR(),
strData.GetLength() );
      GlobalUnlock( hGlobal );
      SetClipboardData( CF_TEXT, hGlobal );
      CloseClipboard();
   }
}
```

The following VB.NET sample copies the selected cells to the clipboard, if the FullRowSelect property is exRectSel:

```
Private Sub AxG2antt1_SelectionChanged(ByVal sender As Object, ByVal e As
System.EventArgs) Handles AxG2antt1.SelectionChanged
   Dim strData As String = ""
   With AxG2antt1
      Dim i As Integer, h As Integer, j As Integer
      For i = 0 To .Items.SelectCount - 1
         h = .Items.SelectedItem(i)
         For j = 0 To .Columns.Count - 1
            Dim c As EXG2ANTTLib.Column = .Columns(j)
            If (c.Selected) Then
               strData = strData + .Items.CellCaption(h, c.Index) + vbTab
            End If
         Next
         strData = strData + vbCrLf
      Next
   End With
   Clipboard.Clear()
   Clipboard.SetText(strData)
End Sub
```

The following C# sample copies the selected cells to the clipboard, if the FullRowSelect property is exRectSel:

```
private void axG2antt1_SelectionChanged(object sender, System.EventArgs e)
{
   string strData = "";
   for (int i = 0; i < axG2antt1.Items.SelectCount; i++)
   {
      for ( int j = 0; j < axG2antt1.Columns.Count; j++ )
         if (axG2antt1.Columns[j].Selected)
         {
            string cellData =
axG2antt1.Items.get_CellCaption(axG2antt1.Items.get_SelectedItem(i), j);
            strData += cellData + "\t";
         }
      strData += "\r\n";
   }
```

```
    Clipboard.Clear();
    Clipboard.SetText(strData);
}
```

The following VFP sample copies the selected cells to the clipboard, if the FullRowSelect property is exRectSel (SelectionChanged event):

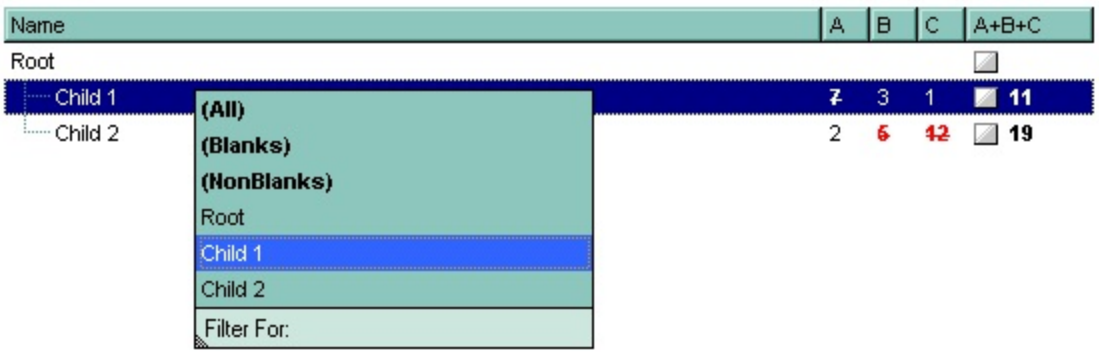
```
*** ActiveX Control Event ***

with thisform.G2antt1.Items
    local strData, i, j, cols
    strData = ""
    cols = thisform.G2antt1.Columns
    for i = 0 to .SelectCount - 1
        .DefaultItem = .SelectedItem( i )
        for j = 0 to cols.Count - 1
            if ( cols.Item(j).Selected )
                strData = strData + .CellCaption(0,j) + chr(9)
            endif
        next
        strData = strData + chr(13) + chr(10)
    next
    _CLIPTEXT = strData
endwith
```

method Column.ShowFilter ([Options as Variant])

Shows the column's filter window.

Type	Description
Options as Variant	A string expression that indicates the position (in screen coordinates) and the size (in pixels) where the drop down filter window is shown. The Options parameter is composed like follows: • the first parameter indicates the X coordinate in screen coordinate, -1 if the current cursor position is used, or empty if the coordinate is ignored • the second parameter indicates the Y coordinate in screen coordinate, -1 if the current cursor position is used, or empty if the coordinate is ignored • the third parameter indicates the width in pixels of the drop down window, or empty if the width is ignored • the forth parameter indicates the height in pixels of the drop down window, or empty if the height is ignored By default, the drop down filter window is shown at its default position bellow the column's header.

Use the ShowFilter method to show the column's drop down filter programmatically. By default, the drop down filter window is shown only if the user clicks the filter button in the column's header, if the DisplayFilterButton property is True. The drop down filter window if the user selects a predefined filter, or enters a pattern to match. If the Options parameter is missing, or all parameters inside the Options are missing, the size of the drop down filter window is automatcially computed based on the FilterBarDropDownWidth property and FilterBarDropDownHeight property. Use the ColumnFromPoint property to get the index of the column from the point.



For instance, the following VB sample displays the column's drop down filter window when

the user right clicks the control:

```
Private Sub G2antt1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
  If (Button = 2) Then
    With G2antt1.Columns
      With .Item(G2antt1.ColumnFromPoint(-1, -1))
        .ShowFilter "-1,-1,200,200"
      End With
    End With
  End If
End Sub
```

The following VB.NET sample displays the column's drop down filter window when the user right clicks the control:

```
Private Sub AxG2antt1_MouseUpEvent(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_MouseUpEvent) Handles AxG2antt1.MouseUpEvent
  If (e.button = 2) Then
    With AxG2antt1.Columns
      With .Item(AxG2antt1.get_ColumnFromPoint(-1, -1))
        .ShowFilter("-1,-1,200,200")
      End With
    End With
  End If
End Sub
```

The following C# sample displays the column's drop down filter window when the user right clicks the control:

```
private void axG2antt1_MouseUpEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_MouseUpEvent e)
{
  if (e.button == 2)
  {
    EXG2ANTTLib.Column c = axG2antt1.Columns[axG2antt1.get_ColumnFromPoint(-1,
-1)];
    c.ShowFilter("-1,-1,200,200");
  }
}
```

The following C++ sample displays the column's drop down filter window when the user right clicks the control:

```
void OnMouseUpG2antt1(short Button, short Shift, long X, long Y)
{
    m_g2antt.GetColumns().GetItem( COleVariant( m_g2antt.GetColumnFromPoint( -1, -1 ) )
).ShowFilter( COleVariant( "-1,-1,200,200" ) );
}
```

The following VFP sample displays the column's drop down filter window when the user right clicks the control:

```
*** ActiveX Control Event ***
LPARAMETERS button, shift, x, y

if ( button = 2 ) then
    With thisform.G2antt1.Columns
        With .Item(thisform.G2antt1.ColumnFromPoint(-1, -1))
            .ShowFilter("-1,-1,200,200")
        EndWith
    EndWith
endif
```

property Column.SortOrder as SortOrderEnum

Specifies the column's sort order.

Type	Description
SortOrderEnum	A SortOrderEnum expression that indicates the column's sort order.

The SortOrder property determines the column's sort order. By default, the SortOrder property is SortNone. Use the SortOrder property to sort a column at runtime. Use the SortType property to determine the way how the column is sorted. Use the AllowSort property to avoid sorting a column when the user clicks the column. Use the SingleSort property to specify whether the control supports sorting by single or multiple columns. If the control supports sorting by multiple columns, the SortOrder property adds or removes the column to the sorting columns collection. For instance, if the SortOrder property is set to SortAscending or SortDescending the column is added to the sorting columns collection. If the SortOrder property is set to SortNone the control removes the column from its sorting columns collection. The Sort event is fired when the user sorts a column. The SortPosition property changes the position of the column in the control's sort bar. Use the DefaultSortOrder property to specify the column's default sort order, when the user first clicks the column's header.

The control automatically sorts a column when the user clicks the column's header, if the SortOnClick property is exDefaultSort. If the SortOnClick property is exNoSort, the control disables sorting the items when the user clicks the column's header. There are two methods to get the items sorted like follows:

- Using the SortOrder property of the Column object::

  ```
  G2antt1.Columns(ColIndex).SortOrder = SortAscending
  ```

  The SortOrder property adds the sorting icon to the column's header, if the DisplaySortIcon property is True.

- Using the SortChildren method of the Items collection. The SortChildren sorts the items. The SortChildren method sorts the child items of the given parent item in the control. SortChildren will not recourse through the tree, only the immediate children of the item will be sorted. The following sample sort descending the list of root items on the "Column 1"(if your control displays a list, all items are considered being root items).

  ```
  G2antt1.Items.SortChildren 0, "Column 1", False
  ```

property Column.SortPosition as Long

Returns or sets a value that indicates the position of the column in the sorting columns collection.

Type	Description
Long	A long expression that indicates the position of the column in the control's sort bar. The collection is 0 - based.

Use the SortPosition to change programmatically the position of the column in the control's sort bar. Use the SingleSort property to allow sorting by multiple columns. Use the SortBarVisible property to show the control's sort bar. Use the SortOrder property to add columns to the control's sort bar. The control fires the Sort event when the user sorts a column. Use the ItemBySortPosition property to get the columns being sorted in their order. Use the AllowSort property to avoid sorting a column when the user clicks the column.

property Column.SortType as SortTypeEnum

Returns or sets a value that indicates the way a control sorts the values for a column.

Type	Description
SortTypeEnum	A SortTypeEnum expression that indicates the way a control sorts the values for a column.

The SortType property specifies how the column gets sorted. By default, the column's SortType is String. The CellValue property indicates the values being sorted. Use the SortType property to specifies how the control will sort the column. Use the SortChildren property of Items to do a sort based on a column. Use the SingleSort property to specify whether the control supports sorting by single or multiple columns. The SortOrder property determines the column's sort order. The Sort event is fired when the user sorts a column. The SortPosition property changes the position of the column in the sorting columns collection. The CellData property specifies the values being sorted, if the SortType property is SortUserData, SortUserDataString.

property Column.ToolTip as String

Specifies the column's tooltip description.

Type	Description
String	A string expression that defines the column's tooltip. The column's tooltip supports built-in HTML format

By default, the ToolTip property is "..." (three dots). Use the ToolTip property to assign a tooltip to a column. If the ToolTip property is "...", the control displays the column's caption if it doesn't fit the column's header. Use the Caption or HTMLCaption property to specify the caption of the column. The column's tooltip shows up when the cursor hovers the column's header. Use the CellToolTip property to assign a tooltip to a cell

property Column.Visible as Boolean

Retrieves or sets a value indicating whether the column is visible or hidden.

Type	Description
Boolean	A boolean expression indicating whether the column is visible or hidden.

Use the Visible property to hide a column. Use the Width property to resize the column. The ColumnAutoResize property specifies whether the visible columns fit the control's client area. Use the Position property to specify the column's position. Use the HeaderVisible property to show or hide the control's header bar. Use the ColumnFromPoint property to get the column from point. Use the Remove method to remove a column.

property Column.Width as Long

Retrieves or sets the column's width.

Type	Description
Long	A long expression that indicates the column's width in pixels.

The Width property specifies the column's width in pixels. Use the Visible property to hide a column. Use the SortBarColumnWidth property to specify the column's head in the control's sort bar. Use the ColumnAutoResize property to fit all visible columns in the control's client area. Use the FilterBarDropDownWidth property to specify the width of the drop down filter window.

The following VB sample shows how to set the width for all columns:

```
Private Sub G2antt1_AddColumn(ByVal Column As EXG2ANTTLibCtl.IColumn)
   Column.Width = 128
End Sub
```

The following VB.NET sample changes the column's width when a new column is added:

```
Private Sub AxG2antt1_AddColumn(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_AddColumnEvent) Handles AxG2antt1.AddColumn
   e.column.Width = 128
End Sub
```

The following C# sample changes the column's width when a new column is added:

```
private void axG2antt1_AddColumn(object sender,
AxEXG2ANTTLib._IG2anttEvents_AddColumnEvent e)
{
   e.column.Width = 128;
}
```

The following C++ sample changes the column's width when a new column is added:

```
#include "Column.h"
#include "Columns.h"
void OnAddColumnG2antt1(LPDISPATCH Column)
{
```

```
    CColumn column( Column );
    column.SetWidth( 128 );
}
```

The following VFP sample changes the column's width when a new column is added:

```
*** ActiveX Control Event ***
LPARAMETERS column

with column
    .Width = 128
endwith
```

property Column.WidthAutoResize as Boolean

Retrieves or sets a value that indicates whether the column is automatically resized according to the width of the contents within the column.

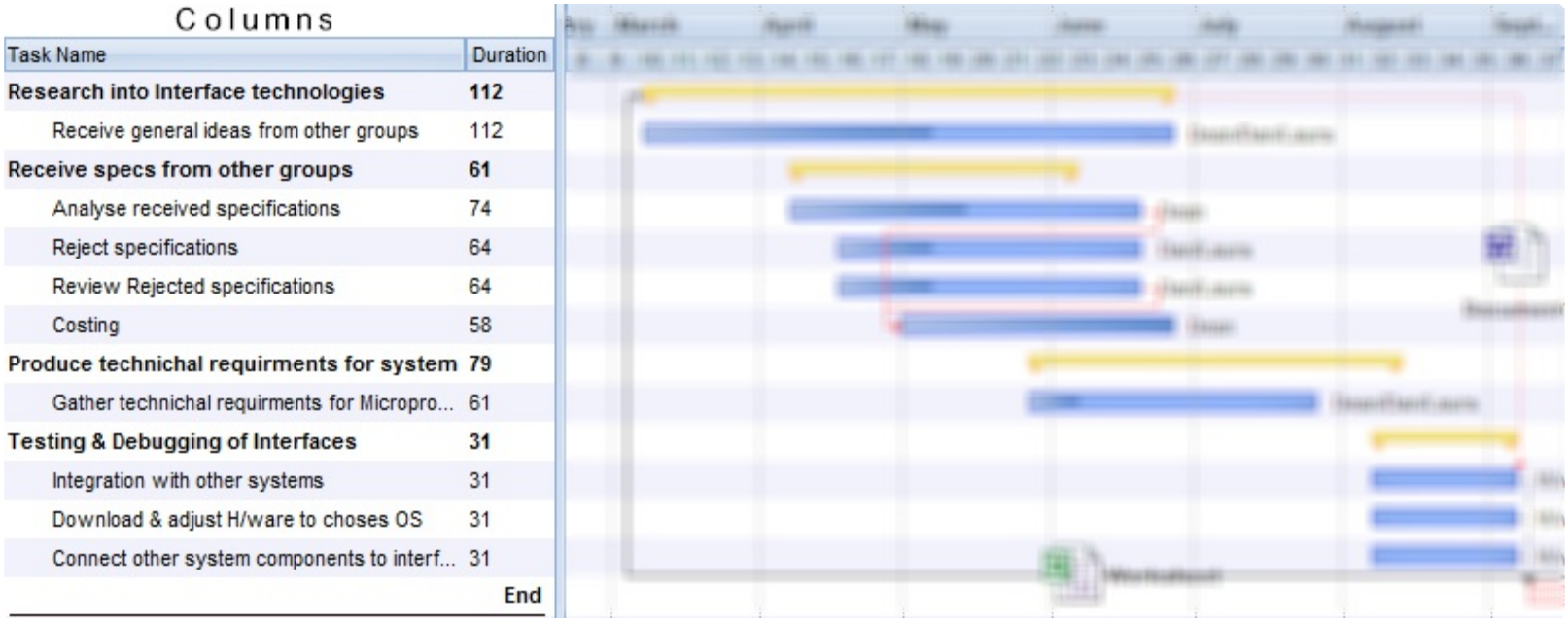
Type	Description
Boolean	A boolean expression that indicates whether the column is automatically resized according to the width of the contents within the column.

If the WidthAutoResize property is True, the column's width is resized after user expands, or collapse the items. Also, the column's width is refreshed if the user adds new items to the control. If the WidthAutoResize property is True, the column's width is not larger than MaxWidthAutoResize value, and it is not less than MinWidthAutoResize value. You can use the AutoWidth property to computes the column's width to fit its content. For instance, if you have a control with one column, and this property True, you can simulate a simple tree, because the control will automatically add a horizontal scroll bar when required. Use the ColumnAutoResize property to specify whether the control resizes the visible columns so they fit the control's client area.

Columns object

The ExG2antt control supports multiple columns. The Columns object contains a collection of Column objects. Use the Columns property of the control to access the control columns. By default, the control's columns collection is empty, so the user must add at least one column, before adding new items and bars. Each item has a cell corresponding to each column. The control's header displays the columns, while the chart's header displays the time scale units. The Column object can be accessed at the adding time, or using the Item or ItemBySortPosition property. Also, the control fires the AddColumn event is fired when a new columns has been added to Columns collection.

The following screen shot shows the list part of the control, in other words, the part that displays the columns of the control:



The Columns object supports the following method and properties:

Name	Description
Add	Adds a Column object to the collection and returns a reference to the newly created object.
Clear	Removes all objects in a collection.
Count	Returns the number of objects in a collection.
Item	Returns a specific Column of the Columns collection.
ItemBySortPosition	Returns a Column object giving its sorting position.
Remove	Removes a specific member from the Columns collection.
SortBarColumn	Returns the Column from control's SortBar giving its position.
SortBarColumnsCount	Retrieves the count of Columns, in the control's SortBar

method Columns.Add (ColumnCaption as String)

Adds a Column object to the collection and returns a reference to the newly created object.

Type	Description
ColumnCaption as String	A string expression that indicates the caption for the column being added

Return	Description
Variant	A Column object that indicates the newly added column.

By default, the control contains no columns. Before adding new items, you need to add columns. Use the Add property to add new columns to the control. Use the LoadXML/SaveXML methods to load/save the control's data from/to XML files. The control fires the AddColumn event is fired when a new columns has been added to Columns collection. Use the Caption property to change the column's caption. Use the HTLMCaption property to display the column's caption using HTML tags. To hide a column use the Visible property of the Column object. Use the AddItem, InsertItem, InsertControlItem, PutItems, DataSource properties to add new items to the control. Use the BeginUpdate and EndUpdate methods to maintain performance while adding new columns and items.

The following VB sample adds columns from a record set:

```
Set rs = CreateObject("ADODB.Recordset")
rs.Open "Orders", "Provider=Microsoft.Jet.OLEDB.3.51;Data Source= D:\Program
Files\Microsoft Visual Studio\VB98\NWIND.MDB", 3 ' Opens the table using static mode
G2antt1.BeginUpdate
' Add the columns
With G2antt1.Columns
For Each f In rs.Fields
   .Add f.Name
Next
End With
G2antt1.PutItems rs.getRows()
G2antt1.EndUpdate
```

The following VC sample adds a column:

```
#include "Columns.h"
#include "Column.h"
CColumns columns = m_g2antt.GetColumns();
```

```
CColumn column( V_DISPATCH( &columns.Add( "Column 1" ) ) );
column.SetHeaderBold( TRUE );
```

The following VB.NET sample adds a column:

```
With AxG2antt1.Columns
   With .Add("Column 1")
      .HeaderBold = True
   End With
End With
```

The Add method returns a Column object in a VARIANT value, so you can use a code like follows:

```
With AxG2antt1.Columns
   Dim c As EXG2ANTTLib.Column
   c = .Add("Column 1")
   With c
      .HeaderBold = True
   End With
End With
```

this way, you can have the properties of the column at design time when typing the '.' character.

The following C# sample adds a column:

```
EXG2ANTTLib.Column column = axG2antt1.Columns.Add( "Column 1" ) as
EXG2ANTTLib.Column;
column.HeaderBold = true;
```

The following VFP sample adds a column:

```
with thisform.G2antt1.Columns.Add( "Column 1" )
   .HeaderBold = .t.
endwith
```

method Columns.Clear ()

Removes all objects in a collection.

Type	Description

Use the Remove method when you need to remove only a column. Use the Clear method to remove all columns in the control. The Clear method removes all items, too. Use the RemoveAllItems method to remove all items in the control.

property Columns.Count as Long

Returns the number of objects in a collection.

Type	Description
Long	Counts the Column object into the collection.

The Count property counts the columns in the collection. Use the Columns property to access the control's Columns collection. Use the Item property to access a column by its index or key. Use the Add method to add new columns to the control. Use the Remove method to remove a column. Use the Clear method to clear the columns collection.

The following VB sample enumerates the columns in the control:

```
For Each c In G2antt1.Columns
   Debug.Print c.Caption
Next
```

The following VB sample enumerates the columns in the control:

```
For i = 0 To G2antt1.Columns.Count - 1
   Debug.Print G2antt1.Columns(i).Caption
Next
```

The following VC sample enumerates the columns in the control:

```
#include "Columns.h"
#include "Column.h"
CColumns columns = m_g2antt.GetColumns();
for ( long i = 0; i < columns.GetCount(); i++ )
{
   CColumn column = columns.GetItem( COleVariant( i ) );
   OutputDebugString( column.GetCaption() );
}
```

The following VB.NET sample enumerates the columns in the control:

```
With AxG2antt1.Columns
   Dim i As Integer
   For i = 0 To .Count - 1
```

```
    Debug.WriteLine(.Item(i).Caption)
  Next
End With
```

The following C# sample enumerates the columns in the control:

```
EXG2ANTTLib.Columns columns =axG2antt1.Columns;
for ( int i = 0; i < columns.Count; i++ )
{
  EXG2ANTTLib.Column column = columns[i];
  System.Diagnostics.Debug.WriteLine( column.Caption );
}
```

The following VFP sample enumerates the columns in the control:

```
with thisform.G2antt1.Columns
  for i = 0 to .Count - 1
    wait window nowait .Item(i).Caption
  next
endwith
```

property Columns.Item (Index as Variant) as Column

Returns a specific Column of the Columns collection.

Type	Description
Index as Variant	A long expression that indicates the column's index or a string expression that indicates the column's key or the column's caption.
Column	A column object being returned.

Use the Item property to access to a specific column. The Count property counts the columns in the control. Use the Columns property to access the control's Columns collection.

The Item property is the default property of the Columns object so the following statements are equivalents:

```
G2antt1.Columns.Item ("Freight")
G2antt1.Columns ("Freight")
```

The following VB sample enumerates the columns in the control:

```
For i = 0 To G2antt1.Columns.Count - 1
   Debug.Print G2antt1.Columns(i).Caption
Next
```

The following VC sample enumerates the columns in the control:

```
#include "Columns.h"
#include "Column.h"
CColumns columns = m_g2antt.GetColumns();
for ( long i = 0; i < columns.GetCount(); i++ )
{
   CColumn column = columns.GetItem( COleVariant( i ) );
   OutputDebugString( column.GetCaption() );
}
```

The following VB.NET sample enumerates the columns in the control:

```
With AxG2antt1.Columns
   Dim i As Integer
```

```
    For i = 0 To .Count - 1
        Debug.WriteLine(.Item(i).Caption)
    Next
End With
```

The following C# sample enumerates the columns in the control:

```
EXG2ANTTLib.Columns columns =axG2antt1.Columns;
for ( int i = 0; i < columns.Count; i++ )
{
    EXG2ANTTLib.Column column = columns[i];
    System.Diagnostics.Debug.WriteLine( column.Caption );
}
```

The following VFP sample enumerates the columns in the control:

```
with thisform.G2antt1.Columns
    for i = 0 to .Count - 1
        wait window nowait .Item(i).Caption
    next
endwith
```

property Columns.ItemBySortPosition (Position as Variant) as Column

Returns a Column object giving its sorting position.

Type	Description
Position as Variant	A long expression that indicates the position of column being requested.
Column	A Column object being accessed.

Use the ItemBySortPosition property to get the list of sorted columns in their order. Use the SortPosition property to specify the position of the column in the sorting columns collection. Use the SingleSort property to specify whether the control supports sorting by single or multiple columns. Use the SortOrder property to sort a column programmatically. The control fires the Sort event when the user sorts a column.

The following VB sample displays the list of columns being sorted:

```
Dim s As String, i As Long, c As Column
i = 0
With G2antt1.Columns
   Set c = .ItemBySortPosition(i)
   While (Not c Is Nothing)
      s = s + """" & c.Caption & """ " & Iif(c.SortOrder = SortAscending, "A", "D") & " "
      i = i + 1
      Set c = .ItemBySortPosition(i)
   Wend
End With
s = "Sort: " & s
Debug.Print s
```

The following VC sample displays the list of columns being sorted:

```
CString strOutput;
CColumns columns = m_g2antt.GetColumns();
long i = 0;
CColumn column = columns.GetItemBySortPosition( COleVariant( i ) );
while ( column.m_lpDispatch )
{
   strOutput += "\"" + column.GetCaption() + "\" " + ( column.GetSortOrder() == 1 ? "A" :
"D" ) + " ";
```

```
    i++;
    column = columns.GetItemBySortPosition( COleVariant( i ) );
}
OutputDebugString( strOutput );
```

The following VB.NET sample displays the list of columns being sorted:

```
With AxG2antt1
    Dim s As String, i As Integer, c As EXG2ANTTLib.Column
    i = 0
    With AxG2antt1.Columns
        c = .ItemBySortPosition(i)
        While (Not c Is Nothing)
            s = s + """" & c.Caption & """ " & IIf(c.SortOrder =
EXG2ANTTLib.SortOrderEnum.SortAscending, "A", "D") & " "
            i = i + 1
            c = .ItemBySortPosition(i)
        End While
    End With
    s = "Sort: " & s
    Debug.WriteLine(s)
End With
```

The following C# sample displays the list of columns being sorted:

```
string strOutput = "";
int i = 0;
EXG2ANTTLib.Column column = axG2antt1.Columns.get_ItemBySortPosition( i );
while ( column != null )
{
    strOutput += column.Caption + " " + ( column.SortOrder ==
EXG2ANTTLib.SortOrderEnum.SortAscending ? "A" : "D" ) + " ";
    column = axG2antt1.Columns.get_ItemBySortPosition( ++i );
}
Debug.WriteLine( strOutput );
```

The following VFP sample displays the list of columns being sorted (the code is listed in the Sort event) :

```
local s, i, c
i = 0
s = ""
With thisform.G2antt1.Columns
   c = .ItemBySortPosition(i)
   do While (!isnull(c))
      with c
         s = s + "'" + .Caption
         s = s + "' " + Iif(.SortOrder = 1, "A", "D") + " "
         i = i + 1
      endwith
      c = .ItemBySortPosition(i)
   enddo
endwith
s = "Sort: " + s
wait window nowait s
```

method Columns.Remove (Index as Variant)

Removes a specific member from the Columns collection.

Type	Description
Index as Variant	A long expression that indicates the column's index, or a string expression that indicates the column's caption or the column's key.

The Remove method removes a specific column in the Columns collection. Use Clear method to remove all Column objects. The RemoveColumn event is fired when a column is about to be removed. Use the Visible property to hide a column.

property Columns.SortBarColumn (Position as Variant) as Column

Returns the Column from control's SortBar giving its position.

Type	Description
Position as Variant	A long expression that specifies the position where the column is requested
Column	A Column object that specifies the sorted/grouped column at giving position, or empty if no column is found.

The SortBarColumn / SortBarColumnsCount properties can be used to enumerate the columns in the control's sort bar. Use the SortOrder property of the Column object on SortAscending / SortDescending to add a column to the sort bar, on SortNone to remove the column from the control's sort bar. Use the SortType property to determine the way how the column is sorted. Use the AllowSort property to avoid sorting a column when the user clicks the column. Use the SingleSort property to specify whether the control supports sorting by single or multiple columns. For instance, the SortBarColumnsCount counts the number of grouped columns, if the control's AllowGroupBy property is True.

property Columns.SortBarColumnsCount as Long

Retrieves the count of Columns, in the control's SortBar

Type	Description
Long	A long expression that specifies the number of columns being shown in the control's sort bar.

By default, the SortBarColumnsCount property is 0. The SortBarColumnsCount property counts the columns being shown in the sort bar. The SortBarColumn / SortBarColumnsCount properties can be used to enumerate the columns in the control's sort bar. Use the SortOrder property of the Column object on SortAscending / SortDescending to add a column to the sort bar, on SortNone to remove the column from the control's sort bar. Use the SortType property to determine the way how the column is sorted. Use the AllowSort property to avoid sorting a column when the user clicks the column. Use the SingleSort property to specify whether the control supports sorting by single or multiple columns. For instance, the SortBarColumnsCount counts the number of grouped columns, if the control's AllowGroupBy property is True.

ConditionalFormat object

The conditional formatting feature allows you to apply formats to a cell or range of cells, bars, and have that formatting change depending on the value of the cell or the value of a formula. Use the Add method to add new ConditionalFormat objects. Use the Item property to access a ConditionalFormat object. The ConditionalFormat object supports the following properties and method:

Name	Description
ApplyTo	Specifies whether the format is applied to items or columns.
ApplyToBars	Specifies the list of bars that the current format is applied to. The list includes the name of the bars separated by comma character.
BackColor	Retrieves or sets the background color for objects that match the condition.
BarColor	Specifies the color to be applied to bars if the conditional expression is accomplished.
BarOverviewColor	Specifies the color to be applied to bars, in the overview portion of the control, if the conditional expression is accomplished.
Bold	Bolds the objects that match the condition.
ChartBackColor	Specifies the color to be applied to item's background in the chart section of the control, if the conditional expression is accomplished.
ClearBackColor	Clears the background color.
ClearBarColor	Clears the bar's color.
ClearBarOverviewColor	Clears the bar's overview color.
ClearChartBackColor	Clears the item's background in the chart section of the control.
ClearForeColor	Clears the foreground color.
Enabled	Specifies whether the condition is enabled or disabled.
Expression	Indicates the expression being used in the conditional format.
Font	Retrieves or sets the font for objects that match the criteria.
	Retrieves or sets the foreground color for objects that

ForeColor	match the condition.
Italic	Specifies whether the objects that match the condition should appear in italic.
Key	Checks whether the expression is syntactically correct.
StrikeOut	Specifies whether the objects that match the condition should appear in strikeout.
Underline	Underlines the objects that match the condition.
Valid	Checks whether the expression is syntactically correct.
Verify	Verifies the current conditional format if it is applied to the giving item.

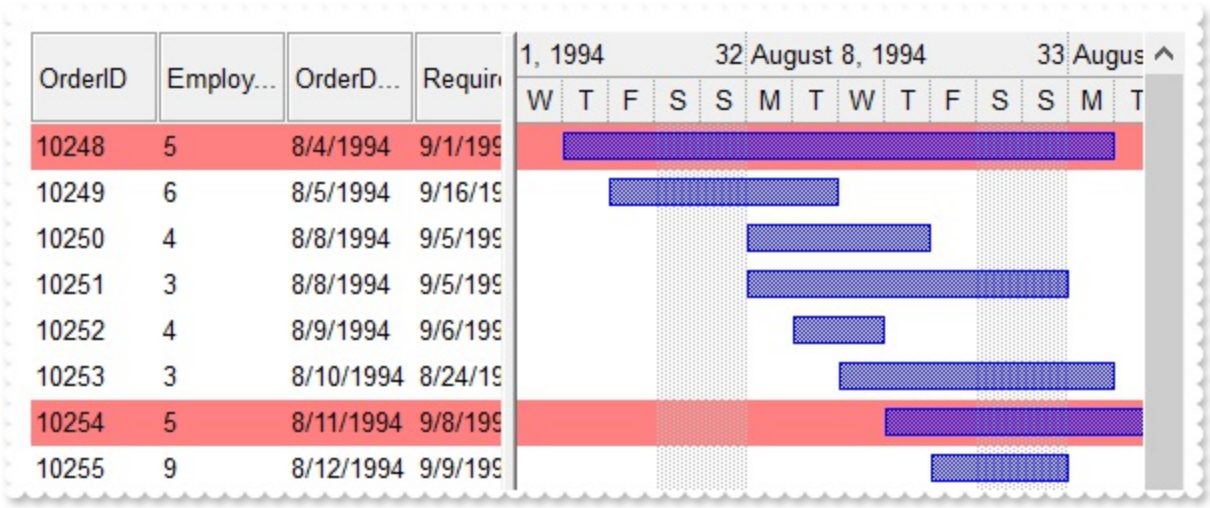
property ConditionalFormat.ApplyTo as FormatApplyToEnum

Specifies whether the format is applied to items or columns.

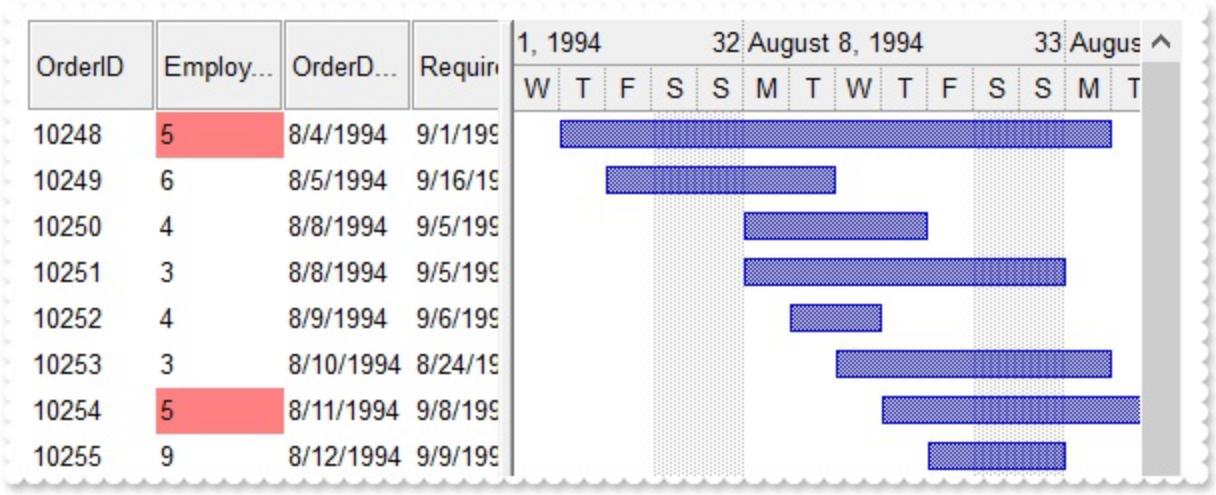
Type	Description
FormatApplyToEnum	A FormatApplyToEnum expression that indicates whether the format is applied to items or to columns. If the ApplyTo property is less than zero, the format is applied to the items.

By default, the format is applied to items. The ApplyTo property specifies whether the format is applied to the items or to the columns. If the ApplyTo property is greater or equal than zero the format is applied to the column with the index ApplyTo. For instance, if the ApplyTo property is 0, the format is applied to the cells in the first column. If the ApplyTo property is 1, the format is applied to the cells in the second column, if the ApplyTo property is 2, the format is applied to the cells in the third column, and so on. If the ApplyTo property is -1, the format is applied to items. The ApplyToBars property specifies the list of bars that the current format is applied to.

The following screen shot shows a conditional expression applied to items:



The following screen shot shows a conditional expression applied to columns:



The following VB sample bolds the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
With G2antt1.ConditionalFormats.Add("%1+%2<%0")
   .ApplyTo = 1
   .Bold = True
End With
```

The following C++ sample bolds the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
COleVariant vtEmpty;
CConditionalFormat cf = m_g2antt.GetConditionalFormats().Add( "%1+%2<%0", vtEmpty );
cf.SetBold( TRUE );
cf.SetApplyTo( 1 );
```

The following VB.NET sample bolds the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
With AxG2antt1.ConditionalFormats.Add("%1+%2<%0")
   .ApplyTo = 1
   .Bold = True
End With
```

The following C# sample bolds the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
EXG2ANTTLib.ConditionalFormat cf =
axG2antt1.ConditionalFormats.Add("%1+%2<%0",null);
```

```
cf.Bold = true;
cf.ApplyTo = (EXG2ANTTLib.FormatApplyToEnum)1;
```

The following VFP sample bolds the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
with thisform.G2antt1.ConditionalFormats.Add("%1+%2<%0")
   .Bold = .t.
   .ApplyTo = 1
endwith
```

property ConditionalFormat.ApplyToBars as String

Specifies the list of bars that the current format is applied to. The list includes the name of the bars separated by comma character.

Type	Description
String	A String expression that indicates the list of bars that the current format is applied to.

By default, the ApplyToBars property is empty, which means that the current format is not applied to any bar. The list includes the name of the bars separated by comma character. The Name property indicates the name of the bar. The ApplyTo property specifies whether the format is applied to item or cell/column. For instance, if the ApplyToBars property is "Task,Milestone", it indicates that the current format is applied to Task and Milestone bars being displayed in the chart. The following properties of the ConditionalFormat object are applied while the ApplyToBars property contains existing bars:

- The BarColor property specifies the color to be applied to bars if the conditional expression is accomplished.
- The BarOverviewColor property specifies the color to be applied to bars, in the overview portion of the control, if the conditional expression is accomplished.

The following screen shot shows different colors applied to different items, using the ConditionalFormat feature:

The following samples show how you can change the bar's color based on its length/duration:

VBA (MS Access, Excell...)

```
With G2antt1
   .BeginUpdate
   With .Columns
      .Add "Tasks"
      With .Add("Duration")
         .Def(18) = 513
         .Editor.EditType = 4
      End With
   End With
   .Items.AllowCellValueToItemBar = True
   With .Chart
      .FirstWeekDay = 1
      .LevelCount = 2
      .FirstVisibleDate = #6/6/2005#
      .PaneWidth(False) = 128
   End With
   With .ConditionalFormats.Add("%1 >= 4")
      .ApplyTo = 1 ' &H1
      .Bold = True
      .ApplyToBars = "Task"
      .BarColor = RGB(255,0,0)
      .ForeColor = .BarColor
   End With
   With .Items
      .AddBar .AddItem("Task"),"Task",#6/10/2005#,#6/13/2005#,""
      .AddBar .AddItem("Task"),"Task",#6/11/2005#,#6/16/2005#,""
      .AddBar .AddItem("Task"),"Task",#6/12/2005#,#6/15/2005#,""
   End With
   .EndUpdate
End With
```

VB6

```
With G2antt1
   .BeginUpdate
   With .Columns
      .Add "Tasks"
      With .Add("Duration")
         .Def(exCellValueToItemBarProperty) = 513
         .Editor.EditType = SpinType
      End With
   End With
   .Items.AllowCellValueToItemBar = True
   With .Chart
      .FirstWeekDay = exMonday
      .LevelCount = 2
      .FirstVisibleDate = #6/6/2005#
      .PaneWidth(False) = 128
   End With
   With .ConditionalFormats.Add("%1 >= 4")
      .ApplyTo = &H1
      .Bold = True
      .ApplyToBars = "Task"
      .BarColor = RGB(255,0,0)
      .ForeColor = .BarColor
   End With
   With .Items
      .AddBar .AddItem("Task"),"Task",#6/10/2005#,#6/13/2005#,""
      .AddBar .AddItem("Task"),"Task",#6/11/2005#,#6/16/2005#,""
      .AddBar .AddItem("Task"),"Task",#6/12/2005#,#6/15/2005#,""
   End With
   .EndUpdate
End With
```

VB.NET

```
With Exg2antt1
   .BeginUpdate()
   With .Columns
      .Add("Tasks")
```

```
      With .Add("Duration")

.set_Def(exontrol.EXG2ANTTLib.DefColumnEnum.exCellValueToItemBarProperty,513)
         .Editor.EditType = exontrol.EXG2ANTTLib.EditTypeEnum.SpinType
      End With
   End With
   .Items.AllowCellValueToItemBar = True
   With .Chart
      .FirstWeekDay = exontrol.EXG2ANTTLib.WeekDayEnum.exMonday
      .LevelCount = 2
      .FirstVisibleDate = #6/6/2005#
      .set_PaneWidth(False,128)
   End With
   With .ConditionalFormats.Add("%1 >= 4")
      .ApplyTo = &H1
      .Bold = True
      .ApplyToBars = "Task"
      .BarColor = Color.FromArgb(255,0,0)
      .ForeColor = .BarColor
   End With
   With .Items
      .AddBar(.AddItem("Task"),"Task",#6/10/2005#,#6/13/2005#,"")
      .AddBar(.AddItem("Task"),"Task",#6/11/2005#,#6/16/2005#,"")
      .AddBar(.AddItem("Task"),"Task",#6/12/2005#,#6/15/2005#,"")
   End With
   .EndUpdate()
End With
```

VB.NET for /COM

```
With AxG2antt1
   .BeginUpdate()
   With .Columns
      .Add("Tasks")
      With .Add("Duration")
         .Def(EXG2ANTTLib.DefColumnEnum.exCellValueToItemBarProperty) = 513
         .Editor.EditType = EXG2ANTTLib.EditTypeEnum.SpinType
```

```
    End With
  End With
  .Items.AllowCellValueToItemBar = True
  With .Chart
    .FirstWeekDay = EXG2ANTTLib.WeekDayEnum.exMonday
    .LevelCount = 2
    .FirstVisibleDate = #6/6/2005#
    .PaneWidth(False) = 128
  End With
  With .ConditionalFormats.Add("%1 >= 4")
    .ApplyTo = &H1
    .Bold = True
    .ApplyToBars = "Task"
    .BarColor = RGB(255,0,0)
    .ForeColor = .BarColor
  End With
  With .Items
    .AddBar(.AddItem("Task"),"Task",#6/10/2005#,#6/13/2005#,"")
    .AddBar(.AddItem("Task"),"Task",#6/11/2005#,#6/16/2005#,"")
    .AddBar(.AddItem("Task"),"Task",#6/12/2005#,#6/15/2005#,"")
  End With
  .EndUpdate()
End With
```

C++

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control
Library'

   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
```

```
EXG2ANTTLib::IColumnsPtr var_Columns = spG2antt1->GetColumns();
   var_Columns->Add(L"Tasks");
   EXG2ANTTLib::IColumnPtr var_Column = ((EXG2ANTTLib::IColumnPtr)
(var_Columns->Add(L"Duration")));
      var_Column->PutDef(EXG2ANTTLib::exCellValueToItemBarProperty,long(513));
      var_Column->GetEditor()->PutEditType(EXG2ANTTLib::SpinType);
spG2antt1->GetItems()->PutAllowCellValueToItemBar(VARIANT_TRUE);
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutFirstWeekDay(EXG2ANTTLib::exMonday);
   var_Chart->PutLevelCount(2);
   var_Chart->PutFirstVisibleDate(COleDateTime(2005,6,6,0,00,00).operator DATE());
   var_Chart->PutPaneWidth(VARIANT_FALSE,128);
EXG2ANTTLib::IConditionalFormatPtr var_ConditionalFormat = spG2antt1-
>GetConditionalFormats()->Add(L"%1 >= 4",vtMissing);
   var_ConditionalFormat->PutApplyTo(EXG2ANTTLib::FormatApplyToEnum(0x1));
   var_ConditionalFormat->PutBold(VARIANT_TRUE);
   var_ConditionalFormat->PutApplyToBars(L"Task");
   var_ConditionalFormat->PutBarColor(RGB(255,0,0));
   var_ConditionalFormat->PutForeColor(var_ConditionalFormat->GetBarColor());
EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems();
   var_Items->AddBar(var_Items-
>AddItem("Task"),"Task",COleDateTime(2005,6,10,0,00,00).operator
DATE(),COleDateTime(2005,6,13,0,00,00).operator DATE(),"",vtMissing);
   var_Items->AddBar(var_Items-
>AddItem("Task"),"Task",COleDateTime(2005,6,11,0,00,00).operator
DATE(),COleDateTime(2005,6,16,0,00,00).operator DATE(),"",vtMissing);
   var_Items->AddBar(var_Items-
>AddItem("Task"),"Task",COleDateTime(2005,6,12,0,00,00).operator
DATE(),COleDateTime(2005,6,15,0,00,00).operator DATE(),"",vtMissing);
spG2antt1->EndUpdate();
```

C++ Builder

```
G2antt1->BeginUpdate();
Exg2anttlib_tlb::IColumnsPtr var_Columns = G2antt1->Columns;
   var_Columns->Add(L"Tasks");
```

```
   Exg2anttlib_tlb::IColumnPtr var_Column = var_Columns->Add(L"Duration");
      var_Column-
>set_Def(Exg2anttlib_tlb::DefColumnEnum::exCellValueToItemBarProperty,TVariant(513

      var_Column->Editor->EditType = Exg2anttlib_tlb::EditTypeEnum::SpinType;
G2antt1->Items->AllowCellValueToItemBar = true;
Exg2anttlib_tlb::IChartPtr var_Chart = G2antt1->Chart;
   var_Chart->FirstWeekDay = Exg2anttlib_tlb::WeekDayEnum::exMonday;
   var_Chart->LevelCount = 2;
   var_Chart->set_FirstVisibleDate(TVariant(TDateTime(2005,6,6).operator double()));
   var_Chart->set_PaneWidth(false,128);
Exg2anttlib_tlb::IConditionalFormatPtr var_ConditionalFormat = G2antt1-
>ConditionalFormats->Add(L"%1 >= 4",TNoParam());
   var_ConditionalFormat->ApplyTo = Exg2anttlib_tlb::FormatApplyToEnum(0x1);
   var_ConditionalFormat->Bold = true;
   var_ConditionalFormat->ApplyToBars = L"Task";
   var_ConditionalFormat->BarColor = RGB(255,0,0);
   var_ConditionalFormat->ForeColor = var_ConditionalFormat->BarColor;
Exg2anttlib_tlb::IItemsPtr var_Items = G2antt1->Items;
   var_Items->AddBar(var_Items-
>AddItem(TVariant("Task")),TVariant("Task"),TVariant(TDateTime(2005,6,10).operator
double()),TVariant(TDateTime(2005,6,13).operator double()),TVariant(""),TNoParam());
   var_Items->AddBar(var_Items-
>AddItem(TVariant("Task")),TVariant("Task"),TVariant(TDateTime(2005,6,11).operator
double()),TVariant(TDateTime(2005,6,16).operator double()),TVariant(""),TNoParam());
   var_Items->AddBar(var_Items-
>AddItem(TVariant("Task")),TVariant("Task"),TVariant(TDateTime(2005,6,12).operator
double()),TVariant(TDateTime(2005,6,15).operator double()),TVariant(""),TNoParam());
G2antt1->EndUpdate();
```

C#

```
exg2antt1.BeginUpdate();
exontrol.EXG2ANTTLib.Columns var_Columns = exg2antt1.Columns;
   var_Columns.Add("Tasks");
   exontrol.EXG2ANTTLib.Column var_Column = (var_Columns.Add("Duration") as
```

```
exontrol.EXG2ANTTLib.Column);

var_Column.set_Def(exontrol.EXG2ANTTLib.DefColumnEnum.exCellValueToItemBarPro

      var_Column.Editor.EditType = exontrol.EXG2ANTTLib.EditTypeEnum.SpinType;
exg2antt1.Items.AllowCellValueToItemBar = true;
exontrol.EXG2ANTTLib.Chart var_Chart = exg2antt1.Chart;
   var_Chart.FirstWeekDay = exontrol.EXG2ANTTLib.WeekDayEnum.exMonday;
   var_Chart.LevelCount = 2;
   var_Chart.FirstVisibleDate =
Convert.ToDateTime("6/6/2005",System.Globalization.CultureInfo.GetCultureInfo("en-
US"));
   var_Chart.set_PaneWidth(false,128);
exontrol.EXG2ANTTLib.ConditionalFormat var_ConditionalFormat =
exg2antt1.ConditionalFormats.Add("%1 >= 4",null);
   var_ConditionalFormat.ApplyTo =
(exontrol.EXG2ANTTLib.FormatApplyToEnum)0x1;
   var_ConditionalFormat.Bold = true;
   var_ConditionalFormat.ApplyToBars = "Task";
   var_ConditionalFormat.BarColor = Color.FromArgb(255,0,0);
   var_ConditionalFormat.ForeColor = var_ConditionalFormat.BarColor;
exontrol.EXG2ANTTLib.Items var_Items = exg2antt1.Items;

var_Items.AddBar(var_Items.AddItem("Task"),"Task",Convert.ToDateTime("6/10/2005",Sy
US")),Convert.ToDateTime("6/13/2005",System.Globalization.CultureInfo.GetCultureInf
US")),"",null);

var_Items.AddBar(var_Items.AddItem("Task"),"Task",Convert.ToDateTime("6/11/2005",Sy
US")),Convert.ToDateTime("6/16/2005",System.Globalization.CultureInfo.GetCultureInf
US")),"",null);

var_Items.AddBar(var_Items.AddItem("Task"),"Task",Convert.ToDateTime("6/12/2005",Sy
US")),Convert.ToDateTime("6/15/2005",System.Globalization.CultureInfo.GetCultureInf
US")),"",null);
exg2antt1.EndUpdate();
```

JScript/JavaScript

```
<BODY onload='Init()'>
<OBJECT CLASSID="clsid:CD481F4D-2D25-4759-803F-752C568F53B7"
id="G2antt1"></OBJECT>

<SCRIPT LANGUAGE="JScript">
function Init()
{
   G2antt1.BeginUpdate();
   var var_Columns = G2antt1.Columns;
      var_Columns.Add("Tasks");
      var var_Column = var_Columns.Add("Duration");
         var_Column.Def(18) = 513;
         var_Column.Editor.EditType = 4;
   G2antt1.Items.AllowCellValueToItemBar = true;
   var var_Chart = G2antt1.Chart;
      var_Chart.FirstWeekDay = 1;
      var_Chart.LevelCount = 2;
      var_Chart.FirstVisibleDate = "6/6/2005";
      var_Chart.PaneWidth(false) = 128;
   var var_ConditionalFormat = G2antt1.ConditionalFormats.Add("%1 >= 4",null);
      var_ConditionalFormat.ApplyTo = 1;
      var_ConditionalFormat.Bold = true;
      var_ConditionalFormat.ApplyToBars = "Task";
      var_ConditionalFormat.BarColor = 255;
      var_ConditionalFormat.ForeColor = var_ConditionalFormat.BarColor;
   var var_Items = G2antt1.Items;

var_Items.AddBar(var_Items.AddItem("Task"),"Task","6/10/2005","6/13/2005","",null);

var_Items.AddBar(var_Items.AddItem("Task"),"Task","6/11/2005","6/16/2005","",null);

var_Items.AddBar(var_Items.AddItem("Task"),"Task","6/12/2005","6/15/2005","",null);
   G2antt1.EndUpdate();
}
</SCRIPT>
```

```
</BODY>
```

VBScript

```
<BODY onload='Init()'>
<OBJECT CLASSID="clsid:CD481F4D-2D25-4759-803F-752C568F53B7"
id="G2antt1"></OBJECT>

<SCRIPT LANGUAGE="VBScript">
Function Init()
   With G2antt1
      .BeginUpdate
      With .Columns
         .Add "Tasks"
         With .Add("Duration")
            .Def(18) = 513
            .Editor.EditType = 4
         End With
      End With
      .Items.AllowCellValueToItemBar = True
      With .Chart
         .FirstWeekDay = 1
         .LevelCount = 2
         .FirstVisibleDate = #6/6/2005#
         .PaneWidth(False) = 128
      End With
      With .ConditionalFormats.Add("%1 >= 4")
         .ApplyTo = 1 ' &H1
         .Bold = True
         .ApplyToBars = "Task"
         .BarColor = RGB(255,0,0)
         .ForeColor = .BarColor
      End With
      With .Items
         .AddBar .AddItem("Task"),"Task",#6/10/2005#,#6/13/2005#,""
         .AddBar .AddItem("Task"),"Task",#6/11/2005#,#6/16/2005#,""
```

```
        .AddBar .AddItem("Task"),"Task",#6/12/2005#,#6/15/2005#,""
      End With
      .EndUpdate
   End With
End Function
</SCRIPT>
</BODY>
```

C# for /COM

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Columns var_Columns = axG2antt1.Columns;
   var_Columns.Add("Tasks");
   EXG2ANTTLib.Column var_Column = (var_Columns.Add("Duration") as
EXG2ANTTLib.Column);

var_Column.set_Def(EXG2ANTTLib.DefColumnEnum.exCellValueToItemBarProperty,513

      var_Column.Editor.EditType = EXG2ANTTLib.EditTypeEnum.SpinType;
axG2antt1.Items.AllowCellValueToItemBar = true;
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.FirstWeekDay = EXG2ANTTLib.WeekDayEnum.exMonday;
   var_Chart.LevelCount = 2;
   var_Chart.FirstVisibleDate =
Convert.ToDateTime("6/6/2005",System.Globalization.CultureInfo.GetCultureInfo("en-
US"));
   var_Chart.set_PaneWidth(false,128);
EXG2ANTTLib.ConditionalFormat var_ConditionalFormat =
axG2antt1.ConditionalFormats.Add("%1 >= 4",null);
   var_ConditionalFormat.ApplyTo = (EXG2ANTTLib.FormatApplyToEnum)0x1;
   var_ConditionalFormat.Bold = true;
   var_ConditionalFormat.ApplyToBars = "Task";
   var_ConditionalFormat.BarColor =
(uint)ColorTranslator.ToWin32(Color.FromArgb(255,0,0));
   var_ConditionalFormat.ForeColor = var_ConditionalFormat.BarColor;
EXG2ANTTLib.Items var_Items = axG2antt1.Items;
```

```
var_Items.AddBar(var_Items.AddItem("Task"),"Task",Convert.ToDateTime("6/10/2005",Sy
US")),Convert.ToDateTime("6/13/2005",System.Globalization.CultureInfo.GetCultureInf
US")),"",null);

var_Items.AddBar(var_Items.AddItem("Task"),"Task",Convert.ToDateTime("6/11/2005",Sy
US")),Convert.ToDateTime("6/16/2005",System.Globalization.CultureInfo.GetCultureInf
US")),"",null);

var_Items.AddBar(var_Items.AddItem("Task"),"Task",Convert.ToDateTime("6/12/2005",Sy
US")),Convert.ToDateTime("6/15/2005",System.Globalization.CultureInfo.GetCultureInf
US")),"",null);
axG2antt1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   COM
com_Chart,com_Column,com_Columns,com_ConditionalFormat,com_Editor,com_Items

   anytype
var_Chart,var_Column,var_Columns,var_ConditionalFormat,var_Editor,var_Items;
   ;

   super();

   exg2antt1.BeginUpdate();
   var_Columns = exg2antt1.Columns(); com_Columns = var_Columns;
      com_Columns.Add("Tasks");
      var_Column = COM::createFromVariant(com_Columns.Add("Duration"));
com_Column = var_Column;

com_Column.Def(18/*exCellValueToItemBarProperty*/,COMVariant::createFromInt(513

         var_Editor = COM::createFromObject(com_Column.Editor()); com_Editor =
```

```
var_Editor;
        com_Editor.EditType(4/*SpinType*/);
    exg2antt1.Items().AllowCellValueToItemBar(true);
    var_Chart = exg2antt1.Chart(); com_Chart = var_Chart;
       com_Chart.FirstWeekDay(1/*exMonday*/);
       com_Chart.LevelCount(2);

com_Chart.FirstVisibleDate(COMVariant::createFromDate(str2Date("6/6/2005",213)));
    /*should be called during the form's activate method*/
com_Chart.PaneWidth(false,128);
    var_ConditionalFormat =
COM::createFromObject(exg2antt1.ConditionalFormats()).Add("%1 >= 4");
com_ConditionalFormat = var_ConditionalFormat;
       com_ConditionalFormat.ApplyTo(1);
       com_ConditionalFormat.Bold(true);
       com_ConditionalFormat.ApplyToBars("Task");
       com_ConditionalFormat.BarColor(WinApi::RGB2int(255,0,0));
       com_ConditionalFormat.ForeColor(com_ConditionalFormat.BarColor());
    var_Items = exg2antt1.Items(); com_Items = var_Items;

com_Items.AddBar(com_Items.AddItem("Task"),"Task",COMVariant::createFromDate(str2


com_Items.AddBar(com_Items.AddItem("Task"),"Task",COMVariant::createFromDate(str2


com_Items.AddBar(com_Items.AddItem("Task"),"Task",COMVariant::createFromDate(str2

    exg2antt1.EndUpdate();
}

/*
public void activate(boolean _active)
{
    ;

    super(_active);
```

```
    exg2antt1.Chart().PaneWidth(false,128);
}
*/
```

Delphi 8 (.NET only)

```
with AxG2antt1 do
begin
   BeginUpdate();
   with Columns do
   begin
      Add('Tasks');
      with (Add('Duration') as EXG2ANTTLib.Column) do
      begin
         Def[EXG2ANTTLib.DefColumnEnum.exCellValueToItemBarProperty] :=
TObject(513);
         Editor.EditType := EXG2ANTTLib.EditTypeEnum.SpinType;
      end;
   end;
   Items.AllowCellValueToItemBar := True;
   with Chart do
   begin
      FirstWeekDay := EXG2ANTTLib.WeekDayEnum.exMonday;
      LevelCount := 2;
      FirstVisibleDate := '6/6/2005';
      PaneWidth[False] := 128;
   end;
   with ConditionalFormats.Add('%1 >= 4',Nil) do
   begin
      ApplyTo := EXG2ANTTLib.FormatApplyToEnum($1);
      Bold := True;
      ApplyToBars := 'Task';
      BarColor := $ff;
      ForeColor := BarColor;
   end;
   with Items do
```

```
  begin
    AddBar(AddItem('Task'),'Task','6/10/2005','6/13/2005','',Nil);
    AddBar(AddItem('Task'),'Task','6/11/2005','6/16/2005','',Nil);
    AddBar(AddItem('Task'),'Task','6/12/2005','6/15/2005','',Nil);
  end;
  EndUpdate();
end
```

Delphi (standard)

```
with G2antt1 do
begin
  BeginUpdate();
  with Columns do
  begin
    Add('Tasks');
    with (IUnknown(Add('Duration')) as EXG2ANTTLib_TLB.Column) do
    begin
      Def[EXG2ANTTLib_TLB.exCellValueToItemBarProperty] := OleVariant(513);
      Editor.EditType := EXG2ANTTLib_TLB.SpinType;
    end;
  end;
  Items.AllowCellValueToItemBar := True;
  with Chart do
  begin
    FirstWeekDay := EXG2ANTTLib_TLB.exMonday;
    LevelCount := 2;
    FirstVisibleDate := '6/6/2005';
    PaneWidth[False] := 128;
  end;
  with ConditionalFormats.Add('%1 >= 4',Null) do
  begin
    ApplyTo := EXG2ANTTLib_TLB.FormatApplyToEnum($1);
    Bold := True;
    ApplyToBars := 'Task';
    BarColor := $ff;
    ForeColor := BarColor;
```

```
  end;
  with Items do
  begin
    AddBar(AddItem('Task'),'Task','6/10/2005','6/13/2005','',Null);
    AddBar(AddItem('Task'),'Task','6/11/2005','6/16/2005','',Null);
    AddBar(AddItem('Task'),'Task','6/12/2005','6/15/2005','',Null);
  end;
  EndUpdate();
end
```

VFP

```
with thisform.G2antt1
  .BeginUpdate
  with .Columns
    .Add("Tasks")
    with .Add("Duration")
      .Def(18) = 513
      .Editor.EditType = 4
    endwith
  endwith
  .Items.AllowCellValueToItemBar = .T.
  with .Chart
    .FirstWeekDay = 1
    .LevelCount = 2
    .FirstVisibleDate = {^2005-6-6}
    .PaneWidth(0) = 128
  endwith
  with .ConditionalFormats.Add("%1 >= 4")
    .ApplyTo = 1 && 0x1
    .Bold = .T.
    .ApplyToBars = "Task"
    .BarColor = RGB(255,0,0)
    .ForeColor = .BarColor
  endwith
  with .Items
    .AddBar(.AddItem("Task"),"Task",{^2005-6-10},{^2005-6-13},"")
```

```
    .AddBar(.AddItem("Task"),"Task",{^2005-6-11},{^2005-6-16},"")
    .AddBar(.AddItem("Task"),"Task",{^2005-6-12},{^2005-6-15},"")
  endwith
  .EndUpdate
endwith
```

dBASE Plus

```
local oG2antt,var_Chart,var_Column,var_Columns,var_ConditionalFormat,var_Items

oG2antt = form.Activex1.nativeObject
oG2antt.BeginUpdate()
var_Columns = oG2antt.Columns
  var_Columns.Add("Tasks")
  var_Column = var_Columns.Add("Duration")
    // var_Column.Def(18) = 513
    with (oG2antt)
      TemplateDef = [Dim var_Column]
      TemplateDef = var_Column
      Template = [var_Column.Def(18) = 513]
    endwith
    var_Column.Editor.EditType = 4
oG2antt.Items.AllowCellValueToItemBar = true
var_Chart = oG2antt.Chart
  var_Chart.FirstWeekDay = 1
  var_Chart.LevelCount = 2
  var_Chart.FirstVisibleDate = "06/06/2005"
  // var_Chart.PaneWidth(false) = 128
  with (oG2antt)
    TemplateDef = [Dim var_Chart]
    TemplateDef = var_Chart
    Template = [var_Chart.PaneWidth(false) = 128]
  endwith
var_ConditionalFormat = oG2antt.ConditionalFormats.Add("%1 >= 4")
  var_ConditionalFormat.ApplyTo = 1 /*0x1 | */
  var_ConditionalFormat.Bold = true
  var_ConditionalFormat.ApplyToBars = "Task"
```

```
   var_ConditionalFormat.BarColor = 0xff
   var_ConditionalFormat.ForeColor = var_ConditionalFormat.BarColor
var_Items = oG2antt.Items
   var_Items.AddBar(var_Items.AddItem("Task"),"Task","06/10/2005","06/13/2005","")
   var_Items.AddBar(var_Items.AddItem("Task"),"Task","06/11/2005","06/16/2005","")
   var_Items.AddBar(var_Items.AddItem("Task"),"Task","06/12/2005","06/15/2005","")
oG2antt.EndUpdate()
```

XBasic (Alpha Five)

```
Dim oG2antt as P
Dim var_Chart as P
Dim var_Column as P
Dim var_Columns as P
Dim var_ConditionalFormat as P
Dim var_Items as P

oG2antt = topparent:CONTROL_ACTIVEX1.activex
oG2antt.BeginUpdate()
var_Columns = oG2antt.Columns
   var_Columns.Add("Tasks")
   var_Column = var_Columns.Add("Duration")
      ' var_Column.Def(18) = 513
      oG2antt.TemplateDef = "Dim var_Column"
      oG2antt.TemplateDef = var_Column
      oG2antt.Template = "var_Column.Def(18) = 513"

      var_Column.Editor.EditType = 4
oG2antt.Items.AllowCellValueToItemBar = .t.
var_Chart = oG2antt.Chart
   var_Chart.FirstWeekDay = 1
   var_Chart.LevelCount = 2
   var_Chart.FirstVisibleDate = {06/06/2005}
   ' var_Chart.PaneWidth(.f.) = 128
   oG2antt.TemplateDef = "Dim var_Chart"
   oG2antt.TemplateDef = var_Chart
```

```
   oG2antt.Template = "var_Chart.PaneWidth(False) = 128"

var_ConditionalFormat = oG2antt.ConditionalFormats.Add("%1 >= 4")
   var_ConditionalFormat.ApplyTo = 1 '1 +
   var_ConditionalFormat.Bold = .t.
   var_ConditionalFormat.ApplyToBars = "Task"
   var_ConditionalFormat.BarColor = 255
   var_ConditionalFormat.ForeColor = var_ConditionalFormat.BarColor
var_Items = oG2antt.Items
   var_Items.AddBar(var_Items.AddItem("Task"),"Task",{06/10/2005},{06/13/2005},"")
   var_Items.AddBar(var_Items.AddItem("Task"),"Task",{06/11/2005},{06/16/2005},"")
   var_Items.AddBar(var_Items.AddItem("Task"),"Task",{06/12/2005},{06/15/2005},"")
oG2antt.EndUpdate()
```

Visual Objects

```
local var_Chart as IChart
local var_Column as IColumn
local var_Columns as IColumns
local var_ConditionalFormat as IConditionalFormat
local var_Items as IItems

oDCOCX_Exontrol1:BeginUpdate()
var_Columns := oDCOCX_Exontrol1:Columns
   var_Columns:Add("Tasks")
   var_Column := IColumn{var_Columns:Add("Duration")}
      var_Column:[Def,exCellValueToItemBarProperty] := 513
      var_Column:Editor:EditType := SpinType
oDCOCX_Exontrol1:Items:AllowCellValueToItemBar := true
var_Chart := oDCOCX_Exontrol1:Chart
   var_Chart:FirstWeekDay := exMonday
   var_Chart:LevelCount := 2
   var_Chart:FirstVisibleDate := SToD("20050606")
   var_Chart:[PaneWidth,false] := 128
var_ConditionalFormat := oDCOCX_Exontrol1:ConditionalFormats:Add("%1 >=
4",nil)
```

```
   var_ConditionalFormat:ApplyTo := 0x1 |
   var_ConditionalFormat:Bold := true
   var_ConditionalFormat:ApplyToBars := "Task"
   var_ConditionalFormat:BarColor := RGB(255,0,0)
   var_ConditionalFormat:ForeColor := var_ConditionalFormat:BarColor
var_Items := oDCOCX_Exontrol1:Items

var_Items:AddBar(var_Items:AddItem("Task"),"Task",SToD("20050610"),SToD("20050613



var_Items:AddBar(var_Items:AddItem("Task"),"Task",SToD("20050611"),SToD("20050616



var_Items:AddBar(var_Items:AddItem("Task"),"Task",SToD("20050612"),SToD("20050615

oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
OleObject
oG2antt,var_Chart,var_Column,var_Columns,var_ConditionalFormat,var_Items

oG2antt = ole_1.Object
oG2antt.BeginUpdate()
var_Columns = oG2antt.Columns
   var_Columns.Add("Tasks")
   var_Column = var_Columns.Add("Duration")
      var_Column.Def(18,513)
      var_Column.Editor.EditType = 4
oG2antt.Items.AllowCellValueToItemBar = true
var_Chart = oG2antt.Chart
   var_Chart.FirstWeekDay = 1
   var_Chart.LevelCount = 2
   var_Chart.FirstVisibleDate = 2005-06-06
   var_Chart.PaneWidth(false,128)
var_ConditionalFormat = oG2antt.ConditionalFormats.Add("%1 >= 4")
```

```
    var_ConditionalFormat.ApplyTo = 1 /*1 /*0x1*/ | */
    var_ConditionalFormat.Bold = true
    var_ConditionalFormat.ApplyToBars = "Task"
    var_ConditionalFormat.BarColor = RGB(255,0,0)
    var_ConditionalFormat.ForeColor = var_ConditionalFormat.BarColor
var_Items = oG2antt.Items
    var_Items.AddBar(var_Items.AddItem("Task"),"Task",2005-06-10,2005-06-13,"")
    var_Items.AddBar(var_Items.AddItem("Task"),"Task",2005-06-11,2005-06-16,"")
    var_Items.AddBar(var_Items.AddItem("Task"),"Task",2005-06-12,2005-06-15,"")
oG2antt.EndUpdate()
```

Visual DataFlex

```
Procedure OnCreate
    Forward Send OnCreate
    Send ComBeginUpdate
    Variant voColumns
    Get ComColumns to voColumns
    Handle hoColumns
    Get Create (RefClass(cComColumns)) to hoColumns
    Set pvComObject of hoColumns to voColumns
        Get ComAdd of hoColumns "Tasks" to Nothing
        Variant voColumn
        Get ComAdd of hoColumns "Duration" to voColumn
        Handle hoColumn
        Get Create (RefClass(cComColumn)) to hoColumn
        Set pvComObject of hoColumn to voColumn
            Set ComDef of hoColumn OLEexCellValueToItemBarProperty to 513
            Variant voEditor
            Get ComEditor of hoColumn to voEditor
            Handle hoEditor
            Get Create (RefClass(cComEditor)) to hoEditor
            Set pvComObject of hoEditor to voEditor
                Set ComEditType of hoEditor to OLESpinType
            Send Destroy to hoEditor
        Send Destroy to hoColumn
```

```
    Send Destroy to hoColumns
    Variant voItems
    Get ComItems to voItems
    Handle hoItems
    Get Create (RefClass(cComItems)) to hoItems
    Set pvComObject of hoItems to voItems
        Set ComAllowCellValueToItemBar of hoItems to True
    Send Destroy to hoItems
    Variant voChart
    Get ComChart to voChart
    Handle hoChart
    Get Create (RefClass(cComChart)) to hoChart
    Set pvComObject of hoChart to voChart
        Set ComFirstWeekDay of hoChart to OLEexMonday
        Set ComLevelCount of hoChart to 2
        Set ComFirstVisibleDate of hoChart to "6/6/2005"
        Set ComPaneWidth of hoChart False to 128
    Send Destroy to hoChart
    Variant voConditionalFormats
    Get ComConditionalFormats to voConditionalFormats
    Handle hoConditionalFormats
    Get Create (RefClass(cComConditionalFormats)) to hoConditionalFormats
    Set pvComObject of hoConditionalFormats to voConditionalFormats
        Variant voConditionalFormat
        Get ComAdd of hoConditionalFormats "%1 >= 4" Nothing to
voConditionalFormat
        Handle hoConditionalFormat
        Get Create (RefClass(cComConditionalFormat)) to hoConditionalFormat
        Set pvComObject of hoConditionalFormat to voConditionalFormat
            Set ComApplyTo of hoConditionalFormat to |CI$1
            Set ComBold of hoConditionalFormat to True
            Set ComApplyToBars of hoConditionalFormat to "Task"
            Set ComBarColor of hoConditionalFormat to (RGB(255,0,0))
            Set ComForeColor of hoConditionalFormat to
(ComBarColor(hoConditionalFormat))
        Send Destroy to hoConditionalFormat
    Send Destroy to hoConditionalFormats
```

```
    Variant voItems1
    Get ComItems to voItems1
    Handle hoItems1
    Get Create (RefClass(cComItems)) to hoItems1
    Set pvComObject of hoItems1 to voItems1
        Send ComAddBar of hoItems1 (ComAddItem(hoItems1,"Task")) "Task"
"6/10/2005" "6/13/2005" "" Nothing
        Send ComAddBar of hoItems1 (ComAddItem(hoItems1,"Task")) "Task"
"6/11/2005" "6/16/2005" "" Nothing
        Send ComAddBar of hoItems1 (ComAddItem(hoItems1,"Task")) "Task"
"6/12/2005" "6/15/2005" "" Nothing
    Send Destroy to hoItems1
    Send ComEndUpdate
End_Procedure
```

XBase++

```
#include "AppEvent.ch"
#include "ActiveX.ch"

PROCEDURE Main
    LOCAL oForm
    LOCAL nEvent := 0, mp1 := NIL, mp2 := NIL, oXbp := NIL
    LOCAL oG2antt
    LOCAL oChart
    LOCAL oColumn
    LOCAL oColumns
    LOCAL oConditionalFormat
    LOCAL oItems

    oForm := XbpDialog():new( AppDesktop() )
    oForm:drawingArea:clipChildren := .T.
    oForm:create( ,,{100,100}, {640,480},, .F. )
    oForm:close  := {|| PostAppEvent( xbeP_Quit )}

    oG2antt := XbpActiveXControl():new( oForm:drawingArea )
    oG2antt:CLSID  := "Exontrol.G2antt.1" /*{CD481F4D-2D25-4759-803F-
```

```
752C568F53B7}*/
   oG2antt:create(,, {10,60},{610,370} )

      oG2antt:BeginUpdate()
      oColumns := oG2antt:Columns()
         oColumns:Add("Tasks")
         oColumn := oColumns:Add("Duration")
            oColumn:SetProperty("Def",18/*exCellValueToItemBarProperty*/,513)
            oColumn:Editor():EditType := 4/*SpinType*/
      oG2antt:Items():AllowCellValueToItemBar := .T.
      oChart := oG2antt:Chart()
         oChart:FirstWeekDay := 1/*exMonday*/
         oChart:LevelCount := 2
         oChart:FirstVisibleDate := "06/06/2005"
         oChart:SetProperty("PaneWidth",.F.,128)
      oConditionalFormat := oG2antt:ConditionalFormats():Add("%1 >= 4")
         oConditionalFormat:ApplyTo := 1/*0x1+*/
         oConditionalFormat:Bold := .T.
         oConditionalFormat:ApplyToBars := "Task"
         oConditionalFormat:SetProperty("BarColor",AutomationTranslateColor(
GraMakeRGBColor  ( { 255,0,0 } )  , .F. ))
         oConditionalFormat:SetProperty("ForeColor",oConditionalFormat:BarColor())
      oItems := oG2antt:Items()
         oItems:AddBar(oItems:AddItem("Task"),"Task","06/10/2005","06/13/2005","")
         oItems:AddBar(oItems:AddItem("Task"),"Task","06/11/2005","06/16/2005","")
         oItems:AddBar(oItems:AddItem("Task"),"Task","06/12/2005","06/15/2005","")
      oG2antt:EndUpdate()

   oForm:Show()
   DO WHILE nEvent != xbeP_Quit
      nEvent := AppEvent( @mp1, @mp2, @oXbp )
      oXbp:handleEvent( nEvent, mp1, mp2 )
   ENDDO
RETURN
```

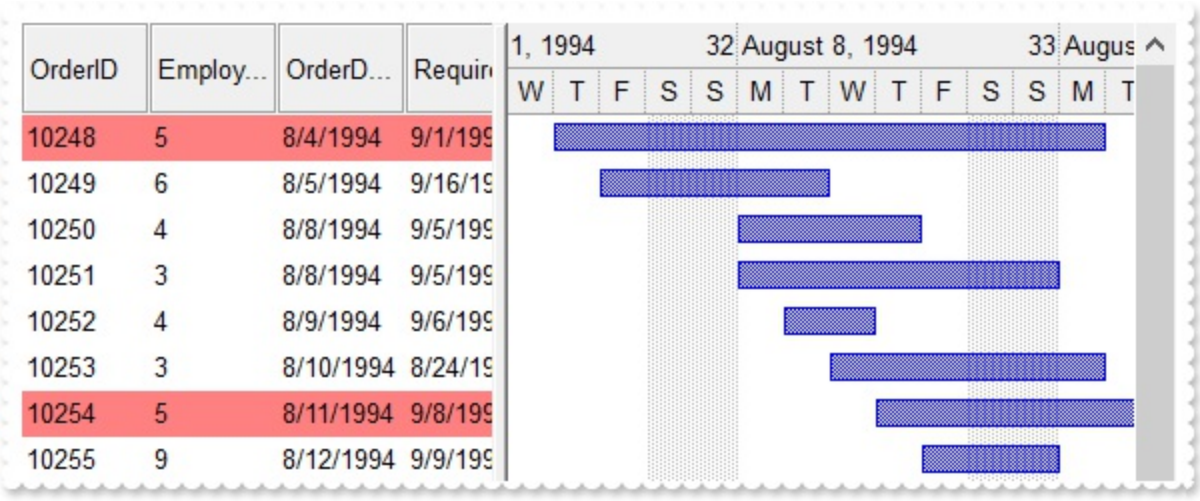
property ConditionalFormat.BackColor as Color

Retrieves or sets the background color for objects that match the condition.

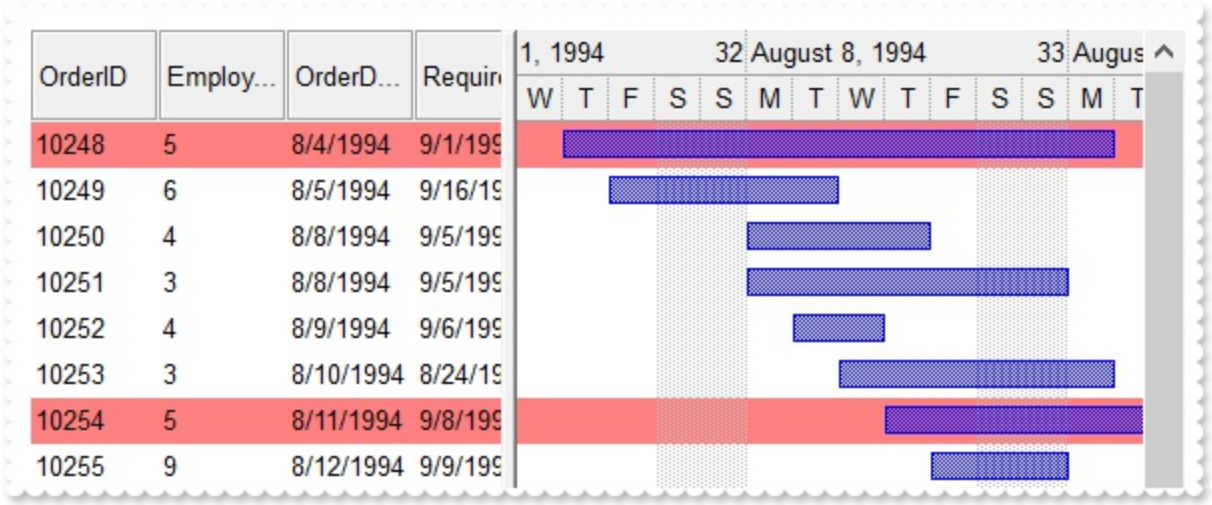
Type	Description
Color	A color expression that indicates the background color for the object that match the criteria. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

Use the BackColor property to change the background color for items or cells in the column when a certain condition is met. The ChartBackColor property applies the specified background-color (in the chart section) for items that verify the conditional expression. Use the ForeColor property to specify the foreground color for objects that match the criteria. Use the ClearBackColor method to remove the background color being set using previously the BackColor property. If the BackColor property is not set, it retrieves 0. The ApplyTo property specifies whether the ConditionalFormat object is applied to items or to cells in the column.

The following screen shot shows the control when the BackColor property is set:



The following screen shot shows the control when the BackColor and ChartBackColor properties are set:


OrderID
Employ...
OrderD...
Requir
1, 1994
32
August 8, 1994
33
Augus
W T F S S M T W T F S S M T
10248 5 8/4/1994 9/1/199
10249 6 8/5/1994 9/16/19
10250 4 8/8/1994 9/5/199
10251 3 8/8/1994 9/5/199
10252 4 8/9/1994 9/6/199
10253 3 8/10/1994 8/24/19
10254 5 8/11/1994 9/8/199
10255 9 8/12/1994 9/9/199

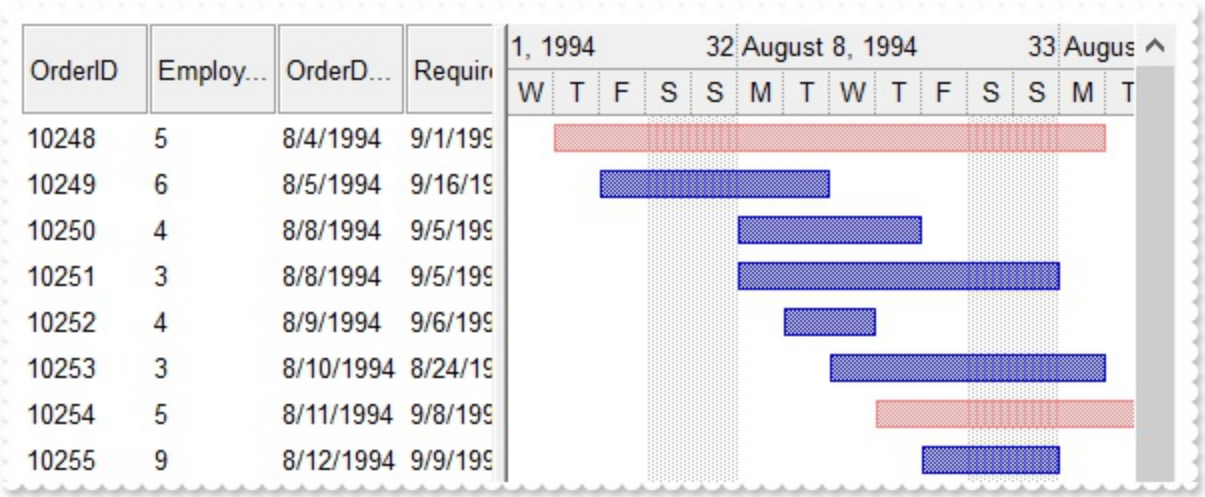
property ConditionalFormat.BarColor as Color

Specifies the color to be applied to bars if the conditional expression is accomplished.

Type	Description
Color	A color expression that indicates the color to show the bar that matches the criteria. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

By default, the BarColor property is 0. The BarColor property has effect, if the ApplyToBars property points to valid bars. The ApplyToBars property specifies the list of bars that the current format is applied to. Use the ClearBarColor method to remove the color being set using previously the BarColor property. If the BarColor property is not set, it retrieves 0. The ItemBar(exBarColor) property specifies the color to show a particular bar. The BarOverviewColor property specifies the color to be applied to bars, in the overview portion of the control, if the conditional expression is accomplished. Use the BackColor property to change the background color for items or cells in the column when a certain condition is met. Use the ForeColor property to specify the foreground color for objects that match the criteria.

The following screen shot shows the control when the BarColor property is set:



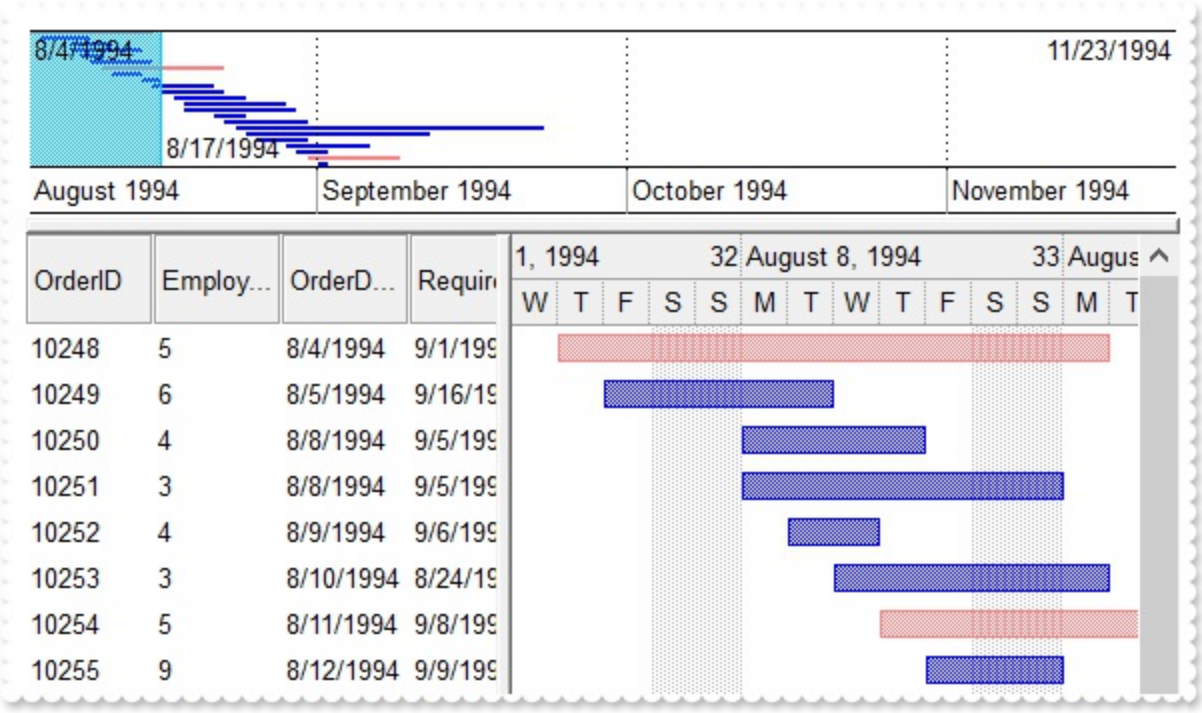
property ConditionalFormat.BarOverviewColor as Color

Specifies the color to be applied to bars, in the overview portion of the control, if the conditional expression is accomplished.

Type	Description
Color	A Color expression that specifies the color to be applied to bars, in the overview portion of the control, if the conditional expression is accomplished.

By default, the BarOverviewColor property is 0. The BarOverviewColor property has effect, if the ApplyToBars property points to valid bars. The ApplyToBars property specifies the list of bars that the current format is applied to. The OverviewVisible property shows or hides the control's overview map. Use the ClearBarOverviewColor method to remove the color being set using previously the BarOverviewColor property. If the BarColor property is not set, it retrieves 0. The ItemBar(exBarOverviewColor) property specifies the color to show a different color in the overview part of the control, for a particular bar.

The following screen shot shows the control when the BarColor and BarOverviewColor properties are set:



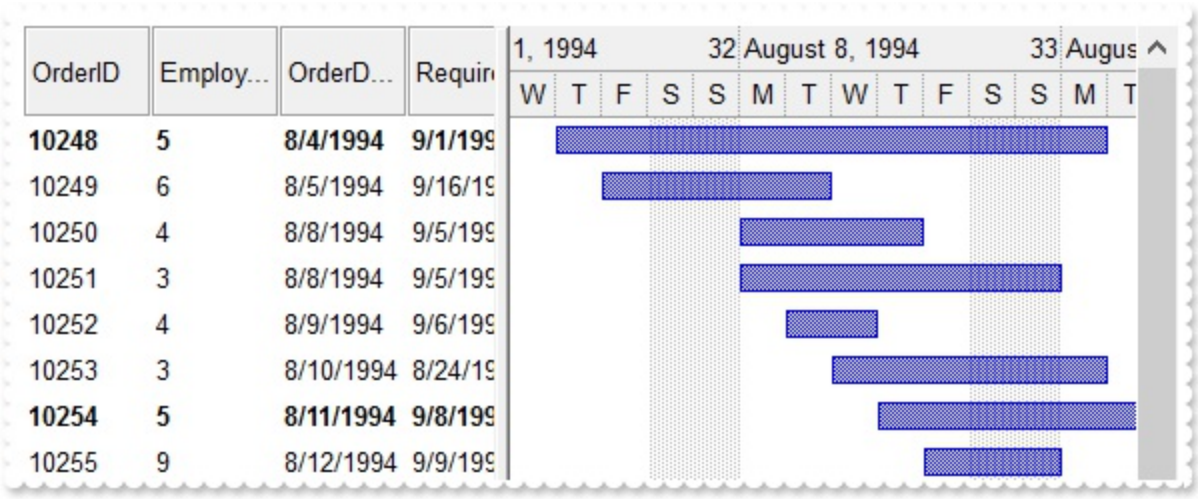
property ConditionalFormat.Bold as Boolean

Bolds the objects that match the condition.

Type	Description
Boolean	A boolean expression that indicates whether the objects should appear in bold.

The ApplyTo property specifies whether the ConditionalFormat object is applied to items or to cells in the column.

The following screen shot shows the control when the Bold property is set:



The following VB sample bolds all cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
With G2antt1.ConditionalFormats.Add("%1+%2<%0")
   .ApplyTo = 1
   .Bold = True
End With
```

The following C++ sample bolds all cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
COleVariant vtEmpty;
CConditionalFormat cf = m_g2antt.GetConditionalFormats().Add( "%1+%2<%0", vtEmpty
);
cf.SetBold( TRUE );
cf.SetApplyTo( 1 );
```

The following VB.NET sample bolds all cells in the second column (1), if the sum between

second and third column (2) is less than the value in the first column (0):

```
With AxG2antt1.ConditionalFormats.Add("%1+%2<%0")
   .ApplyTo = 1
   .Bold = True
End With
```

The following C# sample bolds all cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
EXG2ANTTLib.ConditionalFormat cf =
axG2antt1.ConditionalFormats.Add("%1+%2<%0",null);
cf.Bold = true;
cf.ApplyTo = (EXG2ANTTLib.FormatApplyToEnum)1;
```

The following VFP sample bolds all cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
with thisform.G2antt1.ConditionalFormats.Add("%1+%2<%0")
   .Bold = .t.
   .ApplyTo = 1
endwith
```

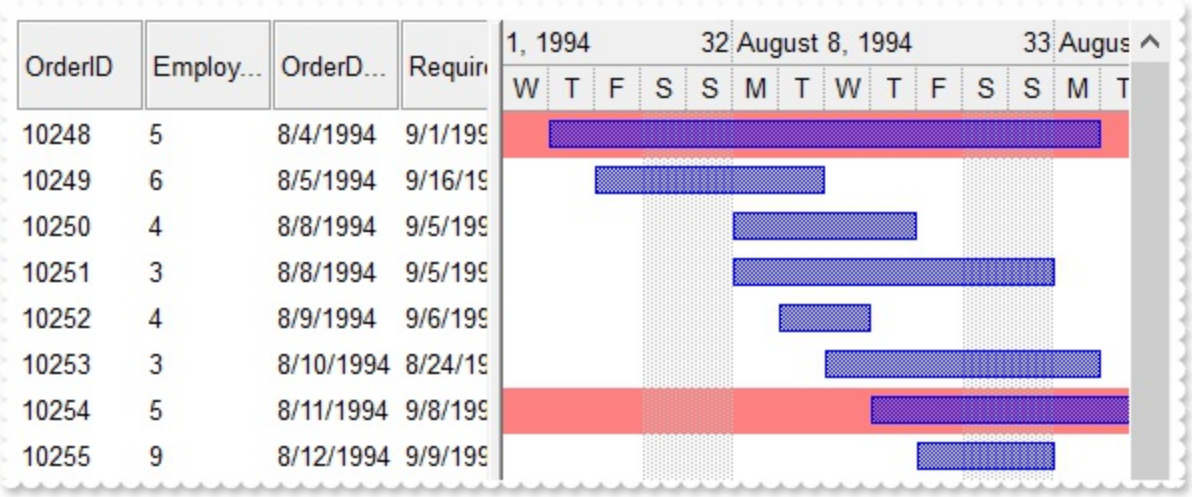
property ConditionalFormat.ChartBackColor as Color

Specifies the color to be applied to item's background in the chart section of the control, if the conditional expression is accomplished.

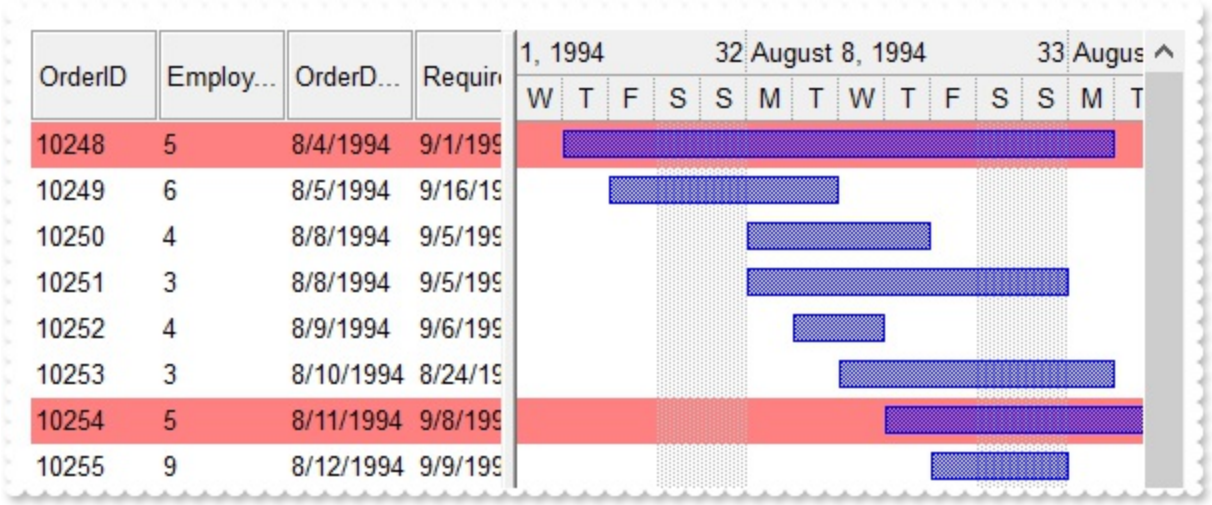
Type	Description
Color	A color expression that indicates the color to show the bar that matches the criteria. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

The ChartBackColor property applies the specified background-color (in the chart section) for items that verify the conditional expression. The BackColor property retrieves or sets the background color for objects that match the condition. The ChartChartBackColor method clears the item's background in the chart section of the control. If the ChartBackColor property is not set, it retrieves 0. The ApplyTo property specifies whether the ConditionalFormat object is applied to items or to cells in the column.

The following screen shot shows the control when the ChartBackColor property is set:



The following screen shot shows the control when the BackColor and ChartBackColor properties are set:


OrderID
Employ...
OrderD...
Requir
1, 1994
32
August 8, 1994
33
Augus
W T F S S M T W T F S S M T
10248 5 8/4/1994 9/1/199
10249 6 8/5/1994 9/16/19
10250 4 8/8/1994 9/5/199
10251 3 8/8/1994 9/5/199
10252 4 8/9/1994 9/6/199
10253 3 8/10/1994 8/24/19
10254 5 8/11/1994 9/8/199
10255 9 8/12/1994 9/9/199

method ConditionalFormat.ClearBackColor ()

Clears the background color.

Type	Description

Use the ClearBackColor method to remove the background color being set using previously the BackColor property. If the BackColor property is not set, it retrieves 0.

method ConditionalFormat.ClearBarColor ()

Clears the bar's color.

Type	Description

Use the ClearBarColor method to remove the color being set using previously the BarColor property. If the BarColor property is not set, it retrieves 0.

method ConditionalFormat.ClearBarOverviewColor ()

Clears the bar's overview color.

Type	Description

Use the ClearBarOverviewColor method to remove the color being set using previously the BarOverviewColor property. If the BarOverviewColor property is not set, it retrieves 0.

method ConditionalFormat.ClearChartBackColor ()

Clears the item's background in the chart section of the control.

Type	Description

You can use the ClearChartBackColor method to prevent changing the item's background color on the chart panel of the control, if the conditional expression is applied to the item. The ChartBackColor property specifies the color to be applied to item's background in the chart section of the control, if the conditional expression is accomplished.

method ConditionalFormat.ClearForeColor ()

Clears the foreground color.

Type	Description

Use the ClearBackColor method to remove the foreground color being set using previously the ForeColor property. If the ForeColor property is not set, it retrieves 0.

property ConditionalFormat.Enabled as Boolean

Specifies whether the condition is enabled or disabled.

Type	Description
Boolean	A boolean expression that indicates whether the expression is enabled or disabled.

By default, all expressions are enabled. A format is applied only if the expression is valid and enabled. Use the Expression property to specify the format's formula. The Valid property checks whether the formula is valid or not valid. Use the Enabled property to disable applying the format for the moment. Use the Remove method to remove an expression from ConditionalFormats collection.

property ConditionalFormat.Expression as String

Indicates the expression being used in the conditional format.

Type	Description
String	A formal expression that indicates the formula being used in formatting. For instance, "%0+%1>%2", highlights the cells or the items, when the sum between first two columns is greater than the value in the third column

The conditional formatting feature allows you to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula. The Expression property specifies a formula that indicates the criteria to format the items or the columns. Use the ApplyTo property to specify when the items or the columns are formatted. Use the Add method to specify the expression at adding time. The Expression property may include variables, constants, operators or ( ) parenthesis. A variable is defined as %n, where n is the index of the column (zero based). For instance, the %0 indicates the first column, the %1, indicates the second column, and so on. A constant is a float expression (for instance, 23.45). Use the Valid property checks whether the expression is syntactically correct, and can be evaluated. If the expression contains a variable that is not known, 0 value is used instead. For instance, if your control has 2 columns, and the expression looks like "%2 +%1 ", the %2 does not exist, 0 is used instead. When the control contains two columns the known variables are %0 and %1.

The expression may be a combination of variables, constants, strings, dates and operators. A string is delimited by ", ` or ' characters, and inside they can have the starting character preceded by \ character, ie "\"This is a quote\"". A date is delimited by two # characters, ie #1/31/2001 10:00# means the January 31th, 2001, 10:00 AM.

The expression supports cell's identifiers as follows:

- ***%0**, **%1**, **%2**, ... specifies the value of the cell in the column with the index 0, 1 2, ... The CellValue property specifies the cell's value. For instance, "%0 format ``" formats the value on the cell with the index 0, using current regional setting, while "int(%1)" converts the value of the column with the index 1, to integer.*
- ***%C0**, **%C1**, **%C2**, ... specifies the caption of the cell, or the string the cell displays in the column with the index 0, 1 2, ... The CellCaption property specifies the cell's caption. The cell's value may be different than what the cell displays as a string. For instance, let's say a cell display HTML format. The %0 returns the html format including the HTML tags, while %C0 returns the cell's content as string without HTML tags. For instance, "upper(%C1)" converts the caption of the cell with the index 1, to upper case, while "%C0 left 2" returns the leftmost two characters on the cell with the index 0.*

- *%CD0, %CD1, %CD2, ... specifies the cell's extra data in the column with the index 0, 1 2, ... The CellData property associates any extra/user data to a cell. For instance, "%CD0 = `your user data`" specifies all cells whose CellData property is `your user data`, on the column with the index 0.*
- *%CS0, %CS1, %CS2, ... specifies the cell's state in the column with the index 0, 1 2, ... The CellState property specifies the cell's state, and so it indicates whether the cell is checked or un-checked. For instance, "%CS0" defines all checked items on the column with the index 0, or "not %CS1" defines all un-checked items in the column with the index 1.*

This property/method supports predefined constants and operators/functions as described here.

Samples:

1. "**1**", highlights all cells or items. Use this form, when you need to highlight all cells or items in the column or control.
2. "**%0 >= 0**", highlights the cells or items, when the cells in the first column have the value greater or equal with zero
3. "**%0 = 1 and %1 = 0**", highlights the cells or items, when the cells in the first column have the value equal with 0, and the cells in the second column have the value equal with 0
4. "**%0+%1>%2**", highlights the cells or the items, when the sum between first two columns is greater than the value in the third column
5. "**%0+%1 > %2+%3**", highlights the cells or items, when the sum between first two columns is greater than the sum between third and forth column.
6. "**%0+%1 >= 0 and (%2+%3)/2 < %4-5**", highlights the cells or the items, when the sum between first two columns is greater than 0 and the half of the sum between third and forth columns is less than fifth column minus 5.
7. "**%0 startwith 'A'**" specifies the cells that starts with A
8. "**%0 endwith 'Bc'**" specifies the cells that ends with Bc
9. "**%0 contains 'aBc'**" specifies the cells that contains the aBc string
10. "**lower(%0) contains 'abc'**" specifies the cells that contains the abc, AbC, ABC, and so on
11. "**upper(%0)**" retrieves the uppercase string
12. "**len(%0)>0**" specifies the not blanks cells
13. "**len %0 = 0**" specifies the blanks cells

The conditional format feature may change the cells and items as follows:

- Bold property. Bolds the cell or items
- Italic property. Indicates whether the cells or items should appear in italic.
- StrikeOut property. Indicates whether the cells or items should appear in strikeout.

- Underline property. Underlines the cells or items
- Font property. Changes the font for cells or items.
- BackColor property. Changes the background color for cells or items, supports skins as well.
- ForeColor property. Changes the foreground color for cells or items.

The following VB samples bolds all items when the sum between first two columns is greater than 0:

```
G2antt1.ConditionalFormats.Add("%0+%1>0").Bold = True
```

The following C++ sample bolds all items when the sum between first two columns is greater than 0:

```
COleVariant vtEmpty;
m_g2antt.GetConditionalFormats().Add( "%0+%1>0", vtEmpty ).SetBold( TRUE );
```

The following VB.NET sample bolds all items when the sum between first two columns is greater than 0:

```
AxG2antt1.ConditionalFormats.Add("%0+%1>0").Bold = True
```

The following C# sample bolds all items when the sum between first two columns is greater than 0:

```
axG2antt1.ConditionalFormats.Add("%0+%1>0", null).Bold = true
```

The following VFP sample bolds all items when the sum between first two columns is greater than 0:

```
thisform.G2antt1.ConditionalFormats.Add("%0+%1>0").Bold = .t.
```

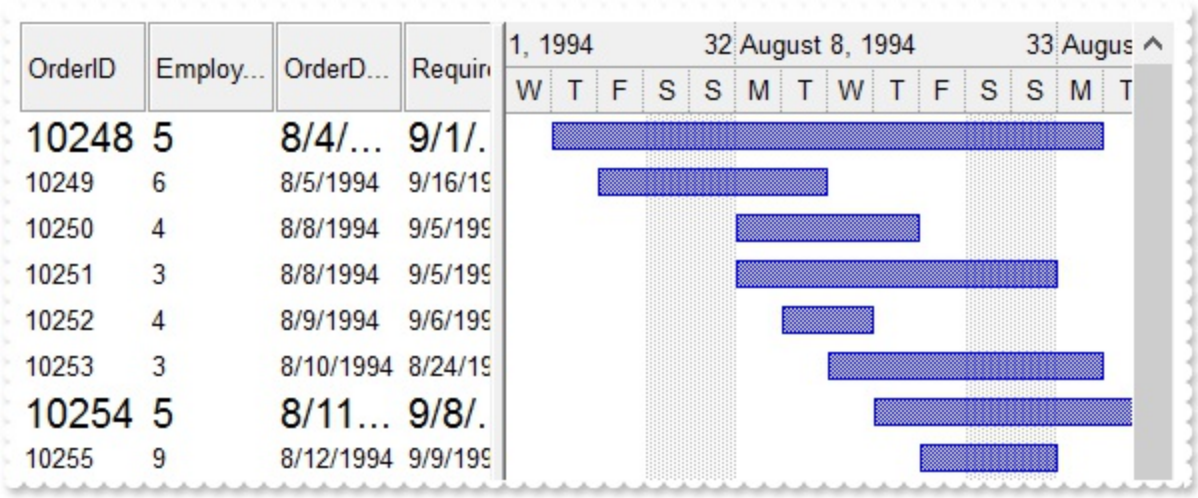
property ConditionalFormat.Font as IFontDisp

Retrieves or sets the font for objects that match the criteria.

Type	Description
IFontDisp	A Font object that's applied to items or columns.

Use the Font property to change the font for items or columns that match the criteria. Use the Font property only, if you need to change to a different font.

The following screen shot shows the control when the Font property is set:



You can change directly the font attributes, like follows:

- Bold property. Bolds the cell or items
- Italic property. Indicates whether the cells or items should appear in italic.
- StrikeOut property. Indicates whether the cells or items should appear in strikeout.
- Underline property. Underlines the cells or items

The following VB sample changes the font for ALL cells in the first column:

```
With G2antt1.ConditionalFormats.Add("1")
   .ApplyTo = 0
   Set .Font = New StdFont
   With .Font
      .Name = "Comic Sans MS"
   End With
End With
```

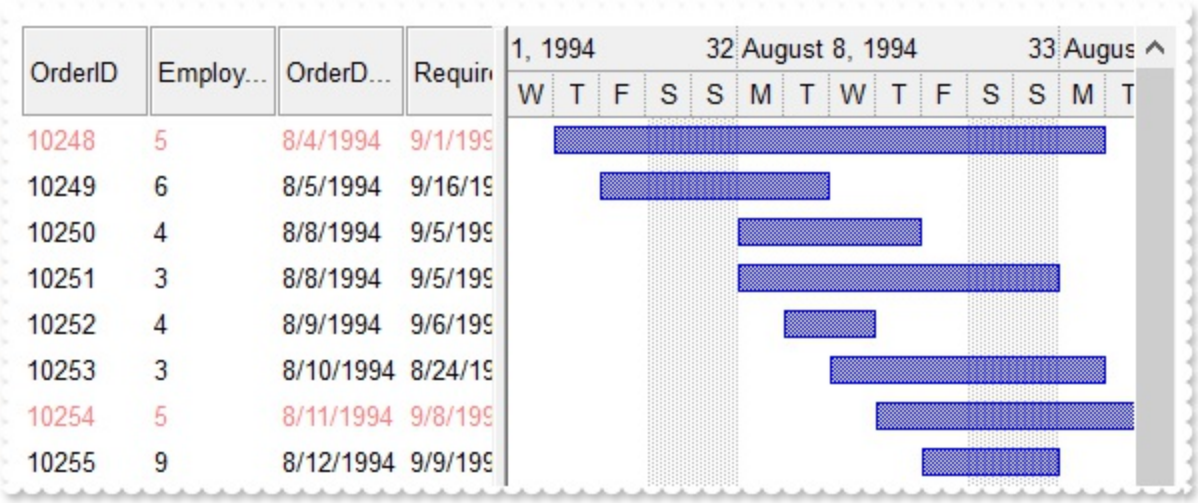
property ConditionalFormat.ForeColor as Color

Retrieves or sets the foreground color for objects that match the condition.

Type	Description
Color	A color expression that indicates the foreground color for the object that match the criteria.

Use the ForeColor property to specify the foreground color for objects that match the criteria. Use the BackColor property to change the background color for items or cells in the column when a certain condition is met. Use the ClearForeColor method to remove the foreground color being set using previously the ForeColor property. If the ForeColor property is not set, it retrieves 0. The ApplyTo property specifies whether the ConditionalFormat object is applied to items or to cells in the column.

The following screen shot shows the control when the ForeColor property is set:



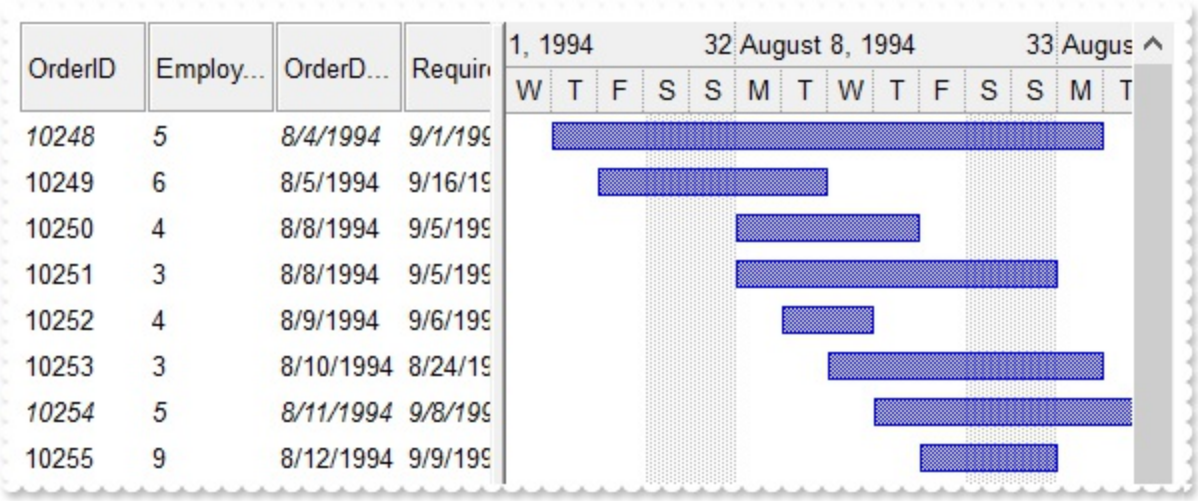
property ConditionalFormat.Italic as Boolean

Specifies whether the objects that match the condition should appear in italic.

Type	Description
Boolean	A boolean expression that indicates whether the objects should look in italic.

The ApplyTo property specifies whether the ConditionalFormat object is applied to items or to cells in the column.

The following screen shot shows the control when the Italic property is set:



The following VB sample makes italic the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
With G2antt1.ConditionalFormats.Add("%1+%2<%0")
   .ApplyTo = 1
   .Italic = True
End With
```

The following C++ sample makes italic the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
COleVariant vtEmpty;
CConditionalFormat cf = m_g2antt.GetConditionalFormats().Add( "%1+%2<%0", vtEmpty );
cf.SetItalic( TRUE );
cf.SetApplyTo( 1 );
```

The following VB.NET sample makes italic the cells in the second column (1), if the sum

between second and third column (2) is less than the value in the first column (0):

```
With AxG2antt1.ConditionalFormats.Add("%1+%2<%0")
   .ApplyTo = 1
   .Italic = True
End With
```

The following C# sample makes italic the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
EXG2ANTTLib.ConditionalFormat cf =
axG2antt1.ConditionalFormats.Add("%1+%2<%0",null);
cf.Italic = true;
cf.ApplyTo = (EXG2ANTTLib.FormatApplyToEnum)1;
```

The following VFP sample makes italic the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
with thisform.G2antt1.ConditionalFormats.Add("%1+%2<%0")
   .Italic = .t.
   .ApplyTo = 1
endwith
```

property ConditionalFormat.Key as Variant

Checks whether the expression is syntactically correct.

Type	Description
Variant	A String expression that indicates the key of the element

The Key property indicates the key of the element. Use the Add method to specify a key at adding time. Use the Remove method to remove a formula giving its key.

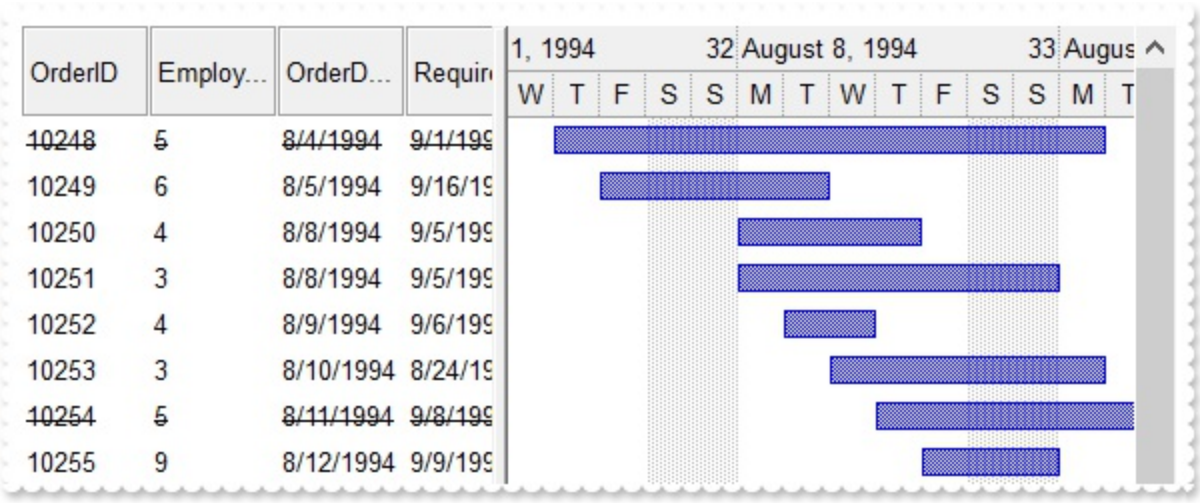
property ConditionalFormat.StrikeOut as Boolean

Specifies whether the objects that match the condition should appear in strikeout.

Type	Description
Boolean	A Boolean expression that indicates whether the objects should appear in strikeout.

The ApplyTo property specifies whether the ConditionalFormat object is applied to items or to cells in the column.

The following screen shot shows the control when the StrikeOut property is set:



The following VB sample applies strikeout font attribute to cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
With G2antt1.ConditionalFormats.Add("%1+%2<%0")
   .ApplyTo = 1
   .Bold = True
End With
```

The following C++ sample applies strikeout font attribute to cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
COleVariant vtEmpty;
CConditionalFormat cf = m_g2antt.GetConditionalFormats().Add( "%1+%2<%0", vtEmpty );
cf.SetBold( TRUE );
cf.SetApplyTo( 1 );
```

The following VB.NET sample applies strikeout font attribute to cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
With AxG2antt1.ConditionalFormats.Add("%1+%2<%0")
   .ApplyTo = 1
   .Bold = True
End With
```

The following C# sample applies strikeout font attribute to cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
EXG2ANTTLib.ConditionalFormat cf =
axG2antt1.ConditionalFormats.Add("%1+%2<%0",null);
cf.Bold = true;
cf.ApplyTo = (EXG2ANTTLib.FormatApplyToEnum)1;
```

The following VFP sample applies strikeout font attribute to cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
with thisform.G2antt1.ConditionalFormats.Add("%1+%2<%0")
   .Bold = .t.
   .ApplyTo = 1
endwith
```

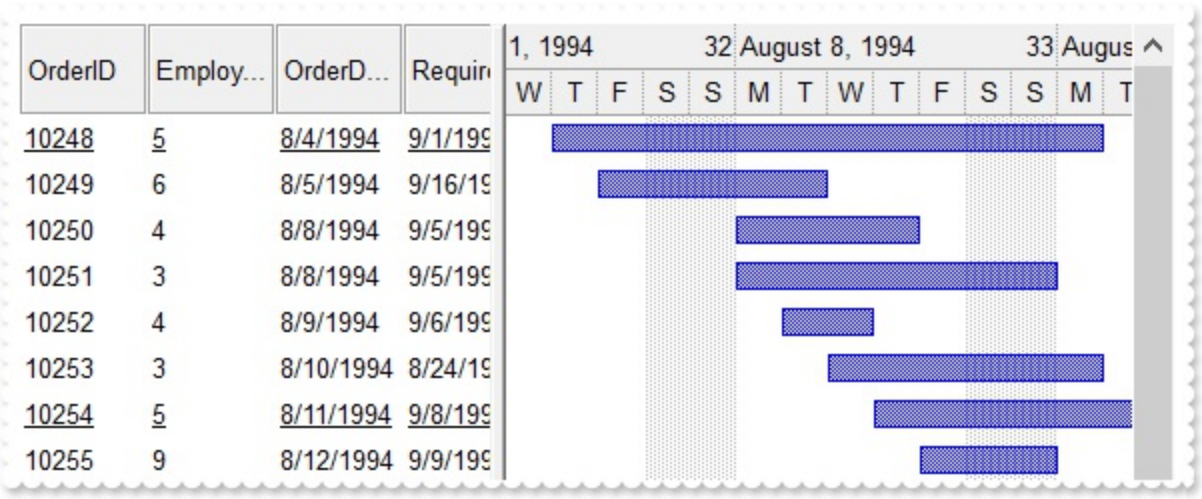
property ConditionalFormat.Underline as Boolean

Underlines the objects that match the condition.

Type	Description
Boolean	A boolean expression that indicates whether the objects are underlined.

The ApplyTo property specifies whether the ConditionalFormat object is applied to items or to cells in the column.

The following screen shot shows the control when the Underline property is set:



The following VB sample underlines the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
With G2antt1.ConditionalFormats.Add("%1+%2<%0")
   .ApplyTo = 1
   .Underline = True
End With
```

The following C++ sample underlines the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
COleVariant vtEmpty;
CConditionalFormat cf = m_g2antt.GetConditionalFormats().Add( "%1+%2<%0", vtEmpty );
cf.SetUnderline( TRUE );
cf.SetApplyTo( 1 );
```

The following VB.NET sample underlines the cells in the second column (1), if the sum

between second and third column (2) is less than the value in the first column (0):

```
With AxG2antt1.ConditionalFormats.Add("%1+%2<%0")
   .ApplyTo = 1
   .Underline = True
End With
```

The following C# sample underlines the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
EXG2ANTTLib.ConditionalFormat cf =
axG2antt1.ConditionalFormats.Add("%1+%2<%0",null);
cf.Underline = true;
cf.ApplyTo = (EXG2ANTTLib.FormatApplyToEnum)1;
```

The following VFP sample underlines the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
with thisform.G2antt1.ConditionalFormats.Add("%1+%2<%0")
   .Underline = .t.
   .ApplyTo = 1
endwith
```

property ConditionalFormat.Valid as Boolean

Checks whether the expression is syntactically correct.

Type	Description
Boolean	A boolean expression that indicates whether the Expression property is valid.

Use the Valid property to check whether the Expression formula is valid. The conditional format is not applied to objects if expression is not valid, or the Enabled property is false. An empty expression is not valid. Use the Enabled property to disable applying the format to columns or items. Use the Remove method to remove an expression from ConditionalFormats collection. The Verify property checks whether the current conditional expression is applied to the giving item.

property ConditionalFormat.Verify (Item as HITEM) as Boolean

Verifies the current conditional format if it is applied to the giving item.

Type	Description
Item as HITEM	A Long expression that specifies whether the conditional expression is applied.
Boolean	A Boolean expression that specifies whether the current conditional expression is applied to the giving item.

The Verify property checks whether the current conditional expression is applied to the giving item. The Verify property returns True, if the conditional expression is valid, and the condition is meet for the specified item. The Verify property returns False, if the conditional expression is not valid or the conditional expression is not applied to the specified item. The Valid property checks whether the Expression formula is valid.

ConditionalFormats object

The conditional formatting feature allows you to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula. The ConditionalFormats collection holds a collection of ConditionalFormat objects. Use the ConditionalFormats property to access the control's ConditionalFormats collection .The ConditionalFormats collection supports the following properties and methods:

Name	Description
Add	Adds a new expression to the collection and returns a reference to the newly created object.
Clear	Removes all expressions in a collection.
Count	Returns the number of objects in a collection.
Item	Returns a specific expression.
Remove	Removes a specific member from the collection.

method ConditionalFormats.Add (Expression as String, [Key as Variant])

Adds a new expression to the collection and returns a reference to the newly created object.

Type	Description
Expression as String	A formal expression that indicates the formula being used when the format is applied. Please check the Expression property that shows the syntax of the expression that may be used. For instance, the **"%0 >= 10 and %1 > 67.23"** means all cells in the first column with the value less or equal than 10, and all cells in the second column with a value greater than 67.23
Key as Variant	A string or long expression that indicates the key of the expression being added. If the Key parameter is missing, by default, the current index in the ConditionalFormats collection is used.

Return	Description
ConditionalFormat	A ConditionalFormat object that indicates the newly format being added.

The conditional formatting feature allows you to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula. Use the Add method to format cells or items based on values. Use the Add method to add new ConditionalFormat objects to the ConditionalFormats collection. By default, the ConditionalFormats collection is empty. A ConditionalFormat object indicates a formula and a format to apply to cells or items. The ApplyTo property specifies whether the ConditionalFormat object is applied to items or to cells in the column. Use the Expression property to retrieve or set the formula. Use the Key property to retrieve the key of the object. Use the Refresh method to update the changes on the control's content.

The conditional format feature may change the cells and items as follows:

- Bold property. Bolds the cell or items
- Italic property. Indicates whether the cells or items should appear in italic.
- StrikeOut property. Indicates whether the cells or items should appear in strikeout.
- Underline property. Underlines the cells or items
- Font property. Changes the font for cells or items.
- BackColor property. Changes the background color for cells or items, supports skins as well.
- ForeColor property. Changes the foreground color for cells or items.

The following VB sample bolds all items when the sum between first two columns is greater than 0:

```
G2antt1.ConditionalFormats.Add("%0+%1>0").Bold = True
```

The following VB sample bolds the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
With G2antt1.ConditionalFormats.Add("%1+%2<%0")
   .ApplyTo = 1
   .Bold = True
End With
```

The following C++ sample bolds all items when the sum between first two columns is greater than 0:

```
COleVariant vtEmpty;
m_g2antt.GetConditionalFormats().Add( "%0+%1>0", vtEmpty ).SetBold( TRUE );
```

The following C++ sample bolds the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
COleVariant vtEmpty;
CConditionalFormat cf = m_g2antt.GetConditionalFormats().Add( "%1+%2<%0", vtEmpty
);
cf.SetBold( TRUE );
cf.SetApplyTo( 1 );
```

The following VB.NET sample bolds all items when the sum between first two columns is greater than 0:

```
AxG2antt1.ConditionalFormats.Add("%0+%1>0").Bold = True
```

The following VB.NET sample bolds the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
With AxG2antt1.ConditionalFormats.Add("%1+%2<%0")
   .ApplyTo = 1
   .Bold = True
End With
```

The following C# sample bolds all items when the sum between first two columns is greater than 0:

```
axG2antt1.ConditionalFormats.Add("%0+%1>0", null).Bold = true
```

The following C# sample bolds the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
EXG2ANTTLib.ConditionalFormat cf =
axG2antt1.ConditionalFormats.Add("%1+%2<%0",null);
cf.Bold = true;
cf.ApplyTo = (EXG2ANTTLib.FormatApplyToEnum)1;
```

The following VFP sample bolds all items when the sum between first two columns is greater than 0:

```
thisform.G2antt1.ConditionalFormats.Add("%0+%1>0").Bold = .t.
```

The following VFP sample bolds the cells in the second column (1), if the sum between second and third column (2) is less than the value in the first column (0):

```
with thisform.G2antt1.ConditionalFormats.Add("%1+%2<%0")
   .Bold = .t.
   .ApplyTo = 1
endwith
```

method ConditionalFormats.Clear ()

Removes all expressions in a collection.

Type	Description

Use the Clear method to remove all objects in the collection. Use the Remove method to remove a particular object from the collection. Use the Enabled property to disable a conditional format.

property ConditionalFormats.Count as Long

Returns the number of objects in a collection.

Type	Description
Long	A long expression that counts the number of elements in the collection.

Use the Item and Count property to enumerate the elements in the collection. Use the Expression property to get the expression of the format.

The following VB sample enumerates all elements in the ConditionalFormats collection:

```
Dim c As ConditionalFormat
For Each c In G2antt1.ConditionalFormats
   Debug.Print c.Expression
Next
```

The following VB sample enumerates all elements in the ConditionalFormats collection:

```
Dim i As Integer
With G2antt1.ConditionalFormats
   For i = 0 To .Count - 1
      Debug.Print .Item(i).Expression
   Next
End With
```

The following C++ sample enumerates all elements in the ConditionalFormats collection:

```
for ( long i = 0; i < m_g2antt.GetConditionalFormats().GetCount(); i++ )
{
   CConditionalFormat cf = m_g2antt.GetConditionalFormats().GetItem( COleVariant( i ) );
   OutputDebugString( cf.GetExpression() );
}
```

The following VB.NET sample enumerates all elements in the ConditionalFormats collection:

```
Dim c As EXG2ANTTLib.ConditionalFormat
For Each c In AxG2antt1.ConditionalFormats
   System.Diagnostics.Debug.Write(c.Expression)
Next
```

The following VB.NET sample enumerates all elements in the ConditionalFormats collection:

```
Dim i As Integer
With AxG2antt1.ConditionalFormats
   For i = 0 To .Count - 1
      System.Diagnostics.Debug.Write(.Item(i).Expression)
   Next
End With
```

The following C# sample enumerates all elements in the ConditionalFormats collection:

```
foreach (EXG2ANTTLib.ConditionalFormat c in axG2antt1.ConditionalFormats)
   System.Diagnostics.Debug.Write(c.Expression);
```

The following C# sample enumerates all elements in the ConditionalFormats collection:

```
for (int i = 0; i < axG2antt1.ConditionalFormats.Count; i++)
   System.Diagnostics.Debug.Write(axG2antt1.ConditionalFormats[i].Expression);
```

The following VFP sample enumerates all elements in the ConditionalFormats collection:

```
with thisform.G2antt1.ConditionalFormats
   for i = 0 to .Count - 1
      wait .Item(i).Expression
   next
endwith
```

property ConditionalFormats.Item (Key as Variant) as ConditionalFormat

Returns a specific expression.

Type	Description
Key as Variant	A long expression that indicates the index of the element being accessed, or a string expression that indicates the key of the element being accessed.
ConditionalFormat	A ConditionalFormat object being returned.

Use the Item and Count property to enumerate the elements in the collection. Use the Expression property to get the expression of the format. Use the Key property to get the key of the format.

The following VB sample enumerates all elements in the ConditionalFormats collection:

```
Dim c As ConditionalFormat
For Each c In G2antt1.ConditionalFormats
   Debug.Print c.Expression
Next
```

The following VB sample enumerates all elements in the ConditionalFormats collection:

```
Dim i As Integer
With G2antt1.ConditionalFormats
   For i = 0 To .Count - 1
      Debug.Print .Item(i).Expression
   Next
End With
```

The following C++ sample enumerates all elements in the ConditionalFormats collection:

```
for ( long i = 0; i < m_g2antt.GetConditionalFormats().GetCount(); i++ )
{
   CConditionalFormat cf = m_g2antt.GetConditionalFormats().GetItem( COleVariant( i ) );
   OutputDebugString( cf.GetExpression() );
}
```

The following VB.NET sample enumerates all elements in the ConditionalFormats collection:

```
Dim c As EXG2ANTTLib.ConditionalFormat
```

```
For Each c In AxG2antt1.ConditionalFormats
   System.Diagnostics.Debug.Write(c.Expression)
Next
```

The following VB.NET sample enumerates all elements in the ConditionalFormats collection:

```
Dim i As Integer
With AxG2antt1.ConditionalFormats
   For i = 0 To .Count - 1
      System.Diagnostics.Debug.Write(.Item(i).Expression)
   Next
End With
```

The following C# sample enumerates all elements in the ConditionalFormats collection:

```
foreach (EXG2ANTTLib.ConditionalFormat c in axG2antt1.ConditionalFormats)
   System.Diagnostics.Debug.Write(c.Expression);
```

The following C# sample enumerates all elements in the ConditionalFormats collection:

```
for (int i = 0; i < axG2antt1.ConditionalFormats.Count; i++)
   System.Diagnostics.Debug.Write(axG2antt1.ConditionalFormats[i].Expression);
```

The following VFP sample enumerates all elements in the ConditionalFormats collection:

```
with thisform.G2antt1.ConditionalFormats
   for i = 0 to .Count - 1
      wait .Item(i).Expression
   next
endwith
```

method ConditionalFormats.Remove (Key as Variant)

Removes a specific member from the collection.

Type	Description
Key as Variant	A Long or String expression that indicates the key of the element to be removed.

Use the Remove method to remove a particular object from the collection. Use the Enabled property to disable a conditional format. Use the Clear method to remove all objects in the collection.

Editor object

The Editor object holds information about an editor. A cell or a column may have assigned an editor. Use the Editor property to access the column's editor. Use the CellEditor property to access the cell's editor. The Editor object supports the following properties and methods

Name	Description
AddButton	Adds a new button to the editor with specified key and aligns it to the left or right side of the editor.
AddItem	Adds a new item to the editor's list.
Appearance	Retrieves or sets the editor's appearance
ButtonWidth	Specifies the width of the buttons in the editor.
ClearButtons	Clears the buttons collection.
ClearItems	Clears the items collection.
DropDown	Displays the drop down list.
DropDownAlignment	Retrieves or sets a value that indicates the item's alignment in the editor's drop-down list.
DropDownAutoWidth	Retrieves or sets a value that indicates whether the editor's drop-down window list is automatically computed to fit the entire list.
DropDownMinWidth	Specifies the minimum drop-down list width if the DropDownAutoWidth is False.
DropDownRows	Retrieves or sets a value that indicates the maximum number of visible rows in the editor's drop- down list.
DropDownVisible	Retrieves or sets a value that indicates whether the editor's drop down button is visible or hidden.
EditType	Retrieves or sets a value that indicates the type of the contained editor.
ExpandAll	Expands all items in the editor's list.
ExpandItem	Expandes or collapses an item in the editor's list.
FindItem	Finds an item given its value or caption.
InsertItem	Inserts a child item to the editor's list.
ItemToolTip	Gets or sets the text displayed when the mouse pointer hovers over a predefined item.
Locked	Determines whether the editor is locked or unlocked.
	Retrieves or sets a value that indicates the mask used by

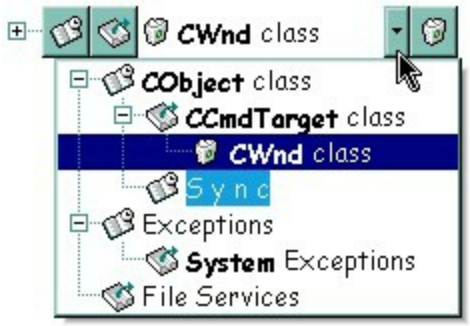
Mask	the editor.
MaskChar	Retrieves or sets a value that indicates the character used for masking.
Numeric	Specifies whether the editor enables numeric values only.
Option	Specifies an option for the editor.
PartialCheck	Retrieves or sets a value that indicates whether the associated check box has two or three states.
PopupAppearance	Retrieves or sets a value that indicates the drop-down window's appearance.
RemoveButton	Removes a button given its key.
RemoveItem	Removes an item from the editor's predefined values list.
SortItems	Sorts the list of items in the editor.
UserEditor	Specifies the control's identifier and the control's runtime license key when EditType is UserEditor.
UserEditorObject	Gets the user editor object when EditType is UserEditor.

method Editor.AddButton (Key as Variant, [Image as Variant], [Align as Variant], [ToolTip as Variant], [ToolTipTitle as Variant], [ShortcutKey as Variant])

Adds a new button to the editor with the given Key and aligned to the left or to the right side. You can specify the button's tooltip too.

Type	Description
Key as Variant	A Variant value that indicates the button's key. The ButtonClick event passes this value to Key parameter
Image as Variant	A long expression that indicates the index of button's icon. The index is valid for Images collection. By default the button has no icon associated.
Align as Variant	An AlignmentEnum expression that defines the button's alignment.
ToolTip as Variant	A string expression that indicates the the button's tooltip description. The tooltip shows up when cursor hovers the button. The ToolTip parameter may include buitl-in HTML tags.
ToolTipTitle as Variant	A string expression that indicates the tooltip's title.
ShortcutKey as Variant	A short expression that indicates the shortcut key being used to simulate clicking the button. The lower byte indicates the code of the virtual key, and the higher byte indicates the states for SHIFT, CTRL and ALT keys (last insignificant bits in the higher byte). The ShortcutKey expression could be 256 *((shift ? 1 : 0) + (ctrl ? 2 : 0) + (alt ? 4 : 0)) + vbKeyCode, For instance, a combination like CTRL + F3 is 256 * 2 + vbKeyF3, SHIFT + CTRL + F2 is 256 *(1 + 2) + vbKeyF2, and SHIFT + CTRL + ALT + F5 is 256 * (1 + 2 + 4) + vbKeyF5.

Use the AddButton method to add multiple buttons to the editor. Make sure that you are using unique keys for the buttons in the same editor, else the previous button is replaced. The editor doesn't allow two buttons with the same key. Use the ButtonWidth property to set the button's width. If the user clicks on one of the editor buttons, the ButtonClick event is fired. Use CellHasButton property to display's the cell's caption as a button. Use the RemoveButton method to remove a button that was previously added using the AddButton method. Use the ClearButtons method to clear the entire collection of buttons added with AddButton method. The control fires the ButtonClick event when the user clicks a button.



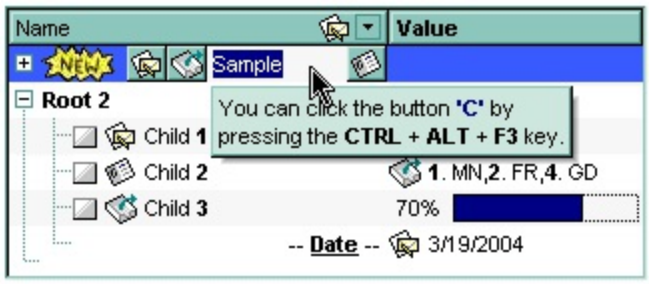
The following VB sample adds multiple buttons to the column's editor:

```
With G2antt1
   With .Columns.Add("Column 1")
      .HeaderBold = True
      .HeaderImage = 1
      With .Editor
         .EditType = DropDownListType
         .DropDownAutoWidth = False

         .AddItem 0, "CS is bad", 3
         .AddItem 1, "xTras is the worst", 3
         .AddItem 2, "Yes, I agree!", 3

         .ButtonWidth = 24
         .AddButton "Key1", 1, , "This is a bit of text that should appear while the cursor is
over the button", "Information"
         .AddButton "Key2", 2
         .AddButton "Key3", 3, AlignmentEnum.RightAlignment
         .AddButton "Key3", 4, AlignmentEnum.RightAlignment
      End With
   End With
End With
```

The following VB sample adds an editor to the first visible item with three buttons, each of the button has a shortcut key to activate it using the keyboard:

```
With G2antt1.Items
   With .CellEditor(.FirstVisibleItem, 0)
      .ButtonWidth = 20
      .EditType = EditType
```

```
    .AddButton "A", 1, , "You can click the button <fgcolor=000080><b>'A'</b>
</fgcolor> by pressing the <b>F3</b> key.", , vbKeyF3
    .AddButton "B", 2, RightAlignment, "You can click the button <fgcolor=000080>
<b>'B'</b></fgcolor> by pressing the <b>CTRL + F3</b> key.", , vbKeyF3 + (256 * (2))
    .AddButton "C", 3, , "You can click the button <fgcolor=000080><b>'C'</b>
</fgcolor> by pressing the <b>CTRL + ALT + F3</b> key.", , vbKeyF3 + (256 * (2 + 4))
  End With
End With
```



The following C++ sample adds an EditType editor with a button to the first visible item:

```
#include "Items.h"
#include "Editor.h"
COleVariant vtMissing; V_VT( &vtMissing) = VT_ERROR;
CItems items = m_g2antt.GetItems();
CEditor editor = items.GetCellEditor( COleVariant( items.GetFirstVisibleItem() ),
COleVariant( long(0) ) );
editor.SetEditType( 1 /*EditType*/ );
editor.AddButton( COleVariant("A"), COleVariant("1"), vtMissing, vtMissing, vtMissing,
vtMissing  );
```

The following VB.NET sample adds an EditType editor with a button to the first visible item:

```
With AxG2antt1.Items
  With .CellEditor(.FirstVisibleItem, 0)
    .EditType = EXG2ANTTLib.EditTypeEnum.EditType
    .AddButton("A", 1)
  End With
End With
```

The following C# sample adds an EditType editor with a button to the first visible item:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
```

```
EXG2ANTTLib.Editor editor = items.get_CellEditor(items.FirstVisibleItem, 0);
editor.EditType = EXG2ANTTLib.EditTypeEnum.EditType;
editor.AddButton("A", 1, null, null, null, null);
```

The following VFP sample adds an EditType editor with a button to the first visible item:

```
with thisform.G2antt1.Items
   With .CellEditor(.FirstVisibleItem, 0)
      .EditType = 1 && EditType
      .AddButton("A", 1)
   EndWith
endwith
```

method Editor.AddItem (Value as Long, Caption as String, [Image as Variant])

Adds a new item to editor's predefined list.

Type	Description
Value as Long	A long expression that defines an unique predefined value
Caption as String	A string expression that specifies the HTML caption associated with the value. The format of the Caption parameter is "key|captionŚcaptionŚ...Ścaption", which indicates an item with the giving key / identifier, which displays multiple captions. The Caption allows using the following special characters: • | character (pipe, vertical bar, ALT + 126) defines the key or identifier of the item to add. Currently, the key is used by a **DropDownListType** editor to specify string codes rather numeric values for the cell's value (CellValue property) • Ś character (vertical broken bar, ALT + 221) defines captions for multiple columns. The Ś character can be escaped, so \Ś displays the Ś character (available for DropDownType, DropDownListType and PickEditType editors, 20.0+) For instance: • "New York City" defines the "**New York** City" item • "NYC|New York City" the "**New York** City" item with the "NYC" as key or identifier • "NYC|New York CityŚ783.8 km²Ś8.42 mil" defines the "**New York** City" item with the "NYC" as key or identifier and sub-captions 783.8 km² and 8.42 mil (in separated columns) • "New York CityŚ783.8 km²Ś8.42 mil" defines the "**New York** City" item and sub-captions 783.8 km² and 8.42 mil (in separated columns)
Image as Variant	A long expression that indicates the index of the item's icon (1-based). Use the Images method to assign a list of icons to the control.

Use the AddItem method to add new items to the editor's predefined list. Use the InsertItem method to insert child items to the editor's predefined list. If the AddItem method uses a Value already defined, the old item is replaced. The AddItem method has effect for the following type of editors: **DropDownType, DropDownListType, PickEditType,** and **CheckListType.** Check each EditType value for what Value argument should contain. Use the RemoveItem method to remove a particular item from the predefined list. Use the ClearItems method to clear the entire list of predefined values. Use the SortItems to sort the items. Use the ItemToolTip property to assign a tooltip to a predefined item into a drop down list. Use the Refresh method update immediately the cell's content when adding new items to a drop down list editor. The Caption parameter supports HTML tags listed here here.

The following VB sample adds items to a CheckListType editor:

```
With G2antt1
   With .Columns.Add("CheckList").Editor
      .EditType = CheckListType
      .AddItem &H1, "ReadOnly", 1
      .AddItem &H2, "Hidden", 2
      .AddItem &H4, "System", 3
      .AddItem &H10, "Directory", 4
      .AddItem &H20, "Archive", 5
      .AddItem &H80, "Normal", 7
      .AddItem &H100, "Temporary", 8
   End With
.Items.AddItem &H1 + &H2
End With
```

The following VB sample adds predefined values to drop down list editor:

```
With G2antt1.Items
   With .CellEditor(.FirstVisibleItem, 0)
      .EditType = DropDownListType
      .AddItem 0, "No border", 1
      .AddItem 1, "Single Border", 2
      .AddItem 2, "Double Border", 3
   End With
End With
```

The following C++ sample adds predefined values to drop down list editor:

```
With G2antt1.Items
   With .CellEditor(.FirstVisibleItem, 0)
      .EditType = DropDownListType
      .AddItem 0, "No border", 1
      .AddItem 1, "Single Border", 2
      .AddItem 2, "Double Border", 3
   End With
End With
```

The following VB.NET sample adds predefined values to drop down list editor:

```
With AxG2antt1.Items
   With .CellEditor(.FirstVisibleItem, 0)
      .EditType = EXG2ANTTLib.EditTypeEnum.DropDownListType
      .AddItem(0, "No border", 1)
      .AddItem(1, "Single Border", 2)
      .AddItem(2, "Double Border", 3)
   End With
End With
```

The following C# sample adds predefined values to drop down list editor:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
EXG2ANTTLib.Editor editor = items.get_CellEditor(items.FirstVisibleItem, 0);
editor.EditType = EXG2ANTTLib.EditTypeEnum.DropDownListType ;
editor.AddItem(0, "No border", 1);
editor.AddItem(1, "Single border", 2);
editor.AddItem(2, "Double border", 3);
```

The following VFP sample adds predefined values to drop down list editor:

```
with thisform.G2antt1.Items
   With .CellEditor(.FirstVisibleItem, 0)
      .EditType = 3 && DropDownList
      .AddItem(0, "No border", 1)
      .AddItem(1, "Single Border", 2)
      .AddItem(2, "Double Border", 3)
```

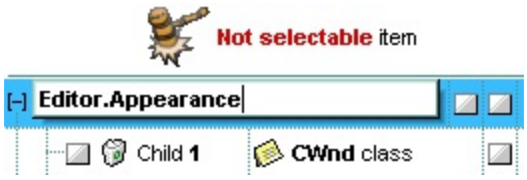
```
    EndWith
endwith
```

property Editor.Appearance as InplaceAppearanceEnum

Retrieves or sets the control's appearance

Type	Description
InplaceAppearanceEnum	An InplaceAppearanceEnum expression that defines the editor's appearance.

Use the Appearance property to change the editor's border style. Use the PopupAppearance property to define the appearance for editor's drop-down window, if it exists. By default, the editor's Appearance is NoApp.



property Editor.ButtonWidth as Long

Specifies the width of the buttons in the editor.

Type	Description
Long	A long expression that defines the width of the buttons in the editor, added using the AddButton method.

Use the ButtonWidth property to increase or decrease the width of buttons in the editor. The button's height is the same with the ItemHeight property. If the ButtonWidth property is zero (0), the control hides the buttons. Use the DropDownVisible property to hide the editor's drop down button.

method Editor.ClearButtons ()

Clears the buttons collection.

Type	Description

Use the ClearButtons method to clear the list of buttons that were added using the AddButton method. Use the RemoveButton method to remove a particular button, given its key. Use the ButtonWidth property to hide all the buttons.

method Editor.ClearItems ()

Clears the items collection.

Type	Description

The ClearItems method clears the predefined values that were added using the AddItem or InsertItem method. Use the RemoveItem method to remove a predefined value. Use the DropDownVisible property to hide the drop-down window.

method Editor.DropDown ()

Displays the drop down list.

Type	Description

The DropDown method shows the drop down portion of the cell's editor. The DropDown method has effect only if the editor has a drop down portion. The following editors have a drop down portion: DropDownType, DropDownListType, CheckListType, DateType, ColorType, FontType, PictureType, PickEditType, ColorListType, MemoDropDownType or CalculatorType. Use the AddItem, InsertItem method to add predefined value. Use the RemoveItem method to remove a predefined value. Use the DropDownVisible property to hide the drop down button, if it exists.

The following VB sample shows the drop down portion of an editor as soon as a cell is focused:

```
Private Sub G2antt1_FocusChanged()
  With G2antt1
    Dim i As Long
    i = .FocusColumnIndex
    With G2antt1.Items
      If (.CellEditorVisible(.FocusItem, i)) Then
        Dim e As EXG2ANTTLibCtl.Editor
        Set e = G2antt1.Columns(i).Editor
        If .HasCellEditor(.FocusItem, i) Then
          Set e = .CellEditor(.FocusItem, i)
        End If
        If Not e Is Nothing Then
          e.DropDown
        End If
      End If
    End With
  End With
End Sub
```

The following C++ sample shows the drop down portion of an editor as soon as a cell is focused:

```
#include "Columns.h"
```

```
#include "Column.h"
#include "Editor.h"
#include "Items.h"
void OnFocusChangedG2antt1()
{
   if ( IsWindow( m_g2antt.m_hWnd ) )
   {
      CItems items = m_g2antt.GetItems();
      COleVariant vtMissing; V_VT( &vtMissing) = VT_ERROR;
      COleVariant vtFocusCell( items.GetItemCell( items.GetFocusItem(),
COleVariant(m_g2antt.GetFocusColumnIndex())));
      if ( m_g2antt.GetSelectColumnInner() > 0 )
         vtFocusCell = items.GetInnerCell( vtMissing, vtFocusCell,
COleVariant(m_g2antt.GetSelectColumnInner()));
      if ( items.GetCellEditorVisible( vtMissing, vtFocusCell ) )
      {
         CEditor editor;
         if ( items.GetHasCellEditor( vtMissing, vtFocusCell ) )
            editor = items.GetCellEditor( vtMissing, vtFocusCell );
         else
         {
            CColumn column( m_g2antt.GetColumns().GetItem( COleVariant(
m_g2antt.GetFocusColumnIndex() ) ) );
            editor = column.GetEditor();
         }
         if ( editor.m_lpDispatch != NULL )
            editor.DropDown();
      }
   }
}
```

The following VB.NET sample shows the drop down portion of an editor as soon as a cell is focused:

```
Private Sub AxG2antt1_FocusChanged(ByVal sender As Object, ByVal e As
System.EventArgs) Handles AxG2antt1.FocusChanged
   With AxG2antt1.Items
      Dim focusCell As Object = .ItemCell(.FocusItem, AxG2antt1.FocusColumnIndex)
```

```
      If (AxG2antt1.SelectColumnInner > 0) Then
         focusCell = .InnerCell(Nothing, focusCell, AxG2antt1.SelectColumnInner)
      End If
      If (.CellEditorVisible(, focusCell)) Then
         Dim ed As EXG2ANTTLib.Editor =
AxG2antt1.Columns(AxG2antt1.FocusColumnIndex).Editor
         If (.HasCellEditor(, focusCell)) Then
            ed = .CellEditor(, focusCell)
         End If
         ed.DropDown()
      End If
   End With
End Sub
```

The following C# sample shows the drop down portion of an editor as soon as a cell is focused:

```
private void axG2antt1_FocusChanged(object sender, EventArgs e)
{
   EXG2ANTTLib.Items items = axG2antt1.Items;
   object focusCell = items.get_ItemCell(items.FocusItem, axG2antt1.FocusColumnIndex);
   if ( axG2antt1.SelectColumnInner > 0 )
      focusCell = items.get_InnerCell( null, focusCell, axG2antt1.SelectColumnInner );
   if ( items.get_CellEditorVisible(null, focusCell ))
   {
      EXG2ANTTLib.Editor editor =
axG2antt1.Columns[axG2antt1.FocusColumnIndex].Editor;
      if ( items.get_HasCellEditor(null, focusCell ))
         editor = items.get_CellEditor(null, focusCell );
      if ( editor != null )
         editor.DropDown();
   }
}
```

The following VFP sample shows the drop down portion of an editor as soon as a cell is focused:

```
*** ActiveX Control Event ***
```

```
with thisform.G2antt1.Items
   local ed
   ed = thisform.G2antt1.Columns(thisform.G2antt1.FocusColumnIndex).Editor
   if ( .HasCellEditor(.FocusItem, thisform.G2antt1.FocusColumnIndex) )
      ed = .CellEditor(.FocusItem, thisform.G2antt1.FocusColumnIndex)
   endif
   ed.DropDown()
endwith
```

property Editor.DropDownAlignment as AlignmentEnum

Retrieves or sets a value that indicates the item's alignment in the editor's drop-down list.

Type	Description
AlignmentEnum	An AlignmentEnum expression that indicates the item's alignment into the editor's drop-down list.

Use the DropDownAlignment property to align the items in the editor's drop-down list. Use the Alignment property to align a column. Use the CellHAlignment property to align a cell. The property has effect only for the drop down type editors.

The following VB sample aligns the predefined items to the right (the editor is assigned to the column):

```
With G2antt1
   .TreeColumnIndex = -1
   With .Columns.Add("CheckList")
   .Alignment = RightAlignment
      With .Editor
         .EditType = CheckListType
         .DropDownAlignment = RightAlignment
         .AddItem &H1, "ReadOnly", 1
         .AddItem &H2, "Hidden", 2
         .AddItem &H4, "System", 3
         .AddItem &H10, "Directory", 4
         .AddItem &H20, "Archive", 5
         .AddItem &H80, "Normal", 7
         .AddItem &H100, "Temporary", 8
      End With
   End With
   .Items.AddItem &H1 + &H2
End With
```

In the above sample, the TreeColumnIndex is set to -1, because the Alignment property is not applied for column that displays the hierarchy.

The following VB sample adds an editor that aligns its predefined items to the right:

```
With G2antt1.Items
```

```
  With .CellEditor(.FirstVisibleItem, 0)
    .DropDownAlignment = RightAlignment
    .EditType = DropDownListType
    .AddItem 0, "No border"
    .AddItem 1, "Single Border"
    .AddItem 2, "Double Border"
  End With
End With
```

The following C++ sample adds an editor that aligns its predefined items to the right:

```
#include "Items.h"
#include "Editor.h"
COleVariant vtMissing; V_VT( &vtMissing) = VT_ERROR;
CItems items = m_g2antt.GetItems();
CEditor editor = items.GetCellEditor( COleVariant( items.GetFirstVisibleItem() ),
COleVariant( long(0) ) );
editor.SetEditType( 3 /*DropDownList*/ );
editor.SetDropDownAlignment( 2 /*RightAlignment*/ );
editor.AddItem( 0, "No border", vtMissing );
editor.AddItem( 1, "Single border", vtMissing );
editor.AddItem( 2, "Double border", vtMissing );
```

The following VB.NET sample adds an editor that aligns its predefined items to the right:

```
With AxG2antt1.Items
  With .CellEditor(.FirstVisibleItem, 0)
    .DropDownAlignment = EXG2ANTTLib.AlignmentEnum.RightAlignment
    .EditType = EXG2ANTTLib.EditTypeEnum.DropDownListType
    .AddItem(0, "No border")
    .AddItem(1, "Single Border")
    .AddItem(2, "Double Border")
  End With
End With
```

The following C# sample adds an editor that aligns its predefined items to the right:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
EXG2ANTTLib.Editor editor = items.get_CellEditor(items.FirstVisibleItem, 0);
```

```
editor.EditType = EXG2ANTTLib.EditTypeEnum.DropDownListType ;
editor.DropDownAlignment = EXG2ANTTLib.AlignmentEnum.RightAlignment;
editor.AddItem(0, "No border", null);
editor.AddItem(1, "Single border", null);
editor.AddItem(2, "Double border", null);
```

The following VFP sample adds an editor that aligns its predefined items to the right:

```
with thisform.G2antt1.Items
   With .CellEditor(.FirstVisibleItem, 0)
      .EditType = 3 && DropDownList
      .DropDownAlignment = 2 && RightAlignment
      .AddItem(0, "No border")
      .AddItem(1, "Single Border")
      .AddItem(2, "Double Border")
   EndWith
endwith
```

property Editor.DropDownAutoWidth as DropDownWidthType

Retrieves or sets a value that indicates whether the editor's drop-down window list is automatically computed to fit the entire list.

Type	Description
DropDownWidthType	A DropDownWidthType expression that indicates whether the editor's drop- down list width is automatically computed to fit the entire list.

Use the DropDownAutoWidth property to specify when you let the control computes the drop-down list width. The DropDownMinWidth property specifies the minimum width for the drop down portion of the editor. By default, the DropDownAutoWidth property is exDropDownAutoWidth. Use the DropDown method to programmatically show the drop down portion of an editor. Use the DropDownRows property to specify the number of visible rows in the drop down portion of the control.

property Editor.DropDownMinWidth as Long

Specifies the minimum drop-down list width if the DropDownAutoWidth is False.

Type	Description
Long	A long expression that specifies the minimum drop- down list width if the DropDownAutoWidth is False

The DropDownMinWidth property has no effect if the DropDownAutoWidth property is True. Use the DropDown method to programmatically show the drop down portion of an editor. Use the DropDownRows property to specify the number of visible rows in the drop down portion of the control.

property Editor.DropDownRows as Long

Retrieves or sets a value that indicates the maximum number of visible rows in the editor's drop- down list.

Type	Description
Long	A long expression that indicates the maximum number of visible rows in the editor's drop- down list.

Use the DropDownRows property to specify the maximum number of visible rows in the editor's drop-down list. By default, the DropDownRows property is set to 7. The DropDownRows property has effect for the following types: DropDownType, DropDownListType, PickEditType, CheckListType and FontType. Use the AddItem method to add predefined values to the drop down portion of the control.

property Editor.DropDownVisible as Boolean

Retrieves or sets a value that indicates whether the editor's drop down button is visible or hidden.

Type	Description
Boolean	A boolean value that indicates whether the editor's drop down button is visible or hidden.

Use the DropDownVisible property to hide the editor's drop-down button. Use the ButtonWidth property to hide the editor buttons. Use the AddItem, InsertItem method to add predefined values to the drop down list. Use the Refresh method update immediately the cell's content when adding new items to a drop down list editor. If the drop down button is hidden, the editor can't open its drop down portion if the user double clicks the editor, or presses the F4 key.

The following VB sample to check whether the editor's drop down portion is visible:

```
Private Declare Function FindWindow Lib "user32" Alias "FindWindowA" (ByVal
lpClassName As String, ByVal lpWindowName As String) As Long

Private Function isDropped()
   ' Specifies whether the control's drop down portion is visible or not
   isDropped = Not FindWindow("HostPopupWindow", "") = 0
End Function
```

The following VB sample advance to the next line when the ENTER key is pressed, and does the default action when an editor of drop down type is opened:

```
 Private Sub G2antt1_KeyDown(KeyCode As Integer, Shift As Integer)
   If (KeyCode = 13) Then
      If Not isDropped() Then
         KeyCode = vbKeyDown
      End If
   End If
End Sub
```

The following VB sample hides the drop down button:

```
With G2antt1.Items
   With .CellEditor(.FirstVisibleItem, 0)
```

```
    .EditType = EXG2ANTTLibCtl.DropDownListType
    .AddItem 0, "0 - <b>No</b>", 1
    .AddItem 1, "1 - <b>Yes</b>", 2
    .DropDownVisible = False
  End With
  .CellValue(.FirstVisibleItem, 0) = 1
  .CellValueFormat(.FirstVisibleItem, 0) = exHTML
End With
```

The following C++ sample hides the drop down button:

```
#include "Items.h"
#include "Editor.h"
COleVariant vtMissing; V_VT( &vtMissing) = VT_ERROR;
CItems items = m_g2antt.GetItems();
COleVariant vtItem( items.GetFirstVisibleItem() ), vtColumn( long(0) );
CEditor editor = items.GetCellEditor( vtItem, vtColumn );
editor.SetEditType( 3 /*DropDownListType*/ );
editor.AddItem( 0, "0 - <b>No</b>", vtMissing );
editor.AddItem( 1, "1- <b>Yes</b>", vtMissing );
editor.SetDropDownVisible( FALSE );
items.SetCellValue( vtItem, vtColumn, COleVariant( long(1)) );
items.SetCellValueFormat( vtItem, vtColumn, 1 /*exHTML*/ );
```

The following VB.NET sample hides the drop down button:

```
With AxG2antt1.Items
  With .CellEditor(.FirstVisibleItem, 0)
    .EditType = EXG2ANTTLib.EditTypeEnum.DropDownListType
    .AddItem(0, "0 - <b>No</b>", 1)
    .AddItem(1, "1 - <b>Yes</b>", 2)
    .DropDownVisible = False
  End With
  .CellValue(.FirstVisibleItem, 0) = 1
  .CellValueFormat(.FirstVisibleItem, 0) = EXG2ANTTLib.ValueFormatEnum.exHTML
End With
```

The following C# sample hides the drop down button:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
EXG2ANTTLib.Editor editor = items.get_CellEditor(items.FirstVisibleItem, 0);
editor.EditType = EXG2ANTTLib.EditTypeEnum.DropDownListType ;
editor.AddItem(0, "0 - <b>No</b>", 1);
editor.AddItem(1, "1 - <b>Yes</b>", 1);
editor.DropDownVisible = false;
items.set_CellValue( items.FirstVisibleItem, 0, 1 );
items.set_CellValueFormat(items.FirstVisibleItem, 0,
EXG2ANTTLib.ValueFormatEnum.exHTML);
```

The following VFP sample hides the drop down button:

```
with thisform.G2antt1.Items
   .DefaultItem = .FirstVisibleItem
   With .CellEditor(0, 0)
      .EditType = 3 && DropDownListType
      .AddItem(0, "0 - <b>No</b>", 1)
      .AddItem(1, "1 - <b>Yes</b>", 2)
      .DropDownVisible = .f.
   EndWith
   .CellValue(0,0) = 1
   .CellValueFormat(0,0) = 1
endwith
```

property Editor.EditType as EditTypeEnum

Specifies the type of the column's editor.

Type	Description
EditTypeEnum	An EditTypeEnum expression that specifies the type of the editor.

Use the EditType property to set the type of the editor. By default, the editor's type is ReadOnly. If the control is bound to an ADO recordset the control looks for appropriate editor for each field. Each column has its own editor, that means that all cells of the column will use the column's editor when users edits a cell. The editor is visible only if the CellEditorVisible property is True. If the EditType is UserEditorType the UserEditor property should be called to initialize the user editor based on the ActiveX's identifier. Use the CellEditor property to specify a particular editor for a specific cell. Use the Option property to define options for a specific type of editor. Use the DefaultEditorOption property to specify default option for the editors of a specified type. Use the Locked property to lock an editor.

The following VB sample sets the column's editor to CheckListType:

```
Set c = .Columns.Add("Description")
With c.Editor
   .EditType = CheckListType
   .AddItem &H1000, "adFldCacheDeferred", 3
   .AddItem &H10, "adFldFixed", 3
   .AddItem &H40000, "adFldIsCollection", 3
   .AddItem &H20000, "adFldIsDefaultStream", 3
   .AddItem &H20, "adFldIsNullable", 3
   .AddItem &H10000, "adFldIsRowURL", 3
   .AddItem &H80, "adFldLong", 3
   .AddItem &H40, "adFldMayBeNull", 3
   .AddItem &H2, "adFldMayDefer", 3
   .AddItem &H4000, "adFldNegativeScale", 3
   .AddItem &H100, "adFldRowID", 3
   .AddItem &H200, "adFldRowVersion", 3
   .AddItem &H8, "adFldUnknownUpdatable", 3
   .AddItem &H4, "adFldUpdatable", 3
End With
```

In this case, the value of CellValue property for any cell in "Description" column should be an OR combination of values added using AddItem, InsertItem methods.

The following VB sample adds an EditType editor to the first visible item:

```
With G2antt1.Items
   With .CellEditor(.FirstVisibleItem, 0)
      .EditType = EXG2ANTTLibCtl.EditType
   End With
End With
```

The following C++ sample adds an EditType editor to the first visible item:

```
#include "Items.h"
#include "Editor.h"
COleVariant vtMissing; V_VT( &vtMissing) = VT_ERROR;
CItems items = m_g2antt.GetItems();
CEditor editor = items.GetCellEditor( COleVariant( items.GetFirstVisibleItem() ),
COleVariant( long(0) ) );
editor.SetEditType( 1 /*EditType*/ );
```

The following VB.NET sample adds an EditType editor to the first visible item:

```
With AxG2antt1.Items
   With .CellEditor(.FirstVisibleItem, 0)
      .EditType = EXG2ANTTLib.EditTypeEnum.EditType
   End With
End With
```

The following C# sample adds an EditType editor to the first visible item:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
EXG2ANTTLib.Editor editor = items.get_CellEditor(items.FirstVisibleItem, 0);
editor.EditType = EXG2ANTTLib.EditTypeEnum.EditType ;
```

The following VFP sample adds an EditType editor to the first visible item:

```
with thisform.G2antt1.Items
   With .CellEditor(.FirstVisibleItem, 0)
      .EditType = 1 && EdiType
```

```
    EndWith
endwith
```

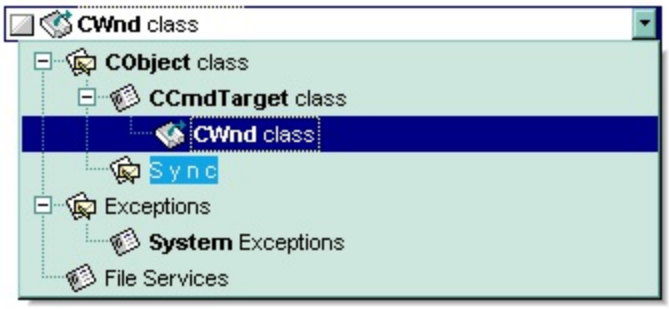
method Editor.ExpandAll ()

Expands all items in the editor's list.

Type	Description

By default, in your editor items that contain child items are collapsed. Use the ExpandAll method to expand all items in the editor. Use the InsertItem method to insert child items. Use the AddItem method to add predefined values to a drop down type editor. Use the ExpandItem method to expand a predefined value. Call the ExpandAll method after inserting the items. Use the SortItems method to sort the items in a drop down editor.

The following screen show shows a simple hierarchy into a built-in DropDownList editor:



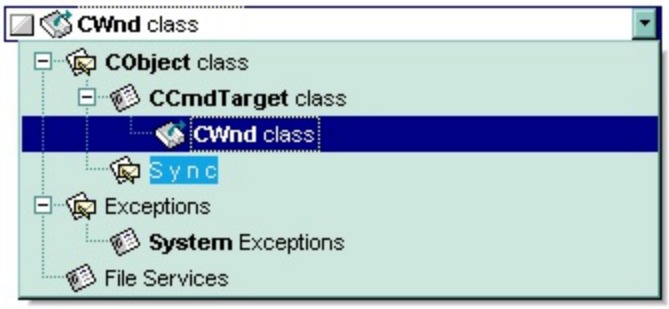
property Editor.ExpandItem(Value as Variant) as Boolean

Expandes or collapses an item in the editor's list.

Type	Description
Value as Variant	A long expression that indicates the value of the item being expanded, a string expression that indicates the caption of the item being expanded.
Boolean	A boolean expression that indicates whether the item is expanded or collapsed.

By default, the items in a drop down editor are collapsed. Use the ExpandItem to expand a specified item. Use the ExpandAll method to expand all items in the editor. Use the InsertItem method to insert a child item to your built-in editor. Use the AddItem method to add new predefined values. Use the AddButton method to add a button to the editor. Use the DropDownAutoWidth property on False, when inserting predefined values as child items.

The following screen show shows a simple hierarchy into a built-in DropDownList editor:



The following VB sample adds a simple hierarchy to a DropDownList built-in editor:

```
Dim h1 As HITEM
With G2antt1
   .BeginUpdate
   .Images
("gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjIgjIBAEijUlk8plUrlktl0vmExı

   .Images
("gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjIgjIBAEijUlk8plUrlktl0vmExı

   .Images
("gBJJgBggAAkGAAQhIAf8Nf4hhkOiRCJo2AEXjAAi0XFEYIEYhUXAIAEEZi8hk0plUrlktl0vmEx
```

```
   .Images
("gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjIgjIBAEijUlk8plUrlktl0vmExı

   .Columns.Add "Column 1"
   .ColumnAutoResize = True
   .HeaderVisible = False
   With .Items
      h1 = .InsertItem(, , "Child 1") ' Inserts a child itme
      .CellValueFormat(h1, 0) = exHTML
      .CellHasCheckBox(h1, 0) = True   ' Associates a check box to a cell
      .CellValue(h1, 0) = 3            ' Sets the cell's value
      .CellImage(h1, 0) = 1            ' Associates an image to a cell
      With .CellEditor(h1, 0)
         .EditType = DropDownListType
         .DropDownAutoWidth = False
         .AddItem 1, "CObject class", 1
         .AddItem 2, "CCmdTarget class", 2, 1
         .AddItem 3, "CWnd class", 3, 2
         .AddItem 6, "S y n c", 1, 1
         .AddItem 4, "Exceptions", 1
         .AddItem 7, "System Exceptions", 2, 4
         .AddItem 5, "File Services", 2
         .ExpandAll
      End With
   End With
   .EndUpdate
End With
```

The following VB samples adds a simple hierarchy to a PickEditType buil-in editor:

```
With G2antt1.Items
   With .CellEditor(.FirstVisibleItem, 0)
      .EditType = EXG2ANTTLibCtl.PickEditType
      .AddItem 0, "Organization"
      .InsertItem 1, "UN", , 0
      .InsertItem 2, "ONU", , 0
      .ExpandItem(0) = True
```

```
    End With
End With
```

The following C++ sample adds a simple hierarchy to a PickEditType built-in editor:

```
#include "Items.h"
#include "Editor.h"
COleVariant vtMissing; V_VT( &vtMissing) = VT_ERROR;
CItems items = m_g2antt.GetItems();
CEditor editor = items.GetCellEditor( COleVariant( items.GetFirstVisibleItem() ),
COleVariant( long(0) ) );
editor.SetEditType( 14 /*PickEditType*/ );
editor.AddItem( 0, "Organization", vtMissing );
editor.InsertItem( 1, "UN", vtMissing, COleVariant( long(0) ) );
editor.InsertItem( 2, "ONU", vtMissing, COleVariant( long(0) ) );
editor.SetExpandItem( COleVariant(long(0)), TRUE);
```

The following VB.NET sample adds a simple hierarchy to a DropDownListType built-in editor:

```
With AxG2antt1.Items
    With .CellEditor(.FirstVisibleItem, 0)
        .EditType = EXG2ANTTLib.EditTypeEnum.DropDownListType
        .AddItem(0, "Organization")
        .InsertItem(1, "UN", , 0)
        .InsertItem(2, "ONU", , 0)
        .ExpandItem(0) = True
    End With
End With
```

The following C# sample adds a simple hierarchy to a DropDownListType built-in editor:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
EXG2ANTTLib.Editor editor = items.get_CellEditor(items.FirstVisibleItem, 0);
editor.EditType = EXG2ANTTLib.EditTypeEnum.DropDownListType;
editor.AddItem(0, "Organization", null);
editor.InsertItem(1, "UN",null,  0);
editor.InsertItem(2, "ONU",null, 0);
editor.set_ExpandItem(0, true);
```

The following VFP sample adds a simple hierarchy to a DropDownType built-in editor:

```
with thisform.G2antt1.Items
   With .CellEditor(.FirstVisibleItem, 0)
      .EditType = 14 && PickEdiType
      .AddItem(0, "Organization")
      .InsertItem(1, "UN", , 0)
      .InsertItem(2, "ONU", , 0)
      .ExpandItem(0) = .t.
   EndWith
endwith
```

property Editor.FindItem (Value as Variant) as Variant

Finds an item given its value or caption.

Type	Description
Value as Variant	A long expression that indicates the value of the item being searched, a string expression that indicates the caption of the item being searched. If searching for a caption (string parameter), it can starts with ">" character, and so the searching is case-insensitive. By default the searching is case-sensitive. For instance, FindItem("One") looks for the exactly caption "One", while if using as FindItem(">One"), it searches case-insensitive for the word "one".
Variant	A string expression that indicates the caption of the item, if the Value is a long expression, a long expression that indicates the item's value if Value is a string expression.

The FindItem property retrieves an empty (VT_EMPTY) value if no item was found. Use the FindItem property to search the caption of the predefined value, in case the Value parameter is a long expression, or look for the predefined value when the Value parameter is a string expression. Use the AddItem method to add predefined values. Use the InsertItem method to add predefined values as child items.

In case you are using Items. CellEditor, the following sample finds the caption of selected value within a cell's editor. For instance, the NewValue can be the NewValue parameter of the Change event:

```
With G2antt1
  With .Items
    If (.HasCellEditor(Item, ColIndex)) Then
      Debug.Print ("Caption: " & .CellEditor(Item, ColIndex).FindItem(NewValue))
    End If
  End With
End With
```

In case you are using Column.Editor, the following sample finds the caption of selected value within a column's editor. For instance, the NewValue can be the NewValue parameter of the Change event:

```
With G2antt1
  With .Columns
```

```
        Debug.Print ("Caption: " & .Item(ColIndex).Editor.FindItem(NewValue))
    End With
End With
```

method Editor.InsertItem (Value as Long, Caption as String, [Image as Variant], [Parent as Variant])

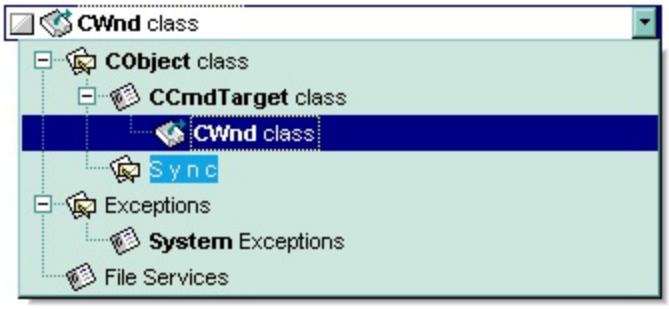
Inserts a child item to the editor's list.

Type	Description
Value as Long	A long expression that defines an unique predefined value
Caption as String	A string expression that specifies the HTML caption associated with the value. The format of the Caption parameter is "key|captionŚcaptionŚ...Ścaption", which indicates an item with the giving key / identifier, which displays multiple captions. The Caption allows using the following special characters: • | character (pipe, vertical bar, ALT + 126) defines the key or identifier of the item to add. Currently, the key is used by a **DropDownListType** editor to specify string codes rather numeric values for the cell's value (CellValue property) • Ś character (vertical broken bar, ALT + 221) defines captions for multiple columns. The Ś character can be escaped, so \Ś displays the Ś character. (available for DropDownType, DropDownListType and PickEditType editors, 20.0+) For instance: • "New York City" defines the "**New York** City" item • "NYC|New York City" the "**New York** City" item with the "NYC" as key or identifier • "NYC|New York CityŚ783.8 km²Ś8.42 mil" defines the "**New York** City" item with the "NYC" as key or identifier and sub-captions 783.8 km² and 8.42 mil (in separated columns) • "New York CityŚ783.8 km²Ś8.42 mil" defines the "**New York** City" item and sub-captions 783.8 km² and 8.42 mil (in separated columns)
Image as Variant	A long expression that indicates the index of the item's icon (1-based). Use the Images method to assign a list of icons to the control.

Parent as Variant	A long expression that defines the value of the parent item.

Use the InsertItem to insert child items to the editor's predefined list. Use the AddItem method to add new items to the editor's list. Use the ExpandItem property to expand an item. Use the ExpandAll items to expand all items. Use the ItemTooltip property to assign a tooltip to a predefined item into a drop down editor.

The following screen show shows a simple hierarchy into a built-in DropDownList editor:



The following VB sample adds a simple hierarchy to a DropDownList built-in editor:

```
Dim h1 As HITEM
With G2antt1
   .BeginUpdate
   .Images
("gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjIgjIBAEijUlk8plUrlktl0vmExı

   .Images
("gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjIgjIBAEijUlk8plUrlktl0vmExı

   .Images
("gBJJgBggAAkGAAQhIAf8Nf4hhkOiRCJo2AEXjAAi0XFEYIEYhUXAIAEEZi8hk0plUrlktl0vmEx

   .Images
("gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjIgjIBAEijUlk8plUrlktl0vmExı

   .Columns.Add "Column 1"
   .ColumnAutoResize = True
   .HeaderVisible = False
   With .Items
      h1 = .InsertItem(, , "Child 1") ' Inserts a child itme
      .CellValueFormat(h1, 0) = exHTML
```

```
    .CellHasCheckBox(h1, 0) = True   ' Associates a check box to a cell
    .CellValue(h1, 0) = 3            ' Sets the cell's value
    .CellImage(h1, 0) = 1            ' Associates an image to a cell
    With .CellEditor(h1, 0)
      .EditType = DropDownListType
      .DropDownAutoWidth = False
      .AddItem 1, "CObject class", 1
      .InsertItem 2, "CCmdTarget class", 2, 1
      .InsertItem 3, "CWnd class", 3, 2
      .InsertItem 6, "S y n c", 1, 1
      .AddItem 4, "Exceptions", 1
      .InsertItem 7, "System Exceptions", 2, 4
      .AddItem 5, "File Services", 2
      .ExpandAll
    End With
  End With
  .EndUpdate
End With
```

The following VB samples adds a simple hierarchy to a PickEditType buil-in editor:

```
With G2antt1.Items
  With .CellEditor(.FirstVisibleItem, 0)
    .EditType = EXG2ANTTLibCtl.PickEditType
    .AddItem 0, "Organization"
    .InsertItem 1, "UN", , 0
    .InsertItem 2, "ONU", , 0
    .ExpandItem(0) = True
  End With
End With
```

The following C++ sample adds a simple hierarchy to a PickEditType built-in editor:

```
#include "Items.h"
#include "Editor.h"
COleVariant vtMissing; V_VT( &vtMissing) = VT_ERROR;
CItems items = m_g2antt.GetItems();
CEditor editor = items.GetCellEditor( COleVariant( items.GetFirstVisibleItem() ),
```

```
COleVariant( long(0) ) );
editor.SetEditType( 14 /*PickEditType*/ );
editor.AddItem( 0, "Organization", vtMissing );
editor.InsertItem( 1, "UN", vtMissing, COleVariant( long(0) ) );
editor.InsertItem( 2, "ONU", vtMissing, COleVariant( long(0) ) );
editor.SetExpandItem( COleVariant(long(0)), TRUE);
```

The following VB.NET sample adds a simple hierarchy to a DropDownListType built-in editor:

```
With AxG2antt1.Items
   With .CellEditor(.FirstVisibleItem, 0)
      .EditType = EXG2ANTTLib.EditTypeEnum.DropDownListType
      .AddItem(0, "Organization")
      .InsertItem(1, "UN", , 0)
      .InsertItem(2, "ONU", , 0)
      .ExpandItem(0) = True
   End With
End With
```

The following C# sample adds a simple hierarchy to a DropDownListType built-in editor:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
EXG2ANTTLib.Editor editor = items.get_CellEditor(items.FirstVisibleItem, 0);
editor.EditType = EXG2ANTTLib.EditTypeEnum.DropDownListType;
editor.AddItem(0, "Organization", null);
editor.InsertItem(1, "UN",null,  0);
editor.InsertItem(2, "ONU",null, 0);
editor.set_ExpandItem(0, true);
```

The following VFP sample adds a simple hierarchy to a DropDownType built-in editor:

```
with thisform.G2antt1.Items
   With .CellEditor(.FirstVisibleItem, 0)
      .EditType = 14 && PickEdiType
      .AddItem(0, "Organization")
      .InsertItem(1, "UN", , 0)
      .InsertItem(2, "ONU", , 0)
      .ExpandItem(0) = .t.
```

```
    EndWith
endwith
```

property Editor.ItemToolTip(Value as Variant) as String

Gets or sets the text displayed when the mouse pointer hovers over a predefined item.

Type	Description
Value as Variant	A long expression that indicates the value of the item whose tooltip is accessed, a string expression that indicates the caption of the item whose tooltip is accessed.
String	A string expression that may include HTML tags, that indicates the text being displayed when the mouse hovers the item.

Use the ItemToolTip property to assign a tooltip for a drop down list value. Use the AddItem or InsertItem methods to insert new items to the drop down predefined list. The ItemToolTip property may include HTML tags that are listed here here.

The following VB sample adds a predefined value that displays a tooltip when the cursor hovers the value in the drop down portion of the editor:

```
With G2antt1.Columns(0).Editor
   .EditType = DropDownListType
   .AddItem 1, "Root Item"
   .ItemToolTip(1) = "This is a bit of text that should appear when the cursor
<b>hovers</b> the item."
   .InsertItem 2, "Child Item", , 1
End With
```

The following C++ sample adds a predefined value that displays a tooltip when the cursor hovers the value in the drop down portion of the editor:

```
#include "Items.h"
#include "Editor.h"
COleVariant vtMissing; V_VT( &vtMissing) = VT_ERROR;
CItems items = m_g2antt.GetItems();
CEditor editor = items.GetCellEditor( COleVariant( items.GetFirstVisibleItem() ),
COleVariant( long(0) ) );
editor.SetEditType( 3 /*DropDownListType*/ );
editor.AddItem( 0, "tooltip", vtMissing );
editor.SetItemToolTip( COleVariant( "tooltip" ), "This is a bit of text that should appear
```

```
when cursor hovers the item.");
```

The following VB.NET sample adds a predefined value that displays a tooltip when the cursor hovers the value in the drop down portion of the editor:

```
With AxG2antt1.Items
   With .CellEditor(.FirstVisibleItem, 0)
      .EditType = EXG2ANTTLib.EditTypeEnum.DropDownListType
      .AddItem(0, "tooltip")
      .ItemToolTip(0) = "This is a bit of text that should appear when cursor hovers the
item."
   End With
End With
```

The following C# sample adds a predefined value that displays a tooltip when the cursor hovers the value in the drop down portion of the editor:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
EXG2ANTTLib.Editor editor = items.get_CellEditor(items.FirstVisibleItem, 0);
editor.EditType = EXG2ANTTLib.EditTypeEnum.DropDownListType;
editor.AddItem(0, "tooltip", null);
editor.set_ItemToolTip(0, "This is a bit of text that should appear when cursor hovers the
item.");
```

The following VFP sample adds a predefined value that displays a tooltip when the cursor hovers the value in the drop down portion of the editor:

```
with thisform.G2antt1.Items
   With .CellEditor(.FirstVisibleItem, 0)
      .EditType = 3 && DropDownListType
      .AddItem(0, "tooltip")
      .ItemToolTip(0) = "This is a bit of text that should appear when cursor hovers the
item."
   EndWith
endwith
```

property Editor.Locked as Boolean

Determines whether the editor is locked or unlocked.

Type	Description
Boolean	A boolean expression that indicates whether the editor is locked or unlocked.

Use the Locked property to lock the editor. If the editor is locked, the user is not able to change the control's content using the editor. Use the CellEditorVisible property to hide the cell's editor. For instance, if the editor displays a drop down portion, even if locked, it is visible, but the user can't select new items to change the cell's value. Use the ReadOnly property to make the control read only. If the ReadOnly property is exLocked, all editors are locked. If the editor is locked, the user still can use the editor's buttons. The control fires the ButtonClick event when the user clicks a button. Use the Option(exEditLockedBackColor) and Option(exEditLockedForeColor) property to specify background and foreground colors while the edit control is locked.

property Editor.Mask as String

Retrieves or sets a value that indicates the mask used by the editor.

Type	Description
String	A string expression that defines the editor's mask.

Use the Mask property to filter characters during data input. Use the Mask property to control the entry of many types of formatted information such as telephone numbers, social security numbers, IP addresses, license keys etc. The Mask property has effect for the following edit types: DropDownType, SpinType, DateType, MaskType, FontType, PickEditType. The Numeric property specifies whether the editor enables numeric values only. Use the MaskChar property to change the masking character. The Mask property is composed by a combination of regular characters, literal escape characters, and placeholders, masking characters. The Mask property can contain also alternative characters, or range rules.

The following special characters are supported only in versions prior to version **12.1**

Here's the list of all rules and masking characters:

- **#** (Digit), Masks a digit character. [0-9]
- **x** (Hexa Lower), Masks a lower hexa character. [0-9],[a-f]
- **X** (Hexa Upper), Masks a upper hexa character. [0-9],[A-F]
- **A** (AlphaNumeric), Masks a letter or a digit. [0-9], [a-z], [A-Z]
- **?** (Alphabetic), Masks a letter. [a-z],[A-Z]
- **<** (Alphabetic Lower), Masks a lower letter. [a-z]
- **>** (Alphabetic Upper), Masks an upper letter. [A-Z]
- ***** (Any), Mask any combination of characters.
- **** (Literal Escape), Displays any masking characters. The following combinations are valid: \#,\x,\X,\A,\?,\<,\>,\\,\{,\[
- **{nMin,nMax}** (Range), Masks a number in a range. The nMin and nMax values should be numbers. For instance the mask {0,255} will mask any number between 0 and 255.
- **[...]** (Alternative), Masks any characters that are contained by brackets []. For instance, the [abcA-C] mask any character: a,b,c,A,B,C

The following VB sample adds a mask for IP addresses:

```
With G2antt1
```

```
  With .Columns.Add("Mask")
    With .Editor
      .EditType = EditTypeEnum.MaskType
      .Mask = "{0,255}\.{0,255}\.{0,255}\.{0,255}"
    End With
  End With
  .Items.AddItem "193.226.40.161"
End With
```

The following VB sample masks a phone number:

```
With G2antt1
  With .Columns.Add("Mask")
    With .Editor
      .EditType = EditTypeEnum.MaskType
      .Mask = "(XXX) - XXX XXXX"
    End With
  End With
  .Items.AddItem "(095) - 889 1234"
End With
```

The following C++ adds a mask editor to filter characters while entering a phone number:

```
#include "Items.h"
#include "Editor.h"
COleVariant vtMissing; V_VT( &vtMissing) = VT_ERROR;
CItems items = m_grid.GetItems();
CEditor editor = items.GetCellEditor( COleVariant( items.GetFirstVisibleItem() ),
COleVariant( long(0) ) );
editor.SetEditType( 8 /*MaskType*/ );
editor.SetMask("(###) ### - ####");
```

The following VB.NET adds a mask editor to filter characters while entering a phone number:

```
With AxG2antt1.Items
  With .CellEditor(.FirstVisibleItem, 0)
    .EditType = EXGRIDLib.EditTypeEnum.MaskType
    .Mask = "(###) ### - ####"
```

```
    End With
End With
```

The following C# adds a mask editor to filter characters while entering a phone number:

```
EXGRIDLib.Items items = axG2antt1.Items;
EXGRIDLib.Editor editor = items.get_CellEditor(items.FirstVisibleItem, 0);
editor.EditType = EXGRIDLib.EditTypeEnum.MaskType;
editor.Mask = "(###) ### - ####";
```

The following VFP adds a mask editor to filter characters while entering a phone number:

```
with thisform.G2antt1.Items
    With .CellEditor(.FirstVisibleItem, 0)
        .EditType = 8 && MaskType
        .Mask = "(###) ### - ####"
    EndWith
endwith
```

Starting from the version **12.1**, the Mask property is changed radically, to support more special characters, validation, float numbers, and so on.

For instance, the following input-mask (ext-phone)

"*!(999) 000 0000;1;;select=1,empty,overtype,warning=invalid character,invalid=The value you entered isn't appropriate for the input mask '<%mask%>' specified for this field.*"

indicates the following:

- The pattern should contain 3 optional digits *999*, and 7 required digits *000 0000*, aligned to the right, *!*.
- The second part of the input mask indicates *1*, which means that all literals are included when the user leaves the field.
- The entire field is selected when it receives the focus, *select=1*
- The field supports *empty* value, so the user can leave the field with no content
- The field enters in *overtype* mode, and insert-type mode is not allowed when user pressed the Insert key
- If the user enters any invalid character, a *warning* tooltip with the message "*invalid character*" is displayed.
- If the user tries to leave the field, while the field is not validated (all 7 required digits completed), the *invalid* tooltip is shown with the message "*The value you entered isn't*

appropriate for the input mask '<%mask%>' specified for this field." The *<%mask%>* is replaced with the first part of the input mask *!(999) 000 0000*

The four parts of an input mask, or the Mask property supports up to four parts, separated by a semicolon (;). For instance, "`Time: `00:00:00;;0;overtype,warning=<fgcolor FF0000>invalid character,beep", indicates the pattern "00:00" with the prefix Time:, the masking character being the 0, instead _, the field enters in over-type mode, insert-type mode is not allowed, and the field beeps and displays a tooltip in red with the message invalid character when the user enters an invalid character.

Input masks are made up one mandatory part and three optional parts, and each part is separated by a semicolon (;). If a part should use the semicolon (;) it must uses the \; instead

The purpose of each part is as follows:

1. The first part (pattern) is mandatory. It includes the mask characters or string (series of characters) along with placeholders and literal data such as, parentheses, periods, and hyphens.

   The following table lists the placeholder and literal characters for an input mask and explains how it controls data entry:

   - ***#**, a digit, +, - or space (entry not required).*
   - ***0**, a digit (0 through 9, entry required; plus [+] and minus [-] signs not allowed).*
   - ***9**, a digit or space (entry not required; plus and minus signs not allowed).*
   - ***x**, a lower case hexa character, [0-9],[a-f] (entry required)*
   - ***X**, an upper case hexa character, [0-9],[A-F] (entry required)*
   - ***A**, any letter, digit (entry required).*
   - ***a**, any letter, digit or space (entry optional).*
   - ***L**, any letter (entry require).*
   - ***?**, any letter or space (entry optional).*
   - ***&**, any character or a space (entry required).*
   - ***C**, any character or a space (entry optional).*
   - ***>**, any letter, converted to uppercase (entry required).*
   - ***<**, any letter, converted to lowercase (entry required).*
   - **, any characters combinations*
   - ***{ min,max }** (Range), indicates a number range. The syntax {min,max} (Range), masks a number in the giving range. The min and max values should be positive integers. For instance the mask {0,255} masks any number between 0 and 255.*
   - ***[...]** (Alternative), masks any characters that are contained in the [] brackets. For*

instance, the [abcdA-D] mask any character: a,b,c,d,A,B,C,D

- ****, *indicates the escape character*
- ***t'**, (ALT + 175) causes the characters that follow to be converted to uppercase, until Ť(ALT + 174) is found.*
- ***Ť**, (ALT + 174) causes the characters that follow to be converted to lowercase, until t'(ALT + 175) is found.*
- ***!**, causes the input mask to fill from right to left instead of from left to right.*

Characters enclosed in double quotation ("" or ``) marks will be displayed literally. If this part should display/use the semicolon (;) character is should be included between double quotation ("" or ``) characters or as \; (escape).

2. The second part is optional and refers to the embedded mask characters and how they are stored within the field. If the second part is set to 0 (default, exClipModeLiteralsNone), all characters are stored with the data, and if it is set to 1 (exClipModeLiteralsInclude), the literals are stored, not including the masking/placeholder characters, if 2 (exClipModeLiteralsExclude), just typed characters are stored, if 3(exClipModeLiteralsEscape), optional, required, editable and escaped entities are included. No double quoted text is included.
3. The third part of the input mask is also optional and indicates a single character or space that is used as a placeholder. By default, the field uses the underscore (_). If you want to use another character, enter it in the third part of your mask. Only the first character is considered. If this part should display/use the semicolon (;) character is should be \; (escape) (MaskChar property)
4. The forth part of the input, indicates a list of options that can be applied to input mask, separated by comma(,) character.

   The known options for the forth part are:

   - ***float**, indicates that the field is edited as a decimal number, integer. The first part of the input mask specifies the pattern to be used for grouping and decimal separators, and - if negative numbers are supported. If the first part is empty, the float is formatted as indicated by current regional settings. For instance, "##;;;float" specifies a 2 digit number in float format. The grouping, decimal, negative and digits options are valid if the float option is present.*
   - ***grouping**=value, Character used to separate groups of digits to the left of the decimal. Valid only if float is present. For instance ";;;float,grouping=" indicates that no grouping is applied to the decimal number (LOCALE_STHOUSAND)*
   - ***decimal**=value, Character used for the decimal separator. Valid only if float is*

present. For instance ";;;float,grouping= ,decimal=\," indicates that the decimal number uses the space for grouping digits to the left, while for decimal separator the comma character is used (LOCALE_SDECIMAL)

- ***negative**=value, indicates whether the decimal number supports negative numbers. The value should be 0 or 1. 1 means negative numbers are allowed. Else 0 or missing, the negative numbers are not accepted. Valid only if float is present.*
- ***digits**=value, indicates the max number of fractional digits placed after the decimal separator. Valid only if float is present. For instance, ";;;float,digits=4" indicates a max 4 digits after decimal separator (LOCALE_IDIGITS)*
- ***password**[=value], displays a black circle for any shown character. For instance, ";;;password", specifies that the field to be displayed as a password. If the value parameter is present, the first character in the value indicates the password character to be used. By default, the * password character is used for non-TrueType fonts, else the black circle character is used. For instance, ";;;password=*", specifies that the field to be displayed as a password, and use the * for password character. If the value parameter is missing, the default password character is used.*
- ***right**, aligns the characters to the right. For instance, "(999) 999-9999;;;right" displays and masks a telephone number aligned to the right*
- ***readonly**, the editor is locked, user can not update the content, the caret is available, so user can copy the text, excepts the password fields.*
- ***inserttype**, indicates that the field enters in insert-type mode, if this is the first option found. If the forth part includes also the overtype option, it indicates that the user can toggle the insert/over-type mode using the Insert key. For instance, the "##:##;;0;inserttype,overtype", indicates that the field enter in insert-type mode, and over-type mode is allowed. The "##:##;;0;inserttype", indicates that the field enter in insert-type mode, and over-type mode is not allowed.*
- ***overtype**, indicates that the field enters in over-type mode, if this is the first option found. If the forth part includes also the inserttype option, it indicates that the user can toggle the insert/over-type mode using the Insert key. For instance, the "##:##;;0;overtype,inserttype", indicates that the field enter in over-type mode, and insert-type mode is allowed. The "##:##;;0;overtype", indicates that the field enter in over-type mode, and insert-type mode is not allowed.*
- ***nocontext**, indicates that the field provides no context menu when user right clicks the field. For instance, ";;;password,nocontext" displays a password field, where the user can not invoke the default context menu, usually when a right click occurs.*
- ***beep**, indicates whether a beep is played once the user enters an invalid character. For instance, "00:00;;;beep" plays a beep once the user types in invalid character, in this case any character that's not a digit.*

- ***warning**=value, indicates the html message to be shown when the user enters an invalid character. For instance, "00:00:00;;;warning=invalid character" displays a "invalid character" tooltip once the user types in invalid character, in this case any character that's not a digit. The <%mask%> keyword in value, substitute the current mask of the field, while the <%value%> keyword substitutes the current value (including the literals). If this option should display/use the semicolon (;) character is should be \; (escape)*
- ***invalid**=value, indicates the html message to be displayed when the user enters an inappropriate value for the field. If the value is missing or empty, the option has no effect, so no validation is performed. If the value is a not-empty value, the validation is performed. If the value is single space, no message is displayed and the field is keep opened while the value is inappropriate. For instance, "!(999) 000 0000;;;invalid=The value you entered isn't appropriate for the input mask '<%mask%>' specified for this field." displays the "The value you entered isn't appropriate for the input mask '...' specified for this field." tooltip once the user leaves the field and it is not-valid (for instance, the field includes entities required and uncompleted). The <%mask%> keyword in value, substitute the current mask of the field, while the <%value%> keyword substitutes the current value (including the literals). If this option should display/use the semicolon (;) character is should be \; (escape). This option can be combined with empty, validateas. The invalid option should be used, with the CauseValidateValue property on True, so the user can not leaves the field while it contains an invalid value.*
- ***validateas**=value, specifies the additional validation is done for the current field. If value is missing or 0 (exValidateAsNone), the option has no effect. The validateas option has effect only if the invalid option specifies a not-empty value. Currently, the value can be 1 (exValidateAsDate), which indicates that the field is validated as a date. For instance, having the mask "!00/00/0000;;0;empty,validateas=1,invalid=Invalid date!,warning=Invalid character!,select=4,overtype", indicates that the field is validate as date (validateas=1).*
- ***empty**, indicates whether the field supports empty values. This option can be used with invalid flag, which indicates that the user can leave the field if it is empty. If empty flag is present, the field displays nothing if no entity is completed (empty). Once the user starts typing characters the current mask is displayed. For instance, having the mask "!(999) 000 0000;;;empty,select=4,overtype,invalid=invalid phone number,beep", it specifies an empty or valid phone to be entered.*
- ***select**=value, indicates what to select from the field when it got the focus. The value could be 0 (nothing, exSelectNoGotFocus), 1 (select all, exSelectAllGotFocus), 2 (select the first empty and editable entity of the field,*

exSelectEditableGotFocus), 3 (moves the cursor to the beginning of the first empty and editable entity of the field, exMoveEditableGotFocus), 4 (select the first empty, required and editable entity of the field, exSelectRequiredEditableGotFocus), 5 (moves the cursor to the beginning of the first empty, required and editable entity of the field, exMoveRequiredEditableGotFocus). For modes 2 and 4 the entire field is selected if no matching entity is found. For instance, "`Time:`XX:XX;;;select=1" indicates that the entire field (including the Time: prefix) is selected once it get the focus. The "`Time:`XX:XX;;;select=3", moves the cursor to first X, if empty, the second if empty, and so on

- ***leading**=value, specifies whether the spaces or masking/placeholder (0,9) characters are replaced with giving value (0 if the value is missing). This option has effect, only for DateType fields (Editor.EditType property is DateType), when the field is entering in edit mode, or the user selects a new date from the drop down calendar. For instance, "!99/99/9999;1;;empty,validateas=1,invalid=Invalid date\, for the input mask
'<%mask%>'!,warning=Invalid character!,select=4,overtype,leading", having the cell's value on #1/1/2001# it displays 01/01/2001, instead 1/1/2001.*

Experimental:
***multiline**, specifies that the field supports multiple lines.*
***rich**, specifies that the field displays a rich type editor. By default, the standard edit field is shown*
***disabled**, shows as disabled the field.*

property Editor.MaskChar as Long

Retrieves or sets a value that indicates the character used for masking.

Type	Description
Long	A long expression that indicates the ASCII code for the masking character.

Use the MaskChar property to change the default masking character, which is '_'. The MaskChar property has effect only if the Mask property is not empty, and the mask is applicable to the editor's type.

property Editor.Numeric as NumericEnum

Specifies whether the editor enables numeric values only.

Type	Description
NumericEnum	A NumericEnum expression that indicates whether integer or floating point numbers are allowed.

The Numeric property has effect only if the editor contains an edit box. Use the Numeric property to add intelligent input filtering for integer, or floating points numbers. Use the exSpinStep option to specify the proposed change when the user clicks a spin control, if the cell's editor is of SpinType type. Use the exEditDecimaSymbol option to specify the symbol being used by decimal value while editing a floating point number.

property Editor.Option(Name as EditorOptionEnum) as Variant

Specifies an option for the editor.

Type	Description
Name as EditorOptionEnum	An EditorOptionEnum expression that indicates the editor's option being changed.
Variant	A Variant expression that indicates the value for editor's option

The Option property of Editor object provides the ability to add scroll bars to a memo editor using the exMemoHScrollBar and exMemoVScrollBar options. Use the DefaultEditorOption property to specify default option for the editors of a specified type.

For instance, the following VB sample adds both scroll bar to the editor of the first column:

```
With G2antt1.Columns(0).Editor
      .Option(exMemoAutoSize) = False     ' Disables auto resizing when user alters the text
      .Option(exMemoVScrollBar) = True    ' Adds the vertical scroll bar
      .Option(exMemoHScrollBar) = True    ' Adds the horizontal scroll bar
End With
```

The following VB sample adds a cell with a password editor:

```
With G2antt1.Items
  Dim h As HITEM
  h = .InsertItem(, , "password")
  With .CellEditor(h, 0)
    .EditType = EXG2ANTTLibCtl.EditType
     .Option(EXG2ANTTLibCtl.EditorOptionEnum.exEditPassword) = True
  End With
End With
```

The following VB sample indicates how to let user uses the left, right arrows, home and end keys to move the cursor inside an editor that displays a caret, instead changing the focused cell:

```
With G2antt1.Columns(ColIndex).Editor
      .Option(exLeftArrow) = exHandleEditor
      .Option(exRightArrow) = exHandleEditor
```

```
    .Option(exHomeKey) = exHandleEditor
    .Option(exEndKey) = exHandleEditor
End With
```

The following C++ sample adds a password editor:

```
#include "Items.h"
#include "Editor.h"
COleVariant vtMissing; V_VT( &vtMissing) = VT_ERROR;
CItems items = m_g2antt.GetItems();
CEditor editor = items.GetCellEditor( COleVariant( items.GetFirstVisibleItem() ),
COleVariant( long(0) ) );
editor.SetEditType( 1 /*EditType*/ );
editor.SetOption( 18 /*exEditPassword*/, COleVariant( VARIANT_TRUE ) );
```

The following VB.NET sample adds a password editor:

```
With AxG2antt1.Items
  With .CellEditor(.FirstVisibleItem, 0)
    .EditType = EXG2ANTTLib.EditTypeEnum.EditType
    .Option(EXG2ANTTLib.EditorOptionEnum.exEditPassword) = True
  End With
End With
```

The following C# sample adds a password editor:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
EXG2ANTTLib.Editor editor = items.get_CellEditor(items.FirstVisibleItem, 0);
editor.EditType = EXG2ANTTLib.EditTypeEnum.EditType;
editor.set_Option(EXG2ANTTLib.EditorOptionEnum.exEditPassword, true );
```

The following VFP sample adds a password editor:

```
with thisform.G2antt1.Items
  With .CellEditor(.FirstVisibleItem, 0)
    .EditType = 1 && EditType
    .Option(18) = .t. && exEditPassword
  EndWith
endwith
```

property Editor.PartialCheck as Boolean

Retrieves or sets a value that indicates whether the associated check box has two or three states.

Type	Description
Boolean	A boolean expression that indicates whether the associated check box has two or three states.

Use the PartialCheck property to allow three-states check boxes into the editor. Use the CellHasCheckBox property to define a check box for the cell. Use the CellState property to specify the cell's state. Use the PartialCheck property to allow partial check feature in a column.

property Editor.PopupAppearance as InplaceAppearanceEnum

Retrieves or sets a value that controls the drop-down window's appearance.

Type	Description
InplaceAppearanceEnum	An InplaceAppearanceEnum expression that defines the drop-down window's border style.

By default, the PopupAppearance property is ShadowApp. Use the PopupAppearance property to change the drop-down window's border style. Use the Appearance property to define the editor's appearance.

method Editor.RemoveButton (Key as Variant)

Removes a button given its key.

Type	Description
Key as Variant	A Variant value that determines the button's key being deleted. The Key should be the same as used in the AddButton method.

Use the RemoveButton method to remove a button, given its key. Use the ButtonWidth property to hide the editor buttons. Use the ClearButtons method to remove all buttons. Use the DropDownVisible property to hide the drop down button. You can remove only buttons added using the AddButton method.

method Editor.RemoveItem (Value as Long)

Removes an item from the editor's predefined values list.

Type	Description
Value as Long	A long expression that indicates the index of the item being removed, or a string expression that indicates the caption of the item being removed.

Use the RemoveItem method to remove an item from the editor's predefined values list. Use the ClearItems method to clear the entire list of editor items. Use the DropDownVisible property to hide the drop down button. You can remove only items that were added using AddItem or InsertItem method. Use the FindItem property to look for a predefined value in the drop down list.

method Editor.SortItems ([Ascending as Variant], [Reserved as Variant])

Sorts the list of items in the editor.

Type	Description
Ascending as Variant	A boolean expression that indicates the sort order of the items. By default, is the Ascending parameter is True, if it is missing.
Reserved as Variant	Not used. For future use only.

Use the SortItems method to sort the items in a drop down editor. Use the ExpandAll method to expand all items. Call the SortItems method after adding predefined values to the drop down list. Use the AddItem or InsertItem method to add predefined values to the drop down list. Use the SortOrder property to sort a column.

method Editor.UserEditor (ControlID as String, License as String)

Specifies the control's identifier and the control's runtime license key when EditType is UserEditor.

Type	Description
ControlID as String	A string expression that indicates the control's program identifier. For instance, if you want to use a multiple column combobox as an user editor, the control's identifier could be: "Exontrol.ComboBox".
License as String	Optional. A string expression that indicates the runtime license key in case is it required. It depends on what control are you using.

The UserEditor property creates a new type of editor based on the ControlID parameter. The EditType property has effect only if it is UserEditorType. Use the UserEditorObject property to access the newly created object. The UserEditorObject property is nothing if the control wasn't able to create the user editor based on the ControlID. Also, if the user control requires a runtime license key, and the License parameter is empty or doesn't match, the UserEditorObject property is nothing. The control fires the UserEditorOpen event when a ActiveX editor is about to be shown. The control fires the UserEditorClose event when the user editor is hidden. The control fires the UserEditorOleEvent event each time when an user editor fires an event. The setup installs the VB\UserEditor, VC\User.Edit sample that uses the Exontrol ExComboBox Component as a new editor.

The following VB sample adds a new column to an editor of of Exontrol.ComboBox type (exComboBox component):

```
With G2antt1
  .BeginUpdate
  With .Columns
    With .Add("Column 0").Editor
      .EditType = EditTypeEnum.UserEditorType
      ' Creates an ExComboBox control, and gets the object created
      .UserEditor "Exontrol.ComboBox", ""
       If Not .UserEditorObject Is Nothing Then
         With .UserEditorObject ' Points to an ExComboBox control
           ' The ExComboBox object
           .BeginUpdate
           .EndUpdate
         End With
```

```
        End If
      End With
    End With
    .EndUpdate
End With
```

The following VB sample adds the Exontrol's eXMaskEdit Component to mask floating point numbers with digit grouping:

```
With G2antt1.Items
    Dim h As HITEM
    h = .AddItem(100)
    With .CellEditor(h, 0)
      .EditType = UserEditorType
      .UserEditor "Exontrol.MaskEdit", ""
      With .UserEditorObject()
        .BackColor = vbWhite
        .MaskFloat = True
        .Mask = "-###.###.###,##"
      End With
    End With
End With
```

The following C++ sample adds the Exontrol's eXMaskEdit Component to mask floating point numbers with digit grouping:

```
CItems items = m_g2antt.GetItems();
long hItem = items.AddItem( COleVariant( (double)100 ) );
CEditor editor = items.GetCellEditor( COleVariant( hItem ), COleVariant( long(0) ) );
editor.SetEditType( 16 /*UserEditorType*/ );
editor.UserEditor( "Exontrol.MaskEdit", "" );
MaskEditLib::IMaskEditPtr spMaskEdit( editor.GetUserEditorObject() );
if ( spMaskEdit != NULL )
{
    spMaskEdit->put_MaskFloat( TRUE );
    spMaskEdit->put_Mask( L"-###.###.###,##" );
    spMaskEdit->put_BackColor( RGB(255,255,255) );
}
```

The sample requires calling the #import <maskedit.dll> to include the type library for the eXMaskEdit component. The #import <maskedit.dll> defines the MaskEditLib namespace used in the sample.

The following VB.NET sample adds the Exontrol's eXMaskEdit Component to mask floating point numbers with digit grouping:

```
With AxG2antt1.Items
   Dim h As Integer = .AddItem(1000)
   With .CellEditor(h, 0)
      .EditType = EXG2ANTTLib.EditTypeEnum.UserEditorType
      .UserEditor("Exontrol.MaskEdit", "")
      With .UserEditorObject()
         .BackColor = ToUInt32(Color.White)
         .MaskFloat = True
         .Mask = "-###.###.###,##"
      End With
   End With
End With
```

where the ToUInt32 function converts a Color expression to an unsigned long expression and may look like follows:

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
   Dim i As Long
   i = c.R
   i = i + 256 * c.G
   i = i + 256 * 256 * c.B
   ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample adds the Exontrol's eXMaskEdit Component to mask floating point numbers with digit grouping:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
int hItem = items.AddItem(100);
EXG2ANTTLib.Editor editor = items.get_CellEditor(hItem, 0);
editor.EditType = EXG2ANTTLib.EditTypeEnum.UserEditorType;
editor.UserEditor("Exontrol.MaskEdit", "");
```

```
MaskEditLib.MaskEdit maskEdit = editor.UserEditorObject as MaskEditLib.MaskEdit;
if (maskEdit != null)
{
   maskEdit.BackColor = ToUInt32(Color.White);
   maskEdit.MaskFloat = true;
   maskEdit.Mask = "-###.###.###,##";
}
```

where the MaskEditLib class is defined by adding a new reference to the ExMaskEdit component to your project. The ToUInt32 function converts a Color expression to an OLE_COLOR expression (unsigned long expression), and may look like follows:

```
private UInt32 ToUInt32(Color c)
{
   long i;
   i = c.R;
   i = i + 256 * c.G;
   i = i + 256 * 256 * c.B;
   return Convert.ToUInt32(i);
}
```

The following VFP sample adds the Exontrol's eXMaskEdit Component to mask floating point numbers with digit grouping:

```
With thisform.G2antt1.Items
   local h
   h = .AddItem(100)
   With .CellEditor(h, 0)
      .EditType = 16 && UserEditorType
      .UserEditor("Exontrol.MaskEdit", "")
      With .UserEditorObject()
         .BackColor = RGB(255,255,255)
         .MaskFloat = .t.
         .Mask = "-###.###.###,##"
      EndWith
   EndWith
EndWith
```

property Editor.UserEditorObject as Object

Gets the user editor object when EditType is UserEditorType.

Type	Description
Object	An ActiveX object being used as an user editor.

Use the UserEditorOpen property to access to the ActiveX user editor. Use the UserEditor property to initialize the ActiveX user editor. The UserEditorObject property retrieves the ActiveX control created when UserEditor method was invoked. The type of object returned by the UserEditorObject depends on the ControlID parameter of the UserEditor method. For instance, the type of the created object when UserEditor("Exontrol.ComboBox") is used, is EXCOMBOBOXLibCtl.ComboBox. The UserEditorObject property gets nothing if the UserEditor method fails. The control fires the UserEditorOpen event when an user editor is about to be shown. The control fires the UserEditorClose event when the control closes an user editor. The control fires the UserEditorOleEvent event each time when an user editor fires an event.

The following VB sample initializes an user editor of EXCOMBOBOXLibCtl.ComboBox:

```
With G2antt1
   .BeginUpdate
   .DefaultItemHeight = 21
   .TreeColumnIndex = -1
   .ColumnAutoResize = True
   .MarkSearchColumn = False
   .FullRowSelect = False
   .DrawGridLines = exVLines
   With .Columns
      With .Add("Column 0").Editor
      .EditType = EditTypeEnum.UserEditorType
      ' Creates an ExComboBox control, and gets the object created
      .UserEditor "Exontrol.ComboBox", ""
      If Not .UserEditorObject Is Nothing Then
         With .UserEditorObject ' Points to an ExComboBox control
            ' Loads the ExComboBox object
            .BeginUpdate
            .LinesAtRoot = exGroupLinesAtRoot
            .ColumnAutoResize = True
            .IntegralHeight = True
```

```
            .HeaderVisible = False
            .AllowSizeGrip = True
            .MinHeightList = 100
            .AutoDropDown = True
            .HasButtons = exArrow
            .Indent = 18
            .MarkSearchColumn = False
            .BackColor = G2antt1.BackColor
            With .Columns
               With .Add("Column 0")
               End With
               With .Add("Column 1")
                  .Position = 0
                  .Width = 16
               End With
            End With
            With .Items
               Dim h1 As HITEM, h2 As HITEM, h12 As HITEM, i As Long
               For i = 1 To 3
                  h1 = .AddItem("Group " & i)
                  .CellValue(h1, 1) = i * 4 - 3
                  h12 = .InsertItem(h1, , "Item 1")
                  .CellValue(h12, 1) = i * 4 - 2
                  h12 = .InsertItem(h1, , "Item 2")
                  .CellValue(h12, 1) = i * 4 - 1
                  h12 = .InsertItem(h1, , "Item 3")
                  .CellValue(h12, 1) = i * 4
                  .ExpandItem(h1) = True
               Next
            End With
            .EndUpdate
         End With
   Else
      MsgBox "YOU NEED TO HAVE INSTALLED THE EXCOMBOBOX CONTROL, else you will not be able to see the UserEditor column"
   End If
End With
```

```
With .Add("Column 1")
   With .Editor
      .EditType = EditTypeEnum.DateType
      .AddItem 10, "Ten"
      .AddItem 20, "Twenty"
    End With
End With
End With
For i = 1 To 11
   With .Items
      Dim h As HITEM
      h = .AddItem(i)
   End With
Next
   .EndUpdate
End With
```

The following VB sample adds the Exontrol's eXMaskEdit Component to mask floating point numbers with digit grouping:

```
With G2antt1.Items
   Dim h As HITEM
   h = .AddItem(100)
   With .CellEditor(h, 0)
      .EditType = UserEditorType
      .UserEditor "Exontrol.MaskEdit", ""
      With .UserEditorObject()
         .BackColor = vbWhite
         .MaskFloat = True
         .Mask = "-###.###.###,##"
      End With
   End With
End With
```

The following C++ sample adds the Exontrol's eXMaskEdit Component to mask floating point numbers with digit grouping:

```
CItems items = m_g2antt.GetItems();
```

```
long hItem = items.AddItem( COleVariant( (double)100 ) );
CEditor editor = items.GetCellEditor( COleVariant( hItem ), COleVariant( long(0) ) );
editor.SetEditType( 16 /*UserEditorType*/ );
editor.UserEditor( "Exontrol.MaskEdit", "" );
MaskEditLib::IMaskEditPtr spMaskEdit( editor.GetUserEditorObject() );
if ( spMaskEdit != NULL )
{
   spMaskEdit->put_MaskFloat( TRUE );
   spMaskEdit->put_Mask( L"-###.###.###,##" );
   spMaskEdit->put_BackColor( RGB(255,255,255) );
}
```

The sample requires calling the #import <maskedit.dll> to include the type library for the eXMaskEdit component. The #import <maskedit.dll> defines the MaskEditLib namespace used in the sample.

The following VB.NET sample adds the Exontrol's eXMaskEdit Component to mask floating point numbers with digit grouping:

```
With AxG2antt1.Items
   Dim h As Integer = .AddItem(1000)
   With .CellEditor(h, 0)
      .EditType = EXG2ANTTLib.EditTypeEnum.UserEditorType
      .UserEditor("Exontrol.MaskEdit", "")
      With .UserEditorObject()
         .BackColor = ToUInt32(Color.White)
         .MaskFloat = True
         .Mask = "-###.###.###,##"
      End With
   End With
End With
```

where the ToUInt32 function converts a Color expression to an unsigned long expression and may look like follows:

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
   Dim i As Long
   i = c.R
   i = i + 256 * c.G
```

```
    i = i + 256 * 256 * c.B
    ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample adds the Exontrol's eXMaskEdit Component to mask floating point numbers with digit grouping:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
int hItem = items.AddItem(100);
EXG2ANTTLib.Editor editor = items.get_CellEditor(hItem, 0);
editor.EditType = EXG2ANTTLib.EditTypeEnum.UserEditorType;
editor.UserEditor("Exontrol.MaskEdit", "");
MaskEditLib.MaskEdit maskEdit = editor.UserEditorObject as MaskEditLib.MaskEdit;
if (maskEdit != null)
{
    maskEdit.BackColor = ToUInt32(Color.White);
    maskEdit.MaskFloat = true;
    maskEdit.Mask = "-###.###.###,##";
}
```

where the MaskEditLib class is defined by adding a new reference to the ExMaskEdit component to your project. The ToUInt32 function converts a Color expression to an OLE_COLOR expression (unsigned long expression), and may look like follows:

```
private UInt32 ToUInt32(Color c)
{
    long i;
    i = c.R;
    i = i + 256 * c.G;
    i = i + 256 * 256 * c.B;
    return Convert.ToUInt32(i);
}
```

The following VFP sample adds the Exontrol's eXMaskEdit Component to mask floating point numbers with digit grouping:

```
With thisform.G2antt1.Items
    local h
    h = .AddItem(100)
```

```
    With .CellEditor(h, 0)
        .EditType = 16 && UserEditorType
        .UserEditor("Exontrol.MaskEdit", "")
        With .UserEditorObject()
            .BackColor = RGB(255,255,255)
            .MaskFloat = .t.
            .Mask = "-###.###.###,##"
        EndWith
    EndWith
EndWith
```

ExDataObject object

The OleDragDrop event notifies your application that the user drags some data on the control. Defines the object that contains OLE drag and drop information. The ExDataObject object supports the following method and properties:

Name	Description
Clear	Deletes the contents of the ExDataObject object.
Files	Returns an ExDataObjectFiles collection, which in turn contains a list of all filenames used by an ExDataObject object.
GetData	Returns data from an ExDataObject object in the form of a variant.
GetFormat	Returns a value indicating whether an item in the ExDataObject object matches a specified format.
SetData	Inserts data into an ExDataObject object using the specified data format.

method ExDataObject.Clear ()

Deletes the contents of the DataObject object.

Type	Description

The Clear method can be called only for drag sources. The OleDragDrop event notifies your application that the user drags some data on the control.

property ExDataObject.Files as ExDataObjectFiles

Returns a DataObjectFiles collection, which in turn contains a list of all filenames used by a DataObject object.

Type	Description
ExDataObjectFiles	An ExDataObjectFiles object that contains a list of filenames used in OLE drag and drop operations.

The Files property is valid only if the format of the clipboard data is exCFFiles. The OleDragDrop event notifies your application that the user drags some data on the control.

method ExDataObject.GetData (Format as Integer)

Returns data from a DataObject object in the form of a variant.

Type	Description
Format as Integer	An exClipboardFormatEnum expression that defines the data's format

Return	Description
Variant	A Variant value that contains the ExDataObject's data in the given format

Use GetData property to retrieve the clipboard's data that has been dragged to the control. It's possible for the GetData and SetData methods to use data formats other than exClipboardFormatEnum , including user-defined formats registered with Windows via the RegisterClipboardFormat() API function. The GetData method always returns data in a byte array when it is in a format that it is not recognized. Use the Files property to retrieves the filenames if the format of data is exCFFiles

method ExDataObject.GetFormat (Format as Integer)

Returns a value indicating whether the ExDataObject's data is of specified format.

Type	Description
Format as Integer	A constant or value that specifies a clipboard data format like described in exClipboardFormatEnum enum.

Return	Description
Boolean	A boolean value that indicates whether the ExDataObject's data is of specified format.

Use the GetFormat property to verify if the ExDataObject's data is of a specified clipboard format. The GetFormat property retrieves True, if the ExDataObject's data format matches the given data format.

method ExDataObject.SetData ([Value as Variant], [Format as Variant])

Inserts data into a ExDataObject object using the specified data format.

Type	Description
Value as Variant	A data that is going to be inserted to ExDataObject object.
Format as Variant	A constant or value that specifies the data format, as described in exClipboardFormatEnum enum

Use SetData property to insert data for OLE drag and drop operations. Use the Files property is you are going to add new files to the clipboard data. The OleDragDrop event notifies your application that the user drags some data on the control.

ExDataObjectFiles object

The ExDataObjectFiles contains a collection of filenames. The ExDataObjectFiles object is used in OLE Drag and drop events. In order to get the list of files used in drag and drop operations you have to use the Files property. The OleDragDrop event notifies your application that the user drags some data on the control. The ExDataObjectFiles object supports the following properties and methods:

Name	Description
Add	Adds a filename to the Files collection
Clear	Removes all file names in the collection.
Count	Returns the number of file names in the collection.
Item	Returns an specific file name.
Remove	Removes an specific file name.

method ExDataObjectFiles.Add (FileName as String)

Adds a filename to the Files collection

Type	Description
FileName as String	A string expression that indicates a filename.

Use Add method to add your files to ExDataObject object. The OleStartDrag event notifies your application that the user starts dragging items.

method ExDataObjectFiles.Clear ()

Removes all file names in the collection.

Type	Description

Use the Clear method to remove all filenames from the collection.

property ExDataObjectFiles.Count as Long

Returns the number of file names in the collection.

Type	Description
Long	A long value that indicates the count of elements into collection.

You can use "for each" statements if you are going to enumerate the elements into ExDataObjectFiles collection.

property ExDataObjectFiles.Item (Index as Long) as String

Returns a specific file name given its index.

Type	Description
Index as Long	A long expression that indicates the filename's index.
String	A string value that indicates the filename.

method ExDataObjectFiles.Remove (Index as Long)

Removes a specific file name given its index into collection.

Type	Description
Index as Long	A long expression that indicates the index of filename into collection.

Use Clear method to remove all filenames,.

G2antt object

Tip The /COM object can be placed on a HTML page (with usage of the HTML object tag: <object classid="clsid:...">) using the class identifier: {CD481F4D-2D25-4759-803F-752C568F53B7}. The object's program identifier is: "Exontrol.G2antt". The /COM object module is: "ExG2antt.dll"

The Exontrol's ExG2antt component is our approach to create timeline charts (also known as Gantt charts). Gantt chart is a time-phased graphic display of activity durations. Activities are listed with other tabular information on the left side with time intervals over the bars. Activity durations are shown in the form of horizontal bars. The G2antt object supports the following properties and methods:

Name	Description
AllowChartScrollHeader	Specifies whether the user can scroll the chart by clicking the chart's header and move the cursor to a new position.
AllowChartScrollPage	Specifies whether the chart's horizontal scroll bar includes buttons to scroll the chart page by page.
AllowGroupBy	Indicates whether the control supports Group-By view.
AllowSelectNothing	Specifies whether the current selection is erased, once the user clicks outside of the items section.
AnchorFromPoint	Retrieves the identifier of the anchor from point.
AntiAliasing	Specifies whether smoothing (antialiasing) is applied to lines, curves, edges of the objects in the control.
Appearance	Retrieves or sets the control's appearance.
ApplyFilter	Applies the filter.
ASCIILower	Specifies the set of lower characters.
ASCIIUpper	Specifies the set of upper characters.
AttachTemplate	Attaches a script to the current object, including the events, from a string, file, a safe array of bytes.
AutoDrag	Gets or sets a value that indicates the way the component supports the AutoDrag feature.
AutoEdit	Specifies whether the cell is edited once that it is focused.
AutoSearch	Enables or disables the auto search feature.
BackColor	Retrieves or sets a value that indicates the control's background color.
BackColorAlternate	Specifies the background color used to display alternate items in the control.

BackColorHeader	Specifies the header's background color.
BackColorLevelHeader	Specifies the multiple levels header's background color.
BackColorLock	Retrieves or sets a value that indicates the control's background color for the locked area.
BackColorSortBar	Retrieves or sets a value that indicates the sort bar's background color.
BackColorSortBarCaption	Returns or sets a value that indicates the caption's background color in the control's sort bar.
Background	Returns or sets a value that indicates the background color for parts in the control.
BeginUpdate	Maintains performance when items are added to the control one at a time. This method prevents the control from painting until the EndUpdate method is called.
BorderStyle	Retrieves or sets the border style of the control.
CauseValidateValue	Returns or sets a value that determines whether the ValidateValue event occurs before the user changes the cell's value.
Chart	Gets the chart object.
ChartOnLeft	Specifies whether the chart area is displayed on the left or right side of the component.
CheckImage	Retrieves or sets a value that indicates the image used by cells of checkbox type.
ClearFilter	Clears the filter.
ColumnAutoResize	Returns or sets a value indicating whether the control will automatically size its visible columns to fit on the control's client width.
ColumnFromPoint	Retrieves the column from point.
Columns	Retrieves the control's column collection.
ColumnsAllowSizing	Retrieves or sets a value that indicates whether a user can resize columns at run-time.
ColumnsFloatBarSortOrder	Specifies the sorting order for the columns being shown in the control's columns floating panel.
ColumnsFloatBarVisible	Retrieves or sets a value that indicates whether the the columns float bar is visible or hidden.
ConditionalFormats	Retrieves the conditional formatting collection.
	Retrieves or sets a value indicating whether the control will

ContinueColumnScroll	automatically scroll the visible columns by pixel or by column width.
Copy	Copies the control's content to the clipboard, in the EMF format.
CopyTo	Exports the control's view to an EMF file.
CountLockedColumns	Retrieves or sets a value indicating the number of locked columns. A locked column is not scrollable.
DataSource	Retrieves or sets a value that indicates the data source for object.
Debug	Displays debug information.
DefaultEditorOption	Specifies a default option for an editor.
DefaultItemHeight	Retrieves or sets a value that indicates the default item height.
Description	Changes descriptions for control objects.
DetectAddNew	Specifies whether the control detects when a new record is added to the bounded recordset.
DetectDelete	Specifies whether the control detects when a record is deleted from the bounded recordset.
DiscardValidateValue	Cancels the current validation process, and restores back the modified cells.
DrawGridLines	Retrieves or sets a value that indicates whether the grid lines are visible or hidden.
DrawPartItem	Indicates the handle of the item where the BeforeDrawPart / AfterDrawPart event occurs.
DrawPartKey	Specifies the key of the owner bar to be painted during BeforeDrawPart / AfterDrawPart event.
Edit	Edits the focused cell.
EditClose	Closes the current editor.
Editing	Specifies the window's handle of the built-in editor while the control is running in edit mode.
EditingText	Specifies the caption of the editor during editing.
Enabled	Enables or disables the control.
EndUpdate	Resumes painting the control after painting is suspended by the BeginUpdate method.
	Specifies whether the control ensures that the focused

EnsureOnSort	item fits the control's client area, when the user sorts the items.
EnsureVisibleColumn	Scrolls the control's content to ensure that the column fits the client area.
EventParam	Retrieves or sets a value that indicates the current's event parameter.
ExecuteTemplate	Executes a template and returns the result.
ExpandOnDblClick	Specifies whether the item is expanded or collapsed if the user dbl clicks the item.
ExpandOnKeys	Specifies a value that indicates whether the control expands or collapses a node when user presses arrow keys.
ExpandOnSearch	Expands items automatically while user types characters to search for a specific item.
Export	Exports the control's data to a CSV format.
FilterBarBackColor	Specifies the background color of the control's filter bar.
FilterBarCaption	Specifies the filter bar's caption.
FilterBarDropDownHeight	Specifies the height of the drop down filter window proportionally with the height of the control's list.
FilterBarFont	Retrieves or sets the font for control's filter bar.
FilterBarForeColor	Specifies the foreground color of the control's filter bar.
FilterBarHeight	Specifies the height of the control's filter bar. If the value is less than 0, the filter bar is automatically resized to fit its description.
FilterBarPrompt	Specifies the caption to be displayed when the filter pattern is missing.
FilterBarPromptColumns	Specifies the list of columns to be used when filtering using the prompt.
FilterBarPromptPattern	Specifies the pattern for the filter prompt.
FilterBarPromptType	Specifies the type of the filter prompt.
FilterBarPromptVisible	Shows or hides the filter prompt.
FilterCriteria	Retrieves or sets the filter criteria.
FilterInclude	Specifies the items being included after the user applies the filter.
FocusColumnIndex	Specifies the index of focused column.

Font	Retrieves or sets the control's font.
ForeColor	Retrieves or sets a value that indicates the control's foreground color.
ForeColorHeader	Specifies the header's foreground color.
ForeColorLock	Retrieves or sets a value that indicates the control's foreground color for the locked area.
ForeColorSortBar	Retrieves or sets a value that indicates the sort bar's foreground color.
FormatABC	Formats the A,B,C values based on the giving expression and returns the result.
FormatAnchor	Specifies the visual effect for anchor elements in HTML captions.
FreezeEvents	Prevents the control to fire any event.
FullRowSelect	Enables full-row selection in the control.
GetItems	Gets the collection of items into a safe array,
GridLineColor	Specifies the grid line color.
GridLineStyle	Specifies the style for gridlines in the list part of the control.
Group	Forces the control to do a regrouping of the columns.
HasButtons	Adds a button to the left side of each parent item. The user can click the button to expand or collapse the child items as an alternative to double-clicking the parent item.
HasButtonsCustom	Specifies the index of icons for +/- signs when the HasButtons property is exCustom.
HasLines	Enhances the graphic representation of a tree control's hierarchy by drawing lines that link child items to their corresponding parent item.
HeaderAppearance	Retrieves or sets a value that indicates the header's appearance.
HeaderEnabled	Enables or disables the control's header.
HeaderHeight	Retrieves or sets a value indicating the control's header height.
HeaderSingleLine	Specifies whether the control resizes the columns header and wraps the captions in single or multiple lines.
HeaderVisible	Retrieves or sets a value that indicates whether the

	control's header is visible or hidden.
HideSelection	Returns a value that determines whether selected item appears highlighted when a control loses the focus.
HotBackColor	Retrieves or sets a value that indicates the hot-tracking background color.
HotForeColor	Retrieves or sets a value that indicates the hot-tracking foreground color.
HTMLPicture	Adds or replaces a picture in HTML captions.
hWnd	Retrieves the control's window handle.
HyperLinkColor	Specifies the hyperlink color.
Images	Sets at runtime the control's image list. The Handle should be a handle to an Image List Control.
ImageSize	Retrieves or sets the size of icons the control displays.
Indent	Retrieves or sets the amount, in pixels, that child items are indented relative to their parent items.
IsGrouping	Indicates whether the control is grouping the items.
ItemFromPoint	Retrieves the item from point.
Items	Retrieves the control's item collection.
ItemsAllowSizing	Retrieves or sets a value that indicates whether a user can resize items at run-time.
Layout	Saves or loads the control's layout, such as positions of the columns, scroll position, filtering values.
LinesAtRoot	Link items at the root of the hierarchy.
LoadXML	Loads an XML document from the specified location, using MSXML parser.
MarkSearchColumn	Retrieves or sets a value that indicates whether the searching column is marked or unmarked
MarkTooltipCells	Retrieves or sets a value that indicates wheter the control marks the cells that have tooltips.
MarkTooltipCellsImage	Specifies a value that indicates the index of icon being displayed in the cells that have tooltips.
OLEDrag	Causes a component to initiate an OLE drag/drop operation.
OLEDropMode	Returns or sets how a target component handles drop operations

OnResizeControl	Specifies whether the list or the chart part is resized once the control is resized.
Picture	Retrieves or sets a graphic to be displayed in the control.
PictureDisplay	Retrieves or sets a value that indicates the way how the graphic is displayed on the control's background
PictureDisplayLevelHeader	Retrieves or sets a value that indicates the way how the graphic is displayed on the control's header background.
PictureLevelHeader	Retrieves or sets a graphic to be displayed in the control's header when multiple levels is on.
PutItems	Adds an array of integer, long, date, string, double, float, or variant arrays to the control.
PutRes	The PutRes method associates an eXG2antt (Source) control with another eXG2antt (Target) control, using the Items.ItemBar(exBarResources).
RadioImage	Retrieves or sets a value that indicates the image used by cells of radio type.
RClickSelect	Retrieves or sets a value that indicates whether an item is selected using right mouse button.
ReadOnly	Retrieves or sets a value that indicates whether the control is readonly.
Refresh	Refreshes the control's content.
RemoveSelection	Removes the selected links/bars from the chart if exists, else it removes the selected items (including the descendents)
ReplaceIcon	Adds a new icon, replaces an icon or clears the control's image list.
ResHandle	The ResHandle property indicates the handle to be used for ResHandle parameter of the PutRes method.
RightToLeft	Indicates whether the component should draw right-to-left for RTL languages.
SaveXML	Saves the control's content as XML document to the specified location, using the MSXML parser.
Scroll	Scrolls the control's content.
ScrollBars	Returns or sets a value that determines whether the control has horizontal and/or vertical scroll bars.
ScrollButtonHeight	Specifies the height of the button in the vertical scrollbar.

ScrollButtonWidth	Specifies the width of the button in the horizontal scrollbar.
ScrollBySingleLine	Retrieves or sets a value that indicates whether the control scrolls the lines to the end. If you have at least a cell that has SingleLine false, you have to check the ScrollBySingleLine property..
ScrollFont	Retrieves or sets the scrollbar's font.
ScrollHeight	Specifies the height of the horizontal scrollbar.
ScrollOrderParts	Specifies the order of the buttons in the scroll bar.
ScrollPartCaption	Specifies the caption being displayed on the specified scroll part.
ScrollPartCaptionAlignment	Specifies the alignment of the caption in the part of the scroll bar.
ScrollPartEnable	Indicates whether the specified scroll part is enabled or disabled.
ScrollPartVisible	Indicates whether the specified scroll part is visible or hidden.
ScrollPos	Specifies the vertical/horizontal scroll position.
ScrollThumbSize	Specifies the size of the thumb in the scrollbar.
ScrollToolTip	Specifies the tooltip being shown when the user moves the scroll box.
ScrollWidth	Specifies the width of the vertical scrollbar.
SearchColumnIndex	Retrieves or sets a value indicating the column's index that is used for auto search feature.
SelBackColor	Retrieves or sets a value that indicates the selection background color.
SelBackMode	Retrieves or sets a value that indicates whether the selection is transparent or opaque.
SelectByDrag	Specifies whether the user selects multiple items by dragging.
SelectColumn	Specifies whether the user selects cells only in SelectColumnIndex column, while FullRowSelect property is False.
SelectColumnIndex	Retrieves or sets a value that indicates control column's index where the user is able to select an item. It has effect only for FullRowSelect = false.

SelectColumnInner	Retrieves or sets a value that indicates the index of the inner cell that's selected.
SelectOnRelease	Indicates whether the selection occurs when the user releases the mouse button.
SelForeColor	Retrieves or sets a value that indicates the selection foreground color.
ShowFocusRect	Retrieves or sets a value indicating whether the control draws a thin rectangle arround the focused item.
ShowImageList	Specifies whether the control's image list window is visible or hidden.
ShowLockedItems	Retrieves or sets a value that indicates whether the control displays the locked items.
ShowToolTip	Shows the specified tooltip at given position.
SingleSel	Retrieves or sets a value that indicates whether the control supports single or multiple selection.
SingleSort	Returns or sets a value that indicates whether the control supports sorting by single or multiple columns.
SortBarCaption	Specifies the caption being displayed on the control's sort bar when the sort bar contains no columns.
SortBarColumnWidth	Specifies the maximum width a column can be in the control's sort bar.
SortBarHeight	Retrieves or sets a value that indicates the height of the control's sort bar.
SortBarVisible	Retrieves or sets a value that indicates whether control's sort bar is visible or hidden.
SortOnClick	Retrieves or sets a value that indicates whether the control sorts automatically the data when the user click on column's caption.
Statistics	Gives statistics data of objects being hold by the control.
Template	Specifies the control's template.
TemplateDef	Defines inside variables for the next Template/ExecuteTemplate call.
TemplatePut	Defines inside variables for the next Template/ExecuteTemplate call.
TooltipCellsColor	Retrieves or sets a value that indicates the color used to make the cells that have tooltips.

ToolTipDelay	Specifies the time in ms that passes before the ToolTip appears.
ToolTipFont	Retrieves or sets the tooltip's font.
ToolTipMargin	Defines the size of the control's tooltip margins.
ToolTipPopDelay	Specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control.
ToolTipWidth	Specifies a value that indicates the width of the tooltip window, in pixels.
TreeColumnIndex	Retrieves or sets a value indicating the column's index where the hierarchy will be displayed.
Ungroup	Ungroups the columns, if they have been previously grouped.
UseTabKey	Specifies whether the TAB key is used to change the searching column.
UseVisualTheme	Specifies whether the control uses the current visual theme to display certain UI parts.
Version	Retrieves the control's version.
VisualAppearance	Retrieves the control's appearance.
VisualDesign	Invokes the control's VisualAppearance designer.
WordFromPoint	Retrieves the word from the cursor.

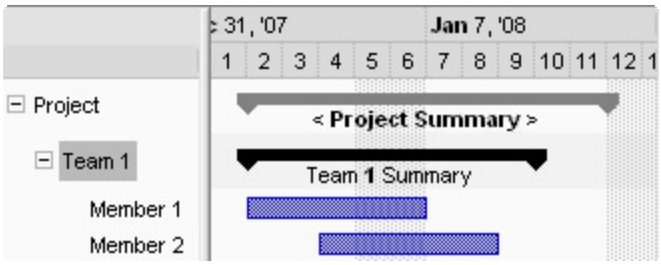
property G2antt.AllowChartScrollHeader as Boolean

Specifies whether the user can scroll the chart by clicking the chart's header and move the cursor to a new position.

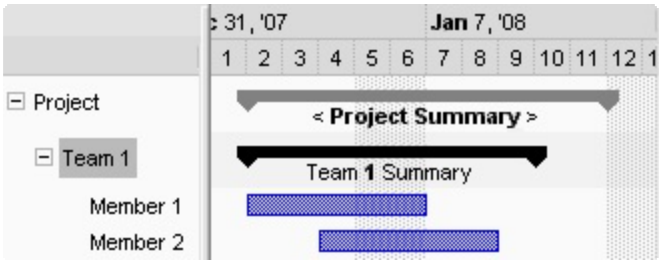
Type	Description
Boolean	A boolean expression that specifies whether the user can click and scroll the chart's header.

By default, the AllowChartScrollHeader property is True. In this case, if the user clicks the chart's header and drag the mouse to a new position the chart gets scrolled. While scrolling the hand cursor is being displayed. You are still able to scroll the control's chart using the horizontal scroll bar in the chart area. Use the FirstVisibleDate property to display a different date in the chart. If the user clicks and releases the mouse in the chart's header a date gets selected so it gets marked using the MarkSelectDateColor property.

The following screen show how the charts gets scrolled once the user clicks and drags the mouse on the chart's header (AllowChartScrollHeader property is True) :



The following screen show how a new date gets selected once the user clicks a date in the chart's header:



property G2antt.AllowChartScrollPage as Boolean

Specifies whether the chart's horizontal scroll bar includes buttons to scroll the chart page by page.

Type	Description
Boolean	A Boolean expression that specifies whether the control includes the exLeftB5Part and exRightB1Part buttons to the chart's horizontal scroll bar.

By default AllowChartScrollPage property is False. Use the AllowChartScrollPage property to add fast scroll to the chart's page. If the AllowChartScrollPage property is True, the control automatically adds the exLeftB5Part and exRightB1Part buttons to the chart's horizontal scroll bar. Once that the user clicks any of these buttons the chart is scrolled page by page. Use the ScrollPartCaption property to specify the caption being displayed in any part of the control's scrollbar. Use the <img> HTML tag to include icons or custom size pictures to your scroll buttons.

property G2antt.AllowGroupBy as Boolean

Indicates whether the control supports Group-By view.

Type	Description
Boolean	A Boolean expression that specifies whether the user can group the items.

By default, the AllowGroupBy property is False. Set the AllowGroupBy property on True, to allow the user to group the items by dragging the column's header to control's sort bar. The SortBarVisible property specifies whether the control's sort bar is visible or hidden. If the control's sort bar is visible, the user can drag and drop columns to it, so the column get sorted and items grouped. The AddGroupItem event is fired when a new grouping items is added to the control's list. *You can use the AddGroupItem event, to add headers or footers during grouping, customize the aggregate formula to be displayed on different columns, while dropping a column to the sortbar.* The Column.AllowGroupBy property may be used to prevent grouping a specific column. The AllowSort property indicates whether the user can sort a column by clicking the column's header. The IsGrouping property specifies whether the control is grouping/ungrouping items. During grouping, the control keeps the items indentation, in other words, a child item will be a child after or before grouping. The LayoutChanged event is fired when the user changes the layout of the control, including dragging a column to the sort bar. The SortBarColumnsCount property indicates the number of the columns being grouped. The SortBarColumn property indicates the column being sorted giving its position in the sort bar. The Group/Ungroup method groups or ungroup the control's list. For instance, you can remove the grouping items, by calling the Ungroup method. The GroupByFormatCell property determines the format of the cell to be displayed in the grouping item, when the column gets sorted.

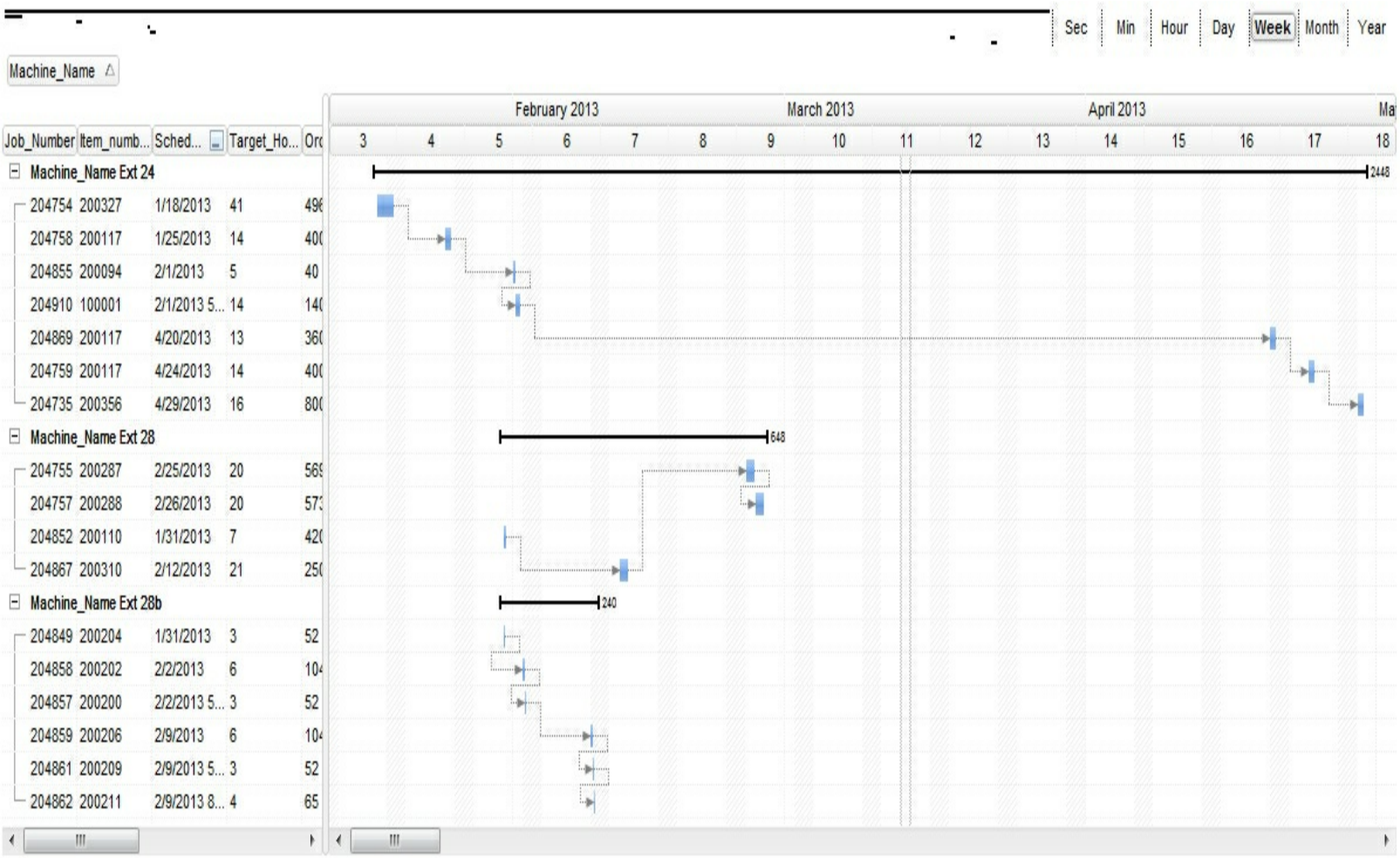
The following movies show how Group-By works:

- Group By support - the user can drag and drop one or more columns to the sort bar or group-by bar so the columns get sorted and grouped accordingly.
- Keep Indent - You can keep the indentation of the sub-items/children, once the user groups the rows.
- Header/Footer - Headers and footers support, to display aggregate functions like sum, min, max, and so on.
- CRD support - Can be combined with the exCRD, so you can have the rows being arranged the way you want.

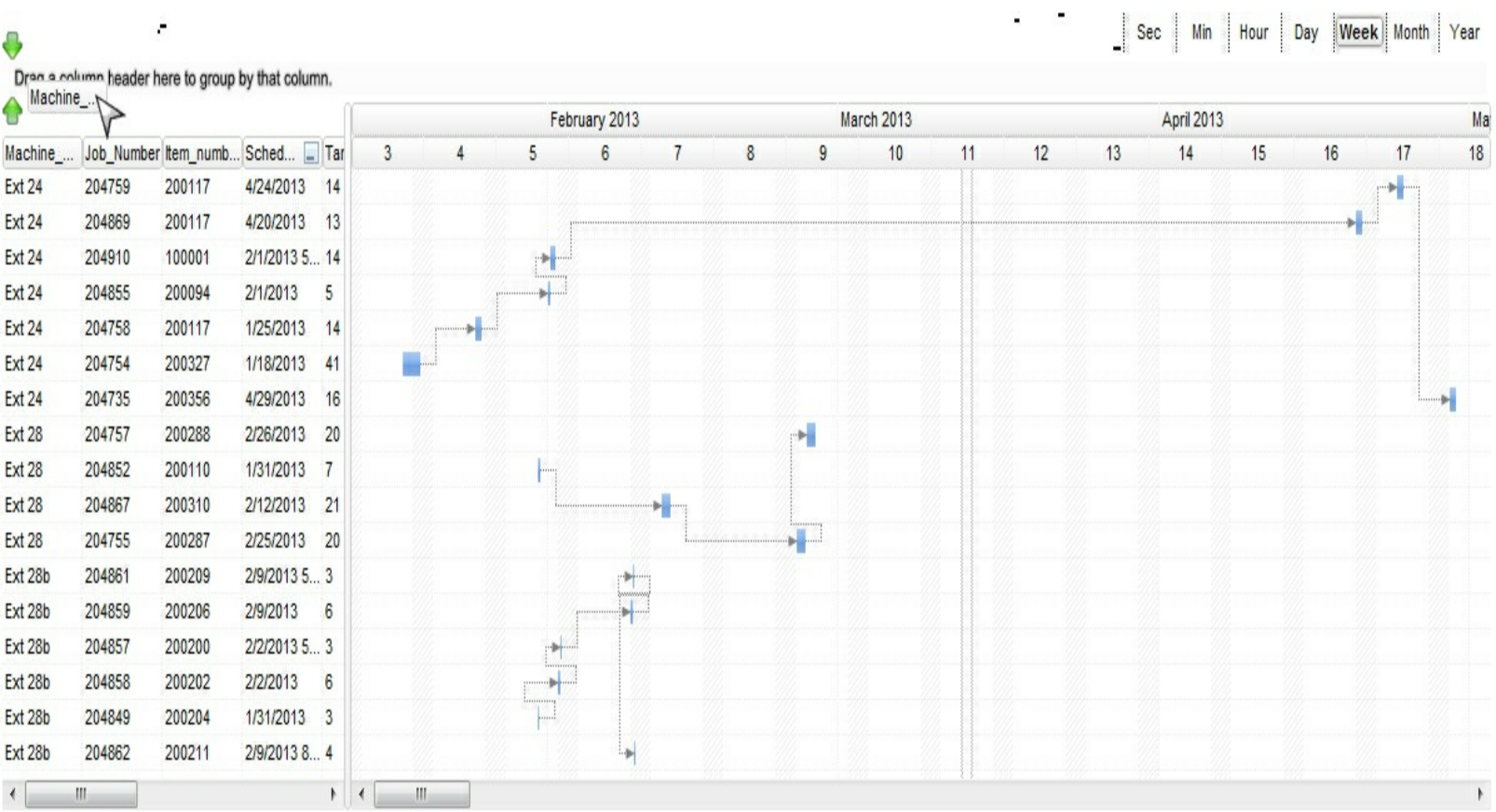
In case you need more than a Group-By feature, you should check the Exontrol's eXPivot. The Exontrol's eXPivot tool is our approach to provide data summarization, as a pivot

table. A pivot-table can automatically sort, count, total or give the average of the data stored in one table or spreadsheet. The user sets up and changes the summary's structure by dragging and dropping fields graphically. The eXPivot component lets the user changes its visual appearance using skins, each one providing an additional visual experience that enhances viewing pleasure. Skins are relatively easy to build and put on any part of the control.

The following screen shot shows the control after grouping:



The following screen shot shows the control before grouping:



The AllowGroupBy property/feature has no effect if:

- SingleSort property is True.

property G2antt.AllowSelectNothing as Boolean

Specifies whether the current selection is erased, once the user clicks outside of the items section.

Type	Description
Boolean	A Boolean expression that specifies whether the current selection is erased, once the user clicks outside of the items section.

By default, the AllowSelectNothing property is False. The AllowSelectNothing property specifies whether the current selection is erased, once the user clicks outside of the items section. For instance, if the control's SingleSel property is True, and AllowSelectNothing property is True, you can un-select the single-selected item if pressing the CTRL + Space, or by CTRL + click.

property G2antt.AnchorFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as String

Retrieves the identifier of the anchor from point.

Type	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in client coordinates.
String	A String expression that specifies the identifier (id) of the anchor element from the point, or empty string if there is no anchor element at the cursor

Use the AnchorFromPoint property to determine the identifier of the anchor from the point. Use the <a id;options> anchor elements to add hyperlinks to cell's caption. The control fires the AnchorClick event when the user clicks an anchor element. Use the ShowToolTip method to show the specified tooltip at given or cursor coordinates. The MouseMove event is generated continually as the mouse pointer moves across the control.

The following VB sample displays (as tooltip) the identifier of the anchor element from the cursor:

```
Private Sub G2antt1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
  With G2antt1
    .ShowToolTip .AnchorFromPoint(-1, -1)
  End With
End Sub
```

The following VB.NET sample displays (as tooltip) the identifier of the anchor element from the cursor:

```
Private Sub AxG2antt1_MouseMoveEvent(ByVal sender As System.Object, ByVal e As AxEXG2ANTTLib._IG2anttEvents_MouseMoveEvent) Handles AxG2antt1.MouseMoveEvent
  With AxG2antt1
    .ShowToolTip(.get_AnchorFromPoint(-1, -1))
  End With
End Sub
```

The following C# sample displays (as tooltip) the identifier of the anchor element from the cursor:

```
private void axG2antt1_MouseMoveEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_MouseMoveEvent e)
{
   axG2antt1.ShowToolTip(axG2antt1.get_AnchorFromPoint(-1, -1));
}
```

The following C++ sample displays (as tooltip) the identifier of the anchor element from the cursor:

```
void OnMouseMoveG2antt1(short Button, short Shift, long X, long Y)
{
   COleVariant vtEmpty; V_VT( &vtEmpty ) = VT_ERROR;
   m_g2antt.ShowToolTip( m_g2antt.GetAnchorFromPoint( -1, -1 ), vtEmpty, vtEmpty,
vtEmpty );
}
```

The following VFP sample displays (as tooltip) the identifier of the anchor element from the cursor:

```
*** ActiveX Control Event ***
LPARAMETERS button, shift, x, y

with thisform
   With .G2antt1
      .ShowToolTip(.AnchorFromPoint(-1, -1))
   EndWith
endwith
```

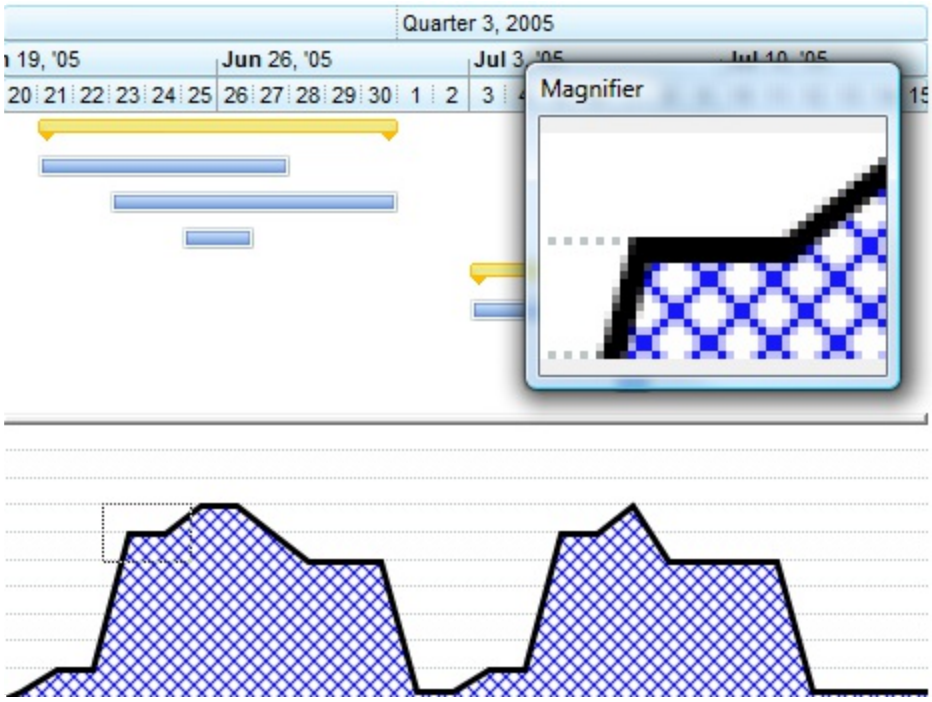
property G2antt.AntiAliasing as Boolean

Specifies whether smoothing (antialiasing) is applied to lines, curves, edges of the objects in the control.

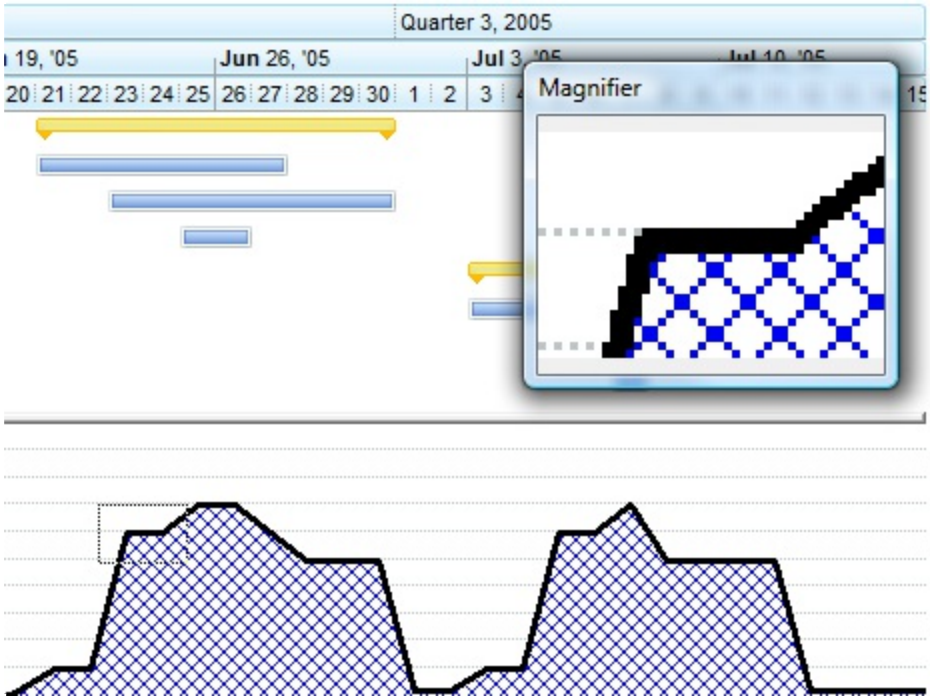
Type	Description
Boolean	A Boolean expression that specifies whether the control uses the antialiasing rendering to show the lines, curves or edges.

By default, the AntiAliasing property is False. In other words, the AntiAliasing property determines the rendering quality for different objects in the control. The most used object where the antialiasing feature can be used is the chart's histogram where curves can be shown smoothly. Use the HistogramVisible property to specifies whether the chart's histogram is visible or hidden. Use the AllowLinkBars property to whether the user can link bars at runtime.

The following figure illustrates the visual distortion that occurs when anti-aliasing is used.



The following figure illustrates the visual distortion that occurs when anti-aliasing is not used.

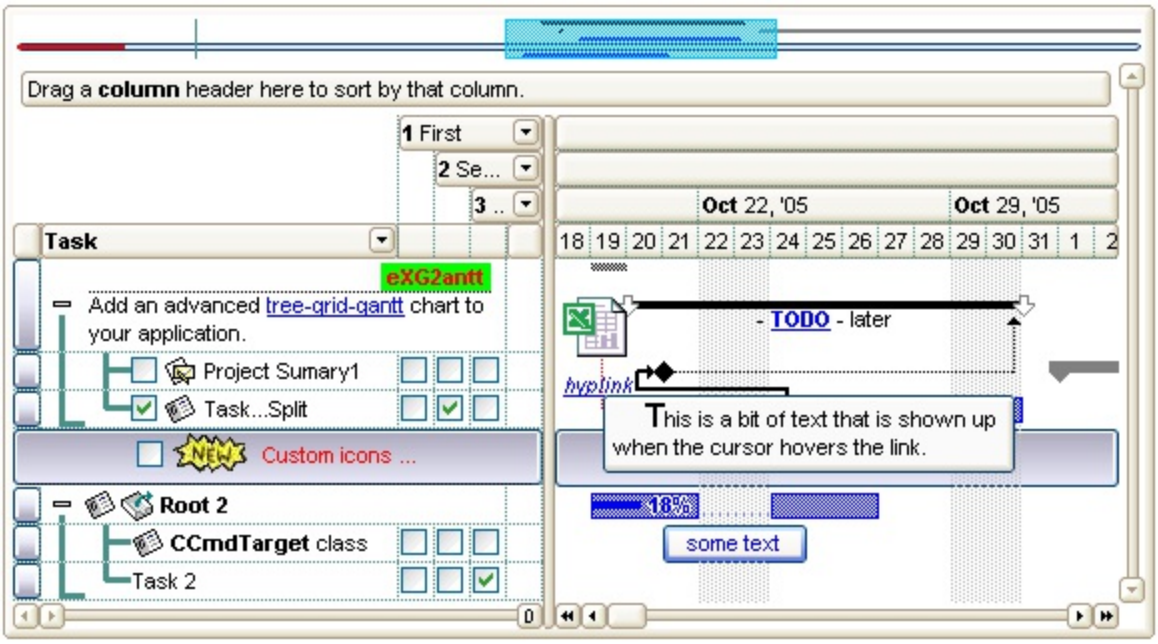

Quarter 3, 2005
19, '05
Jun 26, '05
Jul 3, '05
20 21 22 23 24 25 26 27 28 29 30 1 2 3
Magnifier

property G2antt.Appearance as AppearanceEnum

Retrieves or sets the control's appearance.

Type	Description
AppearanceEnum	An AppearanceEnum expression that indicates the control's appearance, or a color expression whose last 7 bits in the high significant byte of the value indicates the index of the skin in the Appearance collection, being displayed as control's borders. For instance, if the Appearance = 0x1000000, indicates that the first skin object in the Appearance collection defines the control's border. ***The Client object in the skin, defines the client area of the control. The list/hierarchy/chart, scrollbars are always shown in the control's client area. The skin may contain transparent objects, and so you can define round corners. The normal.ebn file contains such of objects. Use the eXButton's Skin builder to view or change this file***

Use the Appearance property to specify the control's border. Use the HeaderAppearance property to change the control's header bar appearance. Use the Add method to add new skins to the control. Use the BackColor property to specify the control's background color. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips.



The following VB sample changes the visual aspect of the borders of the control (please check the above picture for round corners):

```
With G2antt1
   .BeginUpdate
      .VisualAppearance.Add &H16, "c:\temp\normal.ebn"
      .Appearance = &H16000000
      .BackColor = RGB(250, 250, 250)
   .EndUpdate
End With
```

The following VB.NET sample changes the visual aspect of the borders of the control:

```
With AxG2antt1
   .BeginUpdate()
   .VisualAppearance.Add(&H16, "c:\temp\normal.ebn")
   .Appearance = &H16000000
   .BackColor = Color.FromArgb(250, 250, 250)
   .EndUpdate()
End With
```

The following C# sample changes the visual aspect of the borders of the control:

```
axG2antt1.BeginUpdate();
axG2antt1.VisualAppearance.Add(0x16, "c:\\temp\\normal.ebn");
axG2antt1.Appearance = (EXG2ANTTLib.AppearanceEnum)0x16000000;
axG2antt1.BackColor = Color.FromArgb(250, 250, 250);
axG2antt1.EndUpdate();
```

The following C++ sample changes the visual aspect of the borders of the control:

```
m_g2antt.BeginUpdate();
m_g2antt.GetVisualAppearance().Add( 0x16, COleVariant( "c:\\temp\\normal.ebn" ) );
m_g2antt.SetAppearance( 0x16000000 );
m_g2antt.SetBackColor( RGB(250,250,250) );
m_g2antt.EndUpdate();
```

The following VFP sample changes the visual aspect of the borders of the control:

```
with thisform.G2antt1
   .BeginUpdate
      .VisualAppearance.Add(0x16, "c:\temp\normal.ebn")
```

```
      .Appearance = 0x16000000
      .BackColor = RGB(250, 250, 250)
   .EndUpdate
endwith
```

method G2antt.ApplyFilter ()

Applies the filter.

Type	Description

The ApplyFilter method updates the control's content once that user sets the filter using the Filter and FilterType properties. Use the ClearFilter method to clear the control's filter. Use the DisplayFilterButton property to show the filter drop down button in the column's caption. Use the FilterInclude property to specify whether the child items should be included to the list when the user applies the filter. Use the FilterCriteria property to filter items using the AND, OR and NOT operators. Use the ShowFilter method to show programmatically the column's drop down filter window. The VisibleItemCount property retrieves the number of visible items in the list. The control fires the FilterChanging event just before applying the filter, and FilterChange once the list gets filtered. The FilterBarPromptVisible property specifies whether the filter prompt is visible or hidden. The filter prompt feature allows you to filter the items as you type while the filter bar is visible on the bottom part of the list area. The Filter prompt feature allows at runtime filtering data on hidden columns too.

property G2antt.ASCIILower as String

Specifies the set of lower characters.

Type	Description
String	A string expression that indicates the set of lower characters used by auto search feature.

The ASCIILower and ASCIIUpper properties helps you to specify the set of characters that are used by the auto search feature (incremental search). If you want to make the auto search feature case sensitive you have to use ASCIIUpper = "" . By default, the ASCIILower property is = "abcdefghijklmnopqrstuvwxyz�שבסקתס"יגהאוחרכטלמנפצע

property G2antt.ASCIIUpper as String

Specifies the set of upper characters.

Type	Description
String	A string expression that indicates the set of upper characters used by auto search feature.

The ASCIILower and ASCIIUpper properties helps you to specify the set of characters that are used by the auto search (incremental search) feature. If you want to make the auto search feature case sensitive you have to use ASCIIUpper = "" . By default, the ASCIIUpper property is = "ABCDEFGHIJKLMNOPQRSTUVWXYZÜÉÂÄŔĹÇĘËČĎÎĚÔÖŇŰŮÁÍÓÚŃ"

method G2antt.AttachTemplate (Template as Variant)

Attaches a script to the current object, including the events, from a string, file, a safe array of bytes.

Type	Description
Template as Variant	A string expression that specifies the Template to execute.

The AttachTemplate/x-script code is a simple way of calling control/object's properties, methods/events using strings. The AttachTemplate features allows you to attach a x-script code to the component. The AttachTemplate method executes x-script code (including events), from a string, file or a safe array of bytes. This feature allows you to run any x-script code for any configuration of the component /COM, /NET or /WPF. Exontrol owns the x-script implementation in its easiest form and it does not require any VB engine or whatever to get executed. The x-script code can be converted to several programming languages using the eXHelper tool.

The following sample opens the Windows Internet Explorer once the user clicks the control (/COM version):

```
AttachTemplate("handle Click(){ CreateObject(`internetexplorer.application`){ Visible = True; Navigate(`https://www.exontrol.com`) } } ")
```

This script is equivalent with the following VB code:

```
Private Sub G2antt1_Click()
   With CreateObject("internetexplorer.application")
      .Visible = True
      .Navigate ("https://www.exontrol.com")
   End With
End Sub
```

The AttachTemplate/x-script syntax in BNF notation is defined like follows:

<x-script> := <lines>
<lines> := <line>[<eol> <lines>] | <block>
<block> := <call> [<eol>] { [<eol>] <lines> [<eol>] } [<eol>]
<eol> := ";" | "\r\n"
<line> := <dim> | <createobject> | <call> | <set> | <comment> | <handle>[<eol>]{[<eol>] <lines>[<eol>]}[<eol>]
<dim> := "DIM" <variables>
<variables> := <variable> [, <variables>]

<variable> := "ME" | <identifier>
<createobject> := "CREATEOBJECT(`"<type>"`)"
<call> := <variable> | <property> | <variable>"."<property> | <createobject>"."<property>
<property> := [<property>"."]<identifier>["("<parameters>")"]
<set> := <call> "=" <value>
<property> := <identifier> | <identifier>"("[<parameters>]")"
<parameters> := <value> ["," <parameters>]
<value> := <boolean> | <number> | <color> | <date> | <string> | <createobject> | <call>
<boolean> := "TRUE" | "FALSE"
<number> := "0X"<hexa> | ["-"]<integer>["."<integer>]
<digit10> := 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<digit16> := <digit10> | A | B | C | D | E | F
<integer> := <digit10>[<integer>]
<hexa> := <digit16>[<hexa>]
<color> := "RGB("<integer>","<integer>","<integer>")"
<date> := "#"<integer>"/"<integer>"/"<integer>" "[<integer>":"<integer>":"<integer>"]"#"
<string> := ""<text>"" | "`"<text>"`"
<comment> := "'"<text>
<handle> := "handle " <event>
<event> := <identifier>"("[<eparameters>]")"
<eparameters> := <eparameter> ["," <eparameters>]
<parameters> := <identifier>

where:

<identifier> indicates an identifier of the variable, property, method or event, and should start with a letter.
<type> indicates the type the CreateObject function creates, as a progID for /COM version or the assembly-qualified name of the type to create for /NET or /WPF version
<text> any string of characters

The Template or x-script is composed by lines of instructions. Instructions are separated by "\n\r" (newline characters) or ";" character.

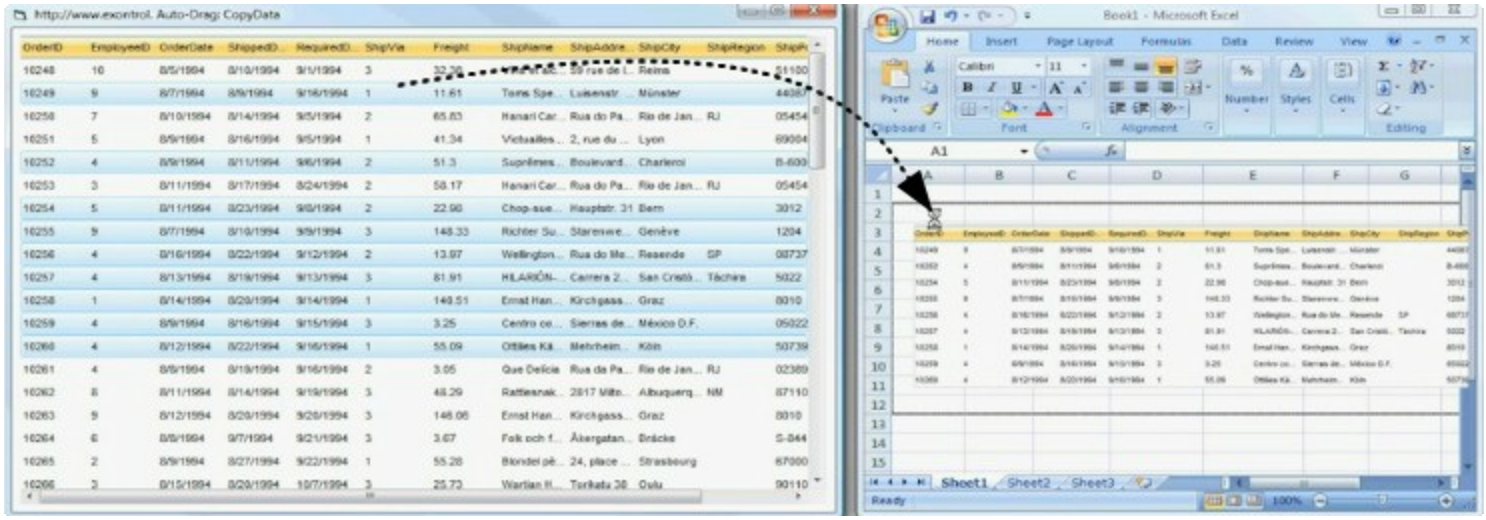
The advantage of the AttachTemplate relative to Template / ExecuteTemplate is that the AttachTemplate can add handlers to the control events.

property G2antt.AutoDrag as AutoDragEnum

Gets or sets a value that indicates the way the component supports the AutoDrag feature.

Type	Description
AutoDragEnum	An AutoDragEnum expression that specifies what the control does once the user clicks and start dragging an item.

By default, the AutoDrag property is exAutoDragNone(0). The AutoDrag feature indicates what the control does when the user clicks an item and starts dragging it. For instance, using the AutoDrag feature you can automatically lets the user to drag and drop the data to OLE compliant applications like Microsoft Word, Excel and so on. The SingleSel property specifies whether the control supports single or multiple selection. The AutoDrag feature adds automatically Drag and Drop, but you can still use the OLEDropMode property to handle the OLE Drag and Drop event for your custom action. If you need moving a bar from an item to another, you should use the Items.ItemBar(exBarCanMoveToAnother) property on True. The AllowAutoDrag event notifies your application once the user drag and drop the item to a new position.



The drag and drop operation starts:

- once the user clicks and moves the cursor up or down, if the SingleSel property is True.
- once the user clicks, and waits for a short period of time, if SingleSel property is False (multiple items in selection is allowed). In this case, you can drag and drop any item that is not selected, or a contiguously selection

Once the drag and drop operation starts the mouse pointer is changed to MOVE cursor if the operation is possible, else if the Drag and Drop operation fails or if it is not possible, the mouse pointer is changed to NO cursor.

If using the AutoDrag property on:

- exAutoDragPosition
- exAutoDragPositionKeepIndent
- exAutoDragPositionAny

the Drag and Drop starts only:

- item from cursor is a selectable (SelectableItem property on True, default) and sortable item (SortableItem property on True, default).
- if multiple items are selected, the selection is contiguously.

Use the AutoDrag property to allow Drag and Drop operations like follows:

- Ability to change the column or row position without having to manually add the OLE drag and drop events
- Ability to drag and drop the data as *text*, to your favorite Office applications, like Word, Excel, or any other OLE-Automation compliant
- Ability to drag and drop the data as it *looks*, to your favorite Office applications, like Word, Excel, or any other OLE-Automation compliant
- Ability to smoothly scroll the control's content moving the mouse cursor up or down
- and more ...

property G2antt.AutoEdit as Boolean

Specifies whether the cell may be edited when it has the focus.

Type	Description
Boolean	A boolean expression that indicates whether the editing operation starts once that a cell is focused.

Use the AutoEdit property to choose how the user edits the data. By default, the AutoEdit property is True. Use the Edit method to start editing the focused cell. Use the EditType property to define the column's editor. Use the ReadOnly property to make the control read only. Use the FocusItem property to retrieve the focused item. Use the FocusColumnIndex property to get the index of the column that's focused. Use the Editing property to check whether the control is in edit mode, or to get the window's handle for the built-in editor that's visible and focused.

The edit events are fired in the following order:

1. Edit event. Prevents editing cells, before showing the cell's editor.
2. EditOpen event. The edit operation started, the cell's editor is shown. The Editing property gives the window's handle of the built-in editor being started.
3. Change event. The Change event is fired only if the user types ENTER key, or the user selects a new value from a predefined data list.
4. EditClose event. The cell's editor is hidden and closed.

The following VB sample starts editing a cell when user presses the F4 key:

```
Private Sub G2antt1_KeyDown(KeyCode As Integer, Shift As Integer)
  ' Starts editing data when the user presses the F4 key
  With G2antt1
    If (Not .AutoEdit) Then
      If (KeyCode = vbKeyF4) Then .Edit
    End If
  End With
End Sub
```

The following C++ sample starts editing the focused cell when user presses the F4 key:

```
void OnKeyDownG2antt1(short FAR* KeyCode, short Shift)
```

```
{
   if ( *KeyCode == VK_F4 )
   {
      COleVariant vtMissing; V_VT( &vtMissing) = VT_ERROR;
      if ( m_g2antt.GetEditing() == 0 )
         m_g2antt.Edit( vtMissing );
   }
}
```

The following VB.NET sample starts editing the focused cell when user presses the F4 key:

```
Private Sub AxG2antt1_KeyDownEvent(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_KeyDownEvent) Handles AxG2antt1.KeyDownEvent
   If (Convert.ToUInt32(e.keyCode) = Convert.ToUInt32(Keys.F4)) Then
      AxG2antt1.Edit(Nothing)
   End If
End Sub
```

The following C# sample starts editing the focused cell when user presses the F4 key:

```
private void axG2antt1_KeyDownEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_KeyDownEvent e)
{
   if (Convert.ToUInt32(e.keyCode) == Convert.ToUInt32(Keys.F4))
      axG2antt1.Edit(null);
}
```

The following VFP sample starts editing the focused cell when user presses the F4 key:

```
private void axG2antt1_KeyDownEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_KeyDownEvent e)
{
   if (Convert.ToUInt32(e.keyCode) == Convert.ToUInt32(Keys.F4))
      axG2antt1.Edit(null);
}
```

property G2antt.AutoSearch as Boolean

Enables or disables the auto search feature.

Type	Description
Boolean	A boolean expression that indicates whether the auto search feature is enabled or disabled.

By default, the AutoSearch property is True. The auto-search feature is is commonly known as incremental search. An incremental search begins searching as soon as you type the first character of the search string. As you type in the search string, the control selects the item (and highlight the portion of the string that match where the string (as you have typed it so far) would be found. The control supports 'starts with' or 'contains' incremental search as described in the AutoSearch property of the Column object. Use the ExpandOnSearch property to expand items while user types characters in the control. Use the MarkSearchColumn property to specify whether the control draws a rectangle around the searching column.

The control highlights the characters as the user types them:

property G2antt.BackColor as Color

Retrieves or sets a value that indicates the control's background color.

Type	Description
Color	A color expression that indicates the control's background color.

The ExG2antt ActiveX Control can group the columns into two categories: locked and unlocked. The Locked category contains all the columns that are fixed to the left area of the client area. These columns cannot be scrolled horizontally. Use the CountLockedColumns to specify the number of locked columns. The unlocked are contains the columns that can be scrolled horizontally. To change the background color of the control's locked area use BackColorLock property. Use the SelBackColor property to specify the background color for selected items. Use the CellBackColor property to assign a different background color for a specified cell. Use the ItemBackColor property to specify the item's background color. Use the BackColorAlternate property to specify the background color used to display alternate items in the control. Use the Picture property to assign a picture to the control's background. Use the BackColor property to specify the chart's background color.

property G2antt.BackColorAlternate as Color

Specifies the background color used to display alternate items in the control.

Type	Description
Color	A color expression that indicates the alternate background color.

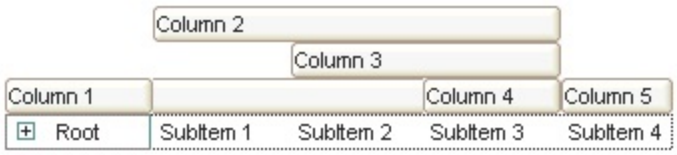
By default, the control's BackColorAlternate property is zero. The control ignores the BackColorAlternate property if it is 0 (zero). Use the BackColor property to specify the control's background color. Use the SelBackColor property to specify the selection background color.

property G2antt.BackColorHeader as Color

Specifies the header's background color.

Type	Description
Color	A color expression that indicates the background color for the control's header.

Use the BackColorHeader and ForeColorHeader properties to customize the control's header. Use the Def(exHeaderBackColor) property to change the background color or the visual appearance for a particular column, in the header area. If the Def(exHeaderForeColor) property is not zero, it defines the foreground color to paint the column's caption in the header area. Use the HeaderVisible property to hide the control's header. Use the BackColor property to specify the control's background color. Use the BackColorLevelHeader property to specify the background color of the header when it displays multiple levels. Use the BackColorSortBar property to specify the background color of the control's sort bar.



The following VB sample changes the visual appearance for the control's header. Shortly, we need to add a skin to the Appearance object using the Add method, and we need to set the last 7 bits in the BackColorHeader property to indicates the index of the skin that we want to use. The sample applies the " " to the control' header bar:

```
With G2antt1
   With .VisualAppearance
      .Add &H24, App.Path + "\header.ebn"
   End With
   .BackColorLevelHeader = RGB(255, 255, 255)
   .BackColorHeader = &H24000000
End With
```

The following C++ sample changes the visual aspect of the control' header bar:

```
#include "Appearance.h"
m_g2antt.GetVisualAppearance().Add( 0x24,
COleVariant(_T("D:\\Temp\\ExG2antt.Help\\header.ebn")) );
m_g2antt.SetBackColorHeader( 0x24000000 );
```

The following VB.NET sample changes the visual aspect of the control' header bar:

```
With AxG2antt1
   With .VisualAppearance
      .Add(&H24, "D:\Temp\ExG2antt.Help\header.ebn")
   End With
   .Template = "BackColorHeader = 603979776"
End With
```

The 603979776 value indicates the &H24000000 in hexadecimal.

The following C# sample changes the visual aspect of the control' header bar:

```
axG2antt1.VisualAppearance.Add(0x24, "D:\\Temp\\ExG2antt.Help\\header.ebn");
axG2antt1.Template = "BackColorHeader = 603979776";
```

The 603979776 value indicates the 0x24000000 in hexadecimal.

The following VFP sample changes the visual aspect of the control' header bar:

```
With thisform.G2antt1
   With .VisualAppearance
      .Add(36, "D:\Temp\ExG2antt.Help\header.ebn")
   EndWith
   .BackColorHeader = 603979776
EndWith
```

property G2antt.BackColorLevelHeader as Color

Specifies the multiple levels header's background color.

Type	Description
Color	A color expression that indicates the background color of the control's header bar.

Use the BackColorHeader and ForeColorHeader properties to define colors used to paint the control's header bar. Use the BackColorLevelHeader property to specify the background color of the control's header bar when multiple levels are displayed. Use the LevelKey property to display the control's header bar using multiple levels. If the control displays the header bar using multiple levels the HeaderHeight property gets the height in pixels of a single level in the header bar. The control's header displays multiple levels if there are two or more neighbor columns with the same non empty level key.

property G2antt.BackColorLock as Color

Retrieves or sets a value that indicates the control's background color for the locked area.

Type	Description
Color	A boolean expression that indicates the control's background color for the locked area.

The ExG2antt ActiveX Control can group the columns into two categories: locked and unlocked. The Locked category contains all the columns that are fixed to the left area of the client area. These columns cannot be scrolled horizontally. Use the CountLockedColumns to specify the number of locked columns. The unlocked are contains the columns that can be scrolled horizontally. To change the background color of the control's unlocked area use BackColor property

property G2antt.BackColorSortBar as Color

Retrieves or sets a value that indicates the sort bar's background color.

Type	Description
Color	A color expression that indicates the background color of the sort bar.

Use the BackColorSortBar property to specify the background color of the control's sort bar. Use the SortBarVisible property to show the control's sort bar. Use the BackColorSortBarCaption property to specify the background color of the caption of the sort bar. The caption of the sort bar is visible, if there are no columns in the sort bar. Use the SortBarCaption property to specify the caption of the sort bar. Use the ForeColorSortBar property to specify the foreground color of the control's sort bar. Use the BackColor property to specify the control's background color. Use the BackColorHeader property to specify the background color of the control's header bar. Use the BackColorLevelHeader property to specify the background color of the control's header bar when multiple levels are displayed.

property G2antt.BackColorSortBarCaption as Color

Returns or sets a value that indicates the caption's background color in the control's sort bar.

Type	Description
Color	A color expression that indicates the caption's background color in the control's sort bar.

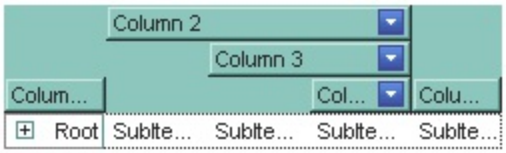
Use the SortBarCaption property to specify the caption of the sort bar, when the control's sort bar contains no columns. Use the BackColorSortBar property to specify the background color of the control's sort bar. Use the ForeColorSortBar property to specify the foreground color of the caption in the control's sort bar.

property G2antt.Background(Part as BackgroundPartEnum) as Color

Returns or sets a value that indicates the background color for parts in the control.

Type	Description
Part as BackgroundPartEnum	A BackgroundPartEnum expression that indicates a part in the control.
Color	A Color expression that indicates the background color for a specified part. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

The Background property specifies a background color or a visual appearance for specific parts in the control. If the Background property is 0, the control draws the part as default. Use the Add method to add new skins to the control. Use the Remove method to remove a specific skin from the control. Use the Clear method to remove all skins in the control. Use the BeginUpdate and EndUpdate methods to maintain performance while init the control. Use the Refresh method to refresh the control.



The following VB sample changes the visual appearance for the "drop down" filter button. The sample applies the skin "" to the "drop down" filter buttons:

```
With G2antt1
   With .VisualAppearance
      .Add &H1, App.Path + "\fbardd.ebn"
   End With
   .Background(exHeaderFilterBarButton) = &H1000000
End With
```

The following C++ sample changes the visual appearance for the "drop down" filter button:

```
#include "Appearance.h"
m_g2antt.GetVisualAppearance().Add( 0x01,
COleVariant(_T("D:\\Temp\\ExG2antt.Help\\fbardd.ebn")) );
```

```
m_g2antt.SetBackground( 0 /*exHeaderFilterBarButton*/, 0x1000000 );
```

The following VB.NET sample changes the visual appearance for the "drop down" filter button:

```
With AxG2antt1
   With .VisualAppearance
      .Add(&H1, "D:\Temp\ExG2antt.Help\fbardd.ebn")
   End With
   .set_Background(EXG2ANTTLib.BackgroundPartEnum.exHeaderFilterBarButton,
&H1000000)
End With
```

The following C# sample changes the visual appearance for the "drop down" filter button:

```
axG2antt1.VisualAppearance.Add(0x1, "D:\\Temp\\ExG2antt.Help\\fbardd.ebn");
axG2antt1.set_Background(EXG2ANTTLib.BackgroundPartEnum.exHeaderFilterBarButton,
0x1000000);
```

The following VFP sample changes the visual appearance for the "drop down" filter button:

```
With thisform.G2antt1
   With .VisualAppearance
      .Add(1, "D:\Temp\ExG2antt.Help\fbardd.ebn")
   EndWith
   .Object.Background(0) = 16777216
EndWith
```

The 16777216 value is the 0x1000000 value in hexadecimal.

method G2antt.BeginUpdate ()

Maintains performance when items are added to the control one at a time.

Type	Description

This method prevents the control from painting until the EndUpdate method is called. The BeginUpdate and EndUpdate methods increases the speed of loading your items, by preventing painting the control when it suffers any change. Once that BeginUpdate method was called, you have to make sure that EndUpdate method will be called too. You can use the FreezeEvents method to prevent the control to fire any event.

The following VB sample prevents painting the control while adding data from a database:

```
Set rs = CreateObject("ADODB.Recordset")
rs.Open "Orders", "Provider=Microsoft.Jet.OLEDB.3.51;Data Source= D:\Program
Files\Microsoft Visual Studio\VB98\NWIND.MDB", 3 ' Opens the table using static mode

G2antt1.BeginUpdate
For Each f In rs.Fields
   G2antt1.Columns.Add f.Name
Next
G2antt1.PutItems rs.GetRows()
G2antt1.EndUpdate
```

The following C++ sample prevents refreshing the control while adding columns and items from an ADODB recordset:

```
#include "Items.h"
#include "Columns.h"
#include "Column.h"

#pragma warning( disable : 4146 )
#import <msado15.dll> rename ( "EOF", "adoEOF" )
using namespace ADODB;

_RecordsetPtr spRecordset;
if ( SUCCEEDED( spRecordset.CreateInstance( "ADODB.Recordset") ) )
{
   // Builds the connection string.
```

```
   CString strTableName = "Employees", strConnection =
"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=";
   CString strPath = "D:\\Program Files\\Microsoft Visual Studio\\VB98\\NWIND.MDB";
   strConnection += strPath;
   try
   {
      // Loads the table
      if ( SUCCEEDED( spRecordset->Open(_variant_t( (LPCTSTR)strTableName ),
_variant_t((LPCTSTR)strConnection), adOpenStatic, adLockPessimistic, NULL ) ) )
      {
         m_g2antt.BeginUpdate();
         m_g2antt.SetColumnAutoResize( FALSE );
         CColumns columns = m_g2antt.GetColumns();
         long nCount = spRecordset->Fields->Count;
         if ( nCount > 0 )
         {
            // Adds the columns
            for ( long i = 0 ; i < nCount; i++ )
               columns.Add( spRecordset->Fields->Item[ i ]->Name );
            CItems items = m_g2antt.GetItems();
            // Adds the items
            while ( !spRecordset->adoEOF )
            {
               long j = 0;
               _variant_t vtI( items.AddItem( spRecordset->Fields->Item[ j ]->Value ) );
               for ( ++j ; j < nCount; j++ )
                  items.SetCellValue( vtI, _variant_t( j ), spRecordset->Fields->Item[ j ]->Value
);
               spRecordset->MoveNext();
            }
         }
         m_g2antt.EndUpdate();
      }
   }
   catch ( _com_error& e )
   {
      AfxMessageBox( e.Description() );
```

```
    }
}
```

The sample adds a column for each field in the recordset, and add a new items for each record. You can use the DataSource property to bind a recordset to the control. The #import statement imports definitions for ADODB type library, that's used to fill the control.

The following VB.NET sample prevents refreshing the control while adding columns and items:

```
With AxG2antt1
    .BeginUpdate()
    With .Columns
        .Add("Column 1")
        .Add("Column 2")
    End With
    With .Items
        Dim iNewItem As Integer
        iNewItem = .AddItem("Item 1")
        .CellValue(iNewItem, 1) = "SubItem 1"
        iNewItem = .AddItem("Item 2")
        .CellValue(iNewItem, 1) = "SubItem 2"
    End With
    .EndUpdate()
End With
```

The following C# sample prevents refreshing the control while adding columns and items:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Columns columns =axG2antt1.Columns;
columns.Add("Column 1");
columns.Add("Column 2");
EXG2ANTTLib.Items items = axG2antt1.Items;
int iNewItem = items.AddItem( "Item 1" );
items.set_CellValue( iNewItem, 1, "SubItem 1" );
items.InsertItem( iNewItem, "", "Child 1" );
iNewItem = items.AddItem( "Item 2" );
items.set_CellValue( iNewItem, 1, "SubItem 2" );
axG2antt1.EndUpdate();
```

The following VFP sample prevents refreshing the control while adding new columns and items:

```
thisform.G2antt1.BeginUpdate()
with thisform.G2antt1.Columns
    .Add("Column 1")
    .Add("Column 2")
endwith

with thisform.G2antt1.Items
    .DefaultItem = .AddItem("Item 1")
    .CellValue(0, 1) = "SubItem 1"
    .DefaultItem = .InsertItem(.DefaultItem,"","Child 1")
    .CellValue(0, 1) = "SubChild 1"
endwith
thisform.G2antt1.EndUpdate()
```

property G2antt.BorderStyle as Long

Retrieves or sets the border style of the control.

Type	Description
Long	A long expression that indicates the border style of the control.

Only for internal use.

property G2antt.CauseValidateValue as ValidateValueType

Returns or sets a value that determines whether the ValidateValue event occurs before the user changes the cell's value.

Type	Description
ValidateValueType	A ValidateValueType expression that indicates whether the ValidateValue event is fired when user leaves the focused cell or focused item.

By default, the CauseValidateValue property is exNoValidate (False). The ValidateValue event is fired only if the CauseValidateValue property is exValidateCell or exValidateItem, once the user alters the focused cell and the user is trying to leave the focused cell or item. Use the exValidateItem option to validate the item once a new item is focused, or use the exValidateCell to validate the cell's value once the user leaves the focused cell. You can use the ValidateValue event to prevent the user enters wrong values to the cells/items. In conclusion, the user can focus a new cell (in the same item) while using validation, only if the CauseValidateValue property is exValidateItem. Else, the user can not move the focus to a new cell or items until the user validates the value (Cancel parameter of the ValidateValue event is False). Call the DiscardValidateValue method to restore back the values being changed during the validation.

The following VB sample displays a message box with Yes, No and Cancel buttons to validate the changed value:

```
Private Sub G2antt1_ValidateValue(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal ColIndex As Long, ByVal NewValue As Variant, Cancel As Boolean)
  Cancel = True
  Dim iResult As Long
  i = MsgBox("Validate G2ANTT1 :" & G2antt1.Items.CellCaption(Item, ColIndex ) & " " & NewValue, vbYesNoCancel)
  If (i = vbCancel) Then
    G2antt1.DiscardValidateValue
  Else
    Cancel = i = vbNo
  End If
End Sub
```

The ValidateValue event provides the Cancel parameter which can be used to validate the value being changed. The sample calls the DiscardValidateValue method if user selects Cancel, so the value are being restored. The validation is accepted once the user selects

the Yes button. If No, the validation continues, and if the control's AutoEdit property is True, the control re-opens the editor for validation.

During the validation you may have the following order of the events:

- Edit - prevent showing the editor for specified cell.
- EditOpen - indicates that the editor for the focused cell is being opened.
- EditClose - indicates that the editor for the focused cell is being closed.
- ValidateValue - notifies your application that the value must be validated (Cancel parameter on False)
- Change - notifies the application once the user validates the newly value. In case the control is bounded to a database, the change is performed to the database too.
- Error - notifies the application for any error (for instance, if the change is not supported by the database, the Error indicates the error being issued).

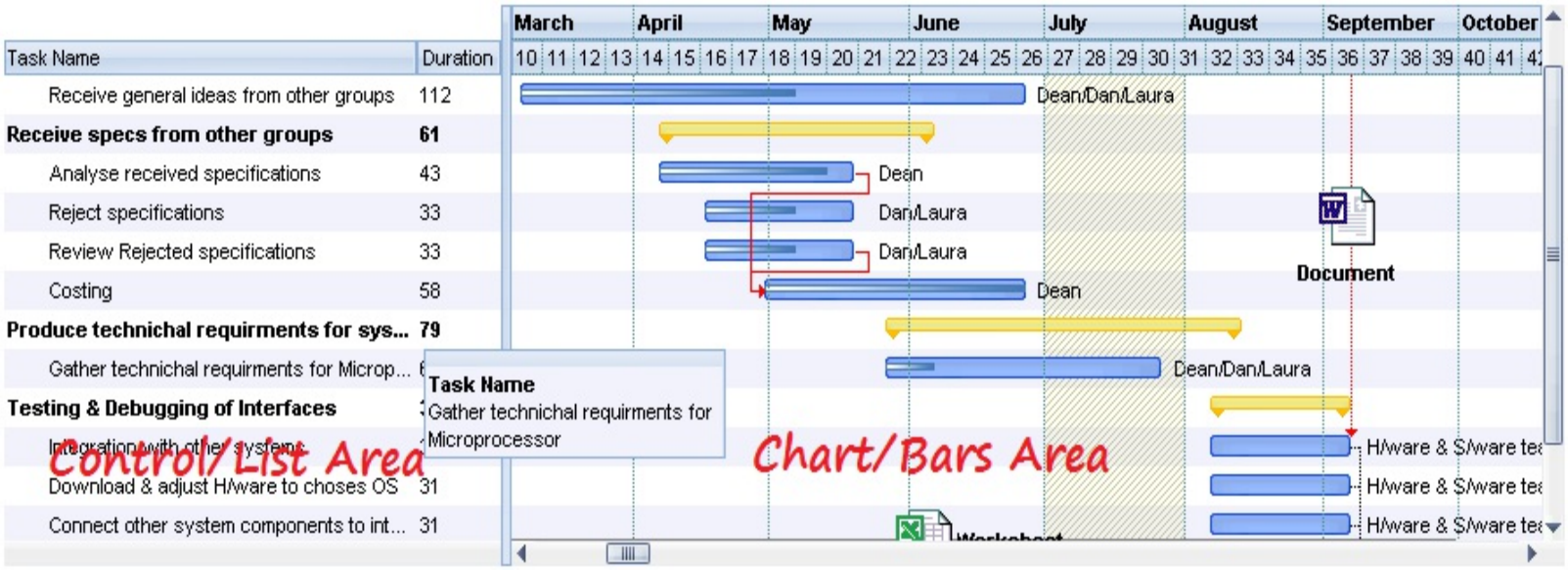
The ValidateValue event is not fired if the CellValue property is called during the event.

property G2antt.Chart as Chart

Gets the chart object.

Type	Description
Chart	A Chart object that indicates the control's chart area.

Use the Chart object to access all properties and methods related to the G2antt chart. Use the Items property to access the items in the control. Use the Columns property to access the control's Columns collection. Use the AddBar method to assign a bar to an item. Use the LevelCount property to specify the number of levels being displayed in the chart's header. Use the Level property to access the level in the chart area. Use the Bars property to access the collection of control's bars. Use the HeaderVisible property to show or hide the control's header. Use the SortBarVisible property to specify whether the control's sort bar is visible or it is hidden. Use the PaneWidth property to specify the width of the control's area or chart's area.



property G2antt.ChartOnLeft as Boolean

Specifies whether the chart area is displayed on the left or right side of the component.

Type	Description
Boolean	A booleane expression that specifies whether the chart is displayed on left or right side of the control.

By default, the ChartOnLeft property is False, so the chart area is displayed on the right side of the control. The RightToLeft property flips the order of the control's elements from right to left. Use the PaneWidth property to specify the width of the panels. The OnResizeControl property specifies which panel is getting resized when the control is resized. Use the Chart property to access the chart's properties and methods.

property G2antt.CheckImage(State as CheckStateEnum) as Long

Retrieves or sets a value that indicates the image used by cells of checkbox type.

Type	Description
State as CheckStateEnum	A CheckStateEnum expression that indicates the check's state: 0 means unchecked, 1 means checked, and 2 means partial checked.
Long	A long expression that indicates the index of image used to paint the cells of check box types. The last 7 bits in the high significant byte of the long expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part.

Use CheckImage and RadioImage properties to define icons for radio and check box cells. The CheckImage property defines the index of the icon being used by check boxes. Use the CellHasCheckBox property to assign a checkbox to a cell. Use the CellHasRadioButton property to assign a radio button to a cell. Use the CellImage or CellImages property to assign one or multiple icons to a cell. Use the CellPicture property to assign a picture to a cell. Use the CellStateChanged event to notify your application when the cell's state is changed. Use the PartialCheck property to allow partial check feature within the column.

method G2antt.ClearFilter ()

Clears the filter.

Type	Description

The method clears the Filter and FilterType properties for all columns in the control, excepts for exNumeric and exCheck values where only the Filter property is set on empty. The ApplyFilter method is automatically called when ClearFilter method is invoked. Use the FilterBarHeight property to hide the control's filter bar. Use the FilterBarCaption property to specify the caption in the control's filter bar. Use the Description property to change predefined strings in the control's filter bar. Use the FilterCriteria property to filter items using the AND, OR and NOT operators. Use the ShowFilter method to show programmatically the column's drop down filter window. The control fires the FilterChanging event just before applying the filter, and FilterChange once the list gets filtered.

property G2antt.ColumnAutoResize as Boolean

Returns or sets a value indicating whether the control will automatically size its visible columns to fit on the control's client width.

Type	Description
Boolean	A boolean expression indicating whether the control will automatically size its visible columns to fit on the control's client width.

By default, the ColumnAutoResize property is true. So, by default, the control displays no horizontal scroll bar, no matter how many visible column are. Use the ColumnAutoResize property to fit all your columns in the client area. If the ColumnAutoResize property is False, and ScrollBars property is exBoth or exHorizontal the horizontal scroll bar is shown when required. If the ScrollBars property is exDisableBoth or exDisableNoHorizontal the horizontal scroll bar is always shown. Use the ScrollBars property to show, enable or disable the control's scroll bars. Use the Width property to specify the column's width. Use the SortBarColumnWidth property to specify the column's head in the control's sort bar. By default, the ColumnAutoResize property is True.

property G2antt.ColumnFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Long

Retrieves the column from point.

Type	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Long	A long expression that indicates the column's index, or -1 if there is no column at the point. The property gets a negative value less or equal with 256, if the point is in the area between columns where the user can resize the column.

Use the ColumnFromPoint property to access the column from the point specified by the {X,Y} coordinates. The ColumnFromPoint property gets the index of the column when the cursor hovers the control's header bar. The X and Y coordinates are expressed in client coordinates, so a conversion must be done in case your coordinates are relative to the screen or to other window. **If the X parameter is -1 and Y parameter is -1 the ColumnFromPoint property determines the index of the column from the cursor.** Use the ItemFromPoint property to retrieve the item from cursor. Use the DateFromPoint property to specify the date from the cursor. The control fires the ColumnClick event when user clicks a column. Use the SortOnClick property to specify the operation that control odes when user clicks the control's header. Use the BarFromPoint property to get the bar from the point. Use the LinkFromPoint property to get the link from the point. Use the LevelFromPoint property to retrieve the index of the level from the cursor.

The following VB sample prints the caption of the column from the point:

```
Private Sub G2antt1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
  With G2antt1
    Dim c As Long
    c = .ColumnFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
    If (c >= 0) Then
      With .Columns(c)
        Debug.Print .Caption
```

```
        End With
      End If
   End With
End Sub
```

The following C++ sample prints the caption of the column from the point:

```
#include "Columns.h"
#include "Column.h"
void OnMouseMoveG2antt1(short Button, short Shift, long X, long Y)
{
   long nColIndex = m_g2antt.GetColumnFromPoint( X, Y );
   if ( nColIndex >= 0 )
   {
      CColumn column = m_g2antt.GetColumns().GetItem( COleVariant( nColIndex ) );
      OutputDebugString( column.GetCaption() );
   }
}
```

The following VB.NET sample prints the caption of the column from the point:

```
Private Sub AxG2antt1_MouseMoveEvent(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_MouseMoveEvent) Handles AxG2antt1.MouseMoveEvent
   With AxG2antt1
      Dim i As Integer = .get_ColumnFromPoint(e.x, e.y)
      If (i >= 0) Then
         With .Columns(i)
            Debug.WriteLine(.Caption)
         End With
      End If
   End With
End Sub
```

The following C# sample prints the caption of the column from the point:

```
private void axG2antt1_MouseMoveEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_MouseMoveEvent e)
{
   int i = axG2antt1.get_ColumnFromPoint( e.x,e.y );
```

```
    if ( i >= 0 )
        System.Diagnostics.Debug.WriteLine( axG2antt1.Columns[i].Caption  );
}
```

The following VFP sample prints the caption of the column from the point:

```
*** ActiveX Control Event ***
LPARAMETERS button, shift, x, y

with thisform.G2antt1
    i = .ColumnFromPoint( x, y )
    if ( i >= 0 )
        wait window nowait .Columns(i).Caption
    endif
endwith
```

property G2antt.Columns as Columns

Retrieves the control's column collection.

Type	Description
Columns	A Columns object that holds the control's columns collection.

Use the Columns property to access the Columns collection. Use the Columns collection to add, remove or change columns. Use the Add method to add a new column to the control. Use the Items property to access the control's items collection. Use the AddItem, InsertItem, InsertControlItem or PutItems method to add new items to the control. Use the DataSource property to add new columns and items to the control. Adding new items fails if the control has no columns. Use the Chart object to access all properties and methods related to the G2antt chart. Use the AddBar method to add bars to the item. Use the PaneWidth property to specify the width of the chart. Use the LevelCount property to specify the number of levels being displayed in the chart's header. Use the Level property to access the level in the chart area.

property G2antt.ColumnsAllowSizing as Boolean

Retrieves or sets a value that indicates whether a user can resize columns at run-time.

Type	Description
Boolean	A Boolean expression that indicates whether a user can resize columns at run-time.

By default, the ColumnsAllowSizing property is False. A column can be resized only if the AllowSizing property is True. Use the DrawGridLines property to show or hide the control's grid lines. Use the HeaderVisible property to show or hide the control's header bar. The HeaderAppearance property specifies the appearance of the column in the control's header bar.

property G2antt.ColumnsFloatBarSortOrder as SortOrderEnum

Specifies the sorting order for the columns being shown in the control's columns floating panel.

Type	Description
SortOrderEnum	A SortOrderEnum expression that specifies how the columns in the columns floating panel are displayed.

By default, the ColumnsFloatBarSortOrder property is SortNone. Use the ColumnsFloatBarSortOrder property to sort the columns to be displayed in the columns floating panel. The ColumnsFloatBarVisible property shows or hides the columns floating panel.

property G2antt.ColumnsFloatBarVisible as ColumnsFloatBarVisibleEnum

Retrieves or sets a value that indicates whether the the columns float bar is visible or hidden.

Type	Description
ColumnsFloatBarVisibleEnum	A ColumnsFloatBarVisibleEnum expression that specifies whether the control's Columns float-bar is visible or hidden.

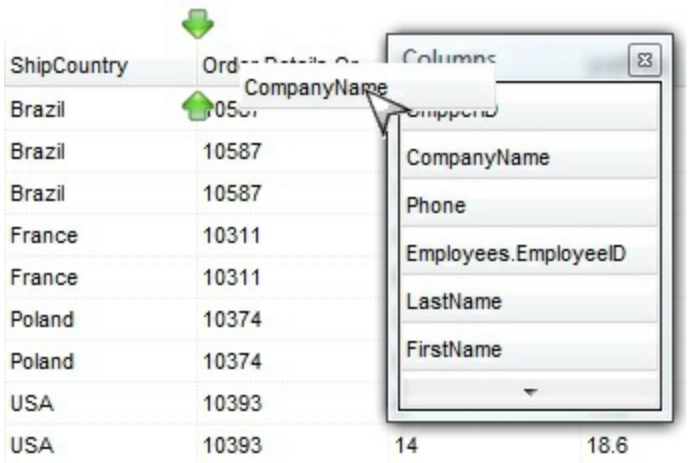
The ColumnsFloatBarVisible property indicates whether the control displays a floating panel that shows the hidden columns, so the user can drag and drop columns on order to show or hide the columns from the control. Use the ColumnsFloatBarSortOrder property to sort the columns to be displayed in the columns floating panel.

The floating panel displays the following columns:

- hidden columns, so the Visible property is False.
- drag able column, so the AllowDragging property is True.

In other words, the AllowDragging property may be used to choose if a hidden column is displayed in the floating bar. The control fires the LayoutChanged event as soon as a new column is drop on the control's header, sort or group-by bar. The Description(exColumnsFloatBar) property indicates the text to be displayed on the caption of the floating bar. The Background(exColumnsFloatAppearance) property specifies the visual appearance of the floating panel's frame.

The following screen shot shows the control's Columns float bar:



The following movies show how ColumnsFloatBarVisible works:

- The ColumnsFloatBarVisible property is used to show or hide columns by drag and

drop

- The movie shows how you can customize the visual appearance of the control's Columns floating bar

property G2antt.ConditionalFormats as ConditionalFormats

Retrieves the conditional formatting collection.

Type	Description
ConditionalFormats	A ConditionalFormats object that indicates the control's ConditionalFormats collection.

The conditional formatting feature allows you to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula. Use the Add method to format cells or items based on a formula. Use the Refresh method to refresh the control, if a change occurs in the conditional format collection. Use the CellValue property indicates the cell's caption or value.

The conditional format feature may change the cells and items as follows:

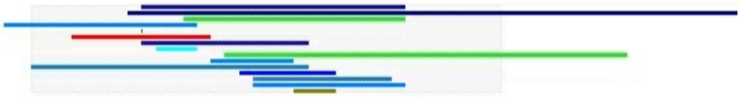
- Bold property. Bolds the cell or items
- Italic property. Indicates whether the cells or items should appear in italic.
- StrikeOut property. Indicates whether the cells or items should appear in strikeout.
- Underline property. Underlines the cells or items
- Font property. Changes the font for cells or items.
- BackColor property. Changes the background color for cells or items, supports skins as well.
- ForeColor property. Changes the foreground color for cells or items.

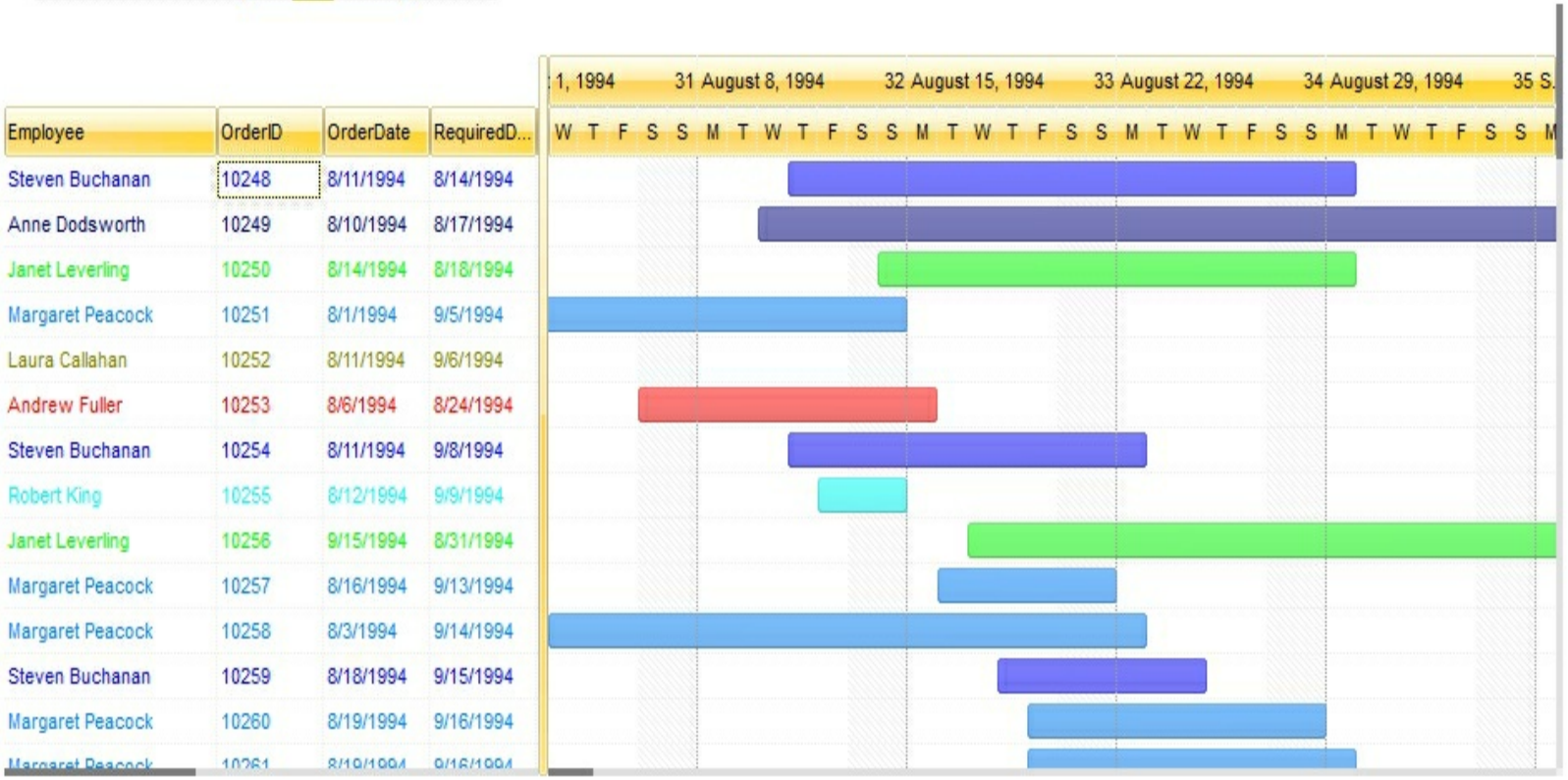
The conditional format feature may change the bars as follows:

- The BarColor property specifies the color to be applied to bars if the conditional expression is accomplished.
- The BarOverviewColor property specifies the color to be applied to bars, in the overview portion of the control, if the conditional expression is accomplished.

The ApplyTo property specifies whether the ConditionalFormat object is applied to items or to a column. The ApplyToBars property specifies the list of bars that the current format is applied to.

The following screen shot shows different colors applied to different items, using the ConditionalFormat feature:





Employee	OrderID	OrderDate	RequiredD...
Steven Buchanan	10248	8/11/1994	8/14/1994
Anne Dodsworth	10249	8/10/1994	8/17/1994
Janet Leverling	10250	8/14/1994	8/18/1994
Margaret Peacock	10251	8/1/1994	9/5/1994
Laura Callahan	10252	8/11/1994	9/6/1994
Andrew Fuller	10253	8/6/1994	8/24/1994
Steven Buchanan	10254	8/11/1994	9/8/1994
Robert King	10255	8/12/1994	9/9/1994
Janet Leverling	10256	9/15/1994	8/31/1994
Margaret Peacock	10257	8/16/1994	9/13/1994
Margaret Peacock	10258	8/3/1994	9/14/1994
Steven Buchanan	10259	8/18/1994	9/15/1994
Margaret Peacock	10260	8/19/1994	9/16/1994
Margaret Peacock	10261	8/19/1994	9/16/1994

property G2antt.ContinueColumnScroll as Boolean

Retrieves or sets a value indicating whether the control will automatically scroll the visible columns by pixel or by column width.

Type	Description
Boolean	A boolean expression indicating whether the control will automatically scroll the visible columns by pixel or by column width.

By default, the columns are scrolled pixel by pixel. Use the ContinueColumnScroll to scroll horizontally the control column by column. Use the EnsureVisibleColumn property to ensure that a visible column fits the control's client area. Use the Visible property to hide a column. The ScrollBySingleLine property retrieves or sets a value that indicates whether the control scrolls the lines to the end, item by item. Use the ScrollBars property to hide the control's scroll bars. Use the Scroll method to programmatically scroll the control's content.

method G2antt.Copy ()

Copies the control's content to the clipboard, in the EMF format.

Type	Description

Use the Copy method to copy the control's content to the clipboard, in Enhanced Metafile (EMF) format. The Enhanced Metafile format is a 32-bit format that can contain both vector information and bitmap information. This format is an improvement over the Windows Metafile Format and contains extended features, such as the following:

Built-in scaling information
Built-in descriptions that are saved with the file
Improvements in color palettes and device independence

The EMF format is an extensible format, which means that a programmer can modify the original specification to add functionality or to meet specific needs. You can paste this format to Microsoft Word, Excel, Front Page, Microsoft Image Composer and any application that know to handle EMF formats.

The Copy method copies the control's header if it's visible, and all visible items. Use the CopyTo method to copy the control's view to an PNG, BMP, GIF, TIF, JPEG, EMF file. The items are not expanded, they are listed in the order as they are displayed on the screen. Use the HeaderVisible property to show or hide the control's header. Use the ExpandItem property to expand or collapse an item. The background of the copied control is transparent.

The following VB sample saves the control's content to a EMF file, when user presses the CTRL+C key:

```
Private Sub G2antt1_KeyDown(KeyCode As Integer, Shift As Integer)
  If (KeyCode = vbKeyC) And Shift = 2 Then
    Clipboard.Clear
    G2antt1.Copy
    SavePicture Clipboard.GetData(), App.Path & "\test.emf"
  End If
End Sub
```

Now, you can open your MS Windows Word application, and you can insert the file using the Insert\Picture\From File menu, or by pressing the CTRL+V key to paste the clipboard.

The following C++ function saves the clipboard's data (EMF format) to a picture file:

```
BOOL saveEMFtoFile( LPCTSTR szFileName )
{
   BOOL bResult = FALSE;
   if ( ::OpenClipboard( NULL ) )
   {
      CComPtr spPicture;
      PICTDESC pictDesc = {0};
      pictDesc.cbSizeofstruct = sizeof(pictDesc);
      pictDesc.emf.hemf = (HENHMETAFILE)GetClipboardData( CF_ENHMETAFILE );
      pictDesc.picType = PICTYPE_ENHMETAFILE;
      if ( SUCCEEDED( OleCreatePictureIndirect( &pictDesc;, lID_IPicture, FALSE,
(LPVOID*)&spPicture; ) ) )
      {
         HGLOBAL hGlobal = NULL;
         CComPtr spStream;
         if ( SUCCEEDED( CreateStreamOnHGlobal( hGlobal = GlobalAlloc( GPTR, 0 ), TRUE,
&spStream; ) ) )
         {
            long dwSize = NULL;
            if ( SUCCEEDED( spPicture->SaveAsFile( spStream, TRUE, &dwSize; ) ) )
            {
               USES_CONVERSION;
               HANDLE hFile = CreateFile( szFileName, GENERIC_WRITE, NULL, NULL,
CREATE_ALWAYS, NULL, NULL );
               if ( hFile != INVALID_HANDLE_VALUE )
               {
                  LARGE_INTEGER l = {NULL};
                  spStream->Seek(l, STREAM_SEEK_SET, NULL);
                  long dwWritten = NULL;
                  while ( dwWritten < dwSize )
                  {
                     unsigned long dwRead = NULL;
                     BYTE b[10240] = {0};
                     spStream->Read( &b;, 10240, &dwRead; );
                     DWORD dwBWritten = NULL;
                     WriteFile( hFile, b, dwRead, &dwBWritten;, NULL );
                     dwWritten += dwBWritten;
```

```
                }
                CloseHandle( hFile );
                bResult = TRUE;
            }
        }
    }
    CloseClipboard();
  }
  return bResult;
}
```

The following VB.NET sample copies the control's content to the clipboard (open the mspaint application and paste the clipboard, after running the following code):

```
Clipboard.Clear()
With AxG2antt1
   .Copy()
End With
```

The following C# sample copies the control's content to a file (open the mspaint application and paste the clipboard, after running the following code):

```
Clipboard.Clear;
axG2antt1.Copy();
```

property G2antt.CopyTo (File as String) as Variant

Exports the control's view to an EMF file.

Type	Description
File as String	A String expression that indicates the name of the file to be saved. If present, the CopyTo property retrieves True, if the operation succeeded, else False it is failed. If the File parameter is missing or empty, the CopyTo property retrieves an one dimension safe array of bytes that contains the EMF content. If the File parameter is not empty, the extension (characters after last dot) determines the graphical/ format of the file to be saved as follows: • ***.bmp *.dib *.rle**, saves the control's content in **BMP** format. • ***.jpg *.jpe *.jpeg *.jfif**, saves the control's content in **JPEG** format. • ***.gif**, , saves the control's content in **GIF** format. • ***.tif *.tiff**, saves the control's content in **TIFF** format. • ***.png**, saves the control's content in **PNG** format. • ***.pdf**, saves the control's content to PDF format. The File argument may carry up to 4 parameters separated by the | character in the following order: ***filename.pdf | paper size | margins | options***. In other words, you can specify the file name of the PDF document, the paper size, the margins and options to build the PDF document. By default, the paper size is 210 **mm** × 297 **mm** (A4 format) and the margins are 12.7 **mm** 12.7 **mm** 12.7 **mm** 12.7 **mm**. The units for the paper size and margins can be **pt** for PostScript Points, **mm** for Millimeters, **cm** for Centimeters, **in** for Inches and **px** for pixels. If PostScript Points are used if unit is missing. For instance, 8.27 in x 11.69 in, indicates the size of the paper in inches. Currently, the options can be **single**, which indicates that the control's content is exported to a single PDF page. For instance, the CopyTo("shot.pdf|33.11 in x 46.81 in|0 0 0 0|single") exports the control's content to an A0 single PDF page, with no margins. • ***.emf** or any other extension determines the control to

	save the control's content in **EMF** format. For instance, the CopyTo("c:\temp\snapshot.png") property saves the control's content in PNG format to snapshot.png file.
Variant	A boolean expression that indicates whether the File was successful saved, if the File parameter is not empty, or a one dimension safe array of bytes, if the File parameter is empty string.

The CopyTo method copies/exports the control's view to BMP, PNG, JPG, GIF, TIFF, PDF or EMF graphical files, including no scroll bars. The StartPrintDate / EndPrintDate properties define the margins of the chart to copy (must be set before calling the CopyTo method) (starting from 22.0.1.5). Use the Copy method to copy the control's content to the clipboard.

- The **BMP** file format, also known as bitmap image file or device independent bitmap (DIB) file format or simply a bitmap, is a raster graphics image file format used to store bitmap digital images, independently of the display device (such as a graphics adapter)
- The **JPEG** file format (seen most often with the .jpg extension) is a commonly used method of lossy compression for digital images, particularly for those images produced by digital photography.
- The **GIF** (Graphics Interchange Format) is a bitmap image format that was introduced by CompuServe in 1987 and has since come into widespread usage on the World Wide Web due to its wide support and portability.
- The **TIFF** (Tagged Image File Format) is a computer file format for storing raster graphics images, popular among graphic artists, the publishing industry, and both amateur and professional photographers in general.
- The **PNG** (Portable Network Graphics) is a raster graphics file format that supports lossless data compression. PNG was created as an improved, non-patented replacement for Graphics Interchange Format (GIF), and is the most used lossless image compression format on the Internet
- The **PDF** (Portable Document Format) is a file format used to present documents in a manner independent of application software, hardware, and operating systems. Each PDF file encapsulates a complete description of a fixed-layout flat document, including the text, fonts, graphics, and other information needed to display it.
- The EMF (Enhanced Metafile Format) is a 32-bit format that can contain both vector information and bitmap information. This format is an improvement over the Windows Metafile Format and contains extended features, such as the following

  Built-in scaling information

Built-in descriptions that are saved with the file
Improvements in color palettes and device independence

The EMF format is an extensible format, which means that a programmer can modify the original specification to add functionality or to meet specific needs. You can paste this format to Microsoft Word, Excel, Front Page, Microsoft Image Composer and any application that know to handle EMF formats.

The following VB sample saves the control's content to a EMF file:

```
If (Control.CopyTo("c:\temp\test.emf")) Then
   MsgBox "test.emf file created, open it using the mspaint editor."
End If
```

The following VB sample prints the EMF content (as bytes, File parameter is empty string):

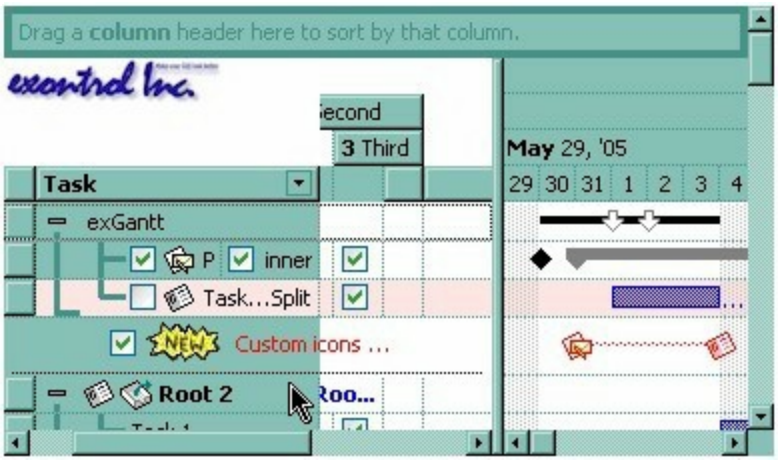
```
Dim i As Variant
For Each i In Control.CopyTo("")
   Debug.Print i
Next
```

property G2antt.CountLockedColumns as Long

Retrieves or sets a value indicating the number of locked columns. A locked column is not scrollable.

Type	Description
Long	A long expression indicating the number of locked columns.

The ExG2antt ActiveX Control can group the columns into two categories: locked and unlocked. The Locked category contains all the columns that are fixed to the left area of the client area. These columns cannot be scrolled horizontally. Use the CountLockedColumns to specify the number of locked columns. The unlocked are contains the columns that can be scrolled horizontally. Use the BackColorLock property to change the control's background color for the locked area. Use the LockedItemCount property to add or remove items locked (fixed) to the top or bottom side of the control.



property G2antt.DataSource as Object

Retrieves or sets a value that indicates the data source for object.

Type	Description
Object	An ADO or DAO Recordset object used to fill data from.

The **/COM** version provides ADO, ADODB and DAO database support. The DataSource property takes a recordset and add a column for each field found, and add a new item for each record in the recordset. Use the Visible property to hide a column. Use the CellValue property to retrieves the value of the cell. Use the PutItems to load an array to the control. Use the DetectAddNew property to allow adding new items to the control when the user adds new records to the table that's linked with the control. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula. Use the DefaultItemHeight property before setting a DataSource property to specify the

The **/NET** version provides the following methods for data binding:

- *DataSource*, gets or sets the data source that the control is displaying data for. By default, this property is empty object. The DataSource property can be: DataTable, DataView, DataSet, DataViewManager, any component that implements the IListSource interface, or any component that implements the IList interface.
- *DataMember*, indicates a sub-list of the DataSource to show in the control. By default, this property is "". For instance, if DataSource property is a DataSet, the DataMember should indicates the name of the table to be loaded.
- *DataTaskStart*, The *DataTaskStart* property gets or sets the specific field in the data source to indicate the starting point of each added task. If missing or empty, no tasks are loaded during binding. In other words, it indicates the field to use be used as the starting point for each task in any record. This member is automatically filled with the first DATE field from the DataSource, when it is set. This member is automatically filled with the first DATE field from the data source (DataSource/DataMember).
- *DataTaskEnd*, DataTaskEnd property gets or sets the specific field in the data source to indicate the ending point of each added task. If missing or empty, no tasks are loaded during binding. If the DataTaskEnd points to a DateTime object, it indicates the ending date of the newly bar, else, it indicates the duration of the task to be added. If the DataTaskEnd is equal with DataTaskBegin, a one-day task is added for each record found, during binding. This member is automatically filled with the second DATE field from the DataSource collection. *This member can be of DATE type, which indicates the exBarEnd property of any bar in the collection, or a DOUBLE, when it indicates the length/duration of the bar to be added.*

Using the data binding on /NET may change the following properties:

- Items.AllowCellValueToItemBar on True
- Column(Start).Def(exCellValueToItemBarProperty) on exBarStart
- Column(End).Def(exCellValueToItemBarProperty) on exBarEnd, exBarStart or exBarDuration (if DOUBLE field is found)

Click here to watch a movie on how to assign a data source to the control, in design mode, for /NET assembly.

The following VB sample binds an ADO recordset to the ExG2antt/COM control:

```
Set rs = CreateObject("ADODB.Recordset")
rs.Open "Orders", "Provider=Microsoft.Jet.OLEDB.3.51;Data Source= D:\Program
Files\Microsoft Visual Studio\VB98\NWIND.MDB", 3 ' Opens the table using static mode

Set G2antt1.DataSource = rs
```

The DataSource clears the columns collection, and load the recordset to the control. Use SetParent method to make your list a hierarchy.

The following C++ sample binds a table to the control:

```
#include "Items.h"
#include "Columns.h"
#include "Column.h"

#pragma warning( disable : 4146 )
#import <msado15.dll> rename ( "EOF", "adoEOF" )
using namespace ADODB;

_RecordsetPtr spRecordset;
if ( SUCCEEDED( spRecordset.CreateInstance( "ADODB.Recordset") ) )
{
   // Builds the connection string.
   CString strTableName = "Employees", strConnection =
"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=";
   CString strPath = "D:\\Program Files\\Microsoft Visual Studio\\VB98\\NWIND.MDB";
   strConnection += strPath;
   try
   {
      // Loads the table
```

```
      if ( SUCCEEDED( spRecordset->Open(_variant_t( (LPCTSTR)strTableName ),
_variant_t((LPCTSTR)strConnection), adOpenStatic, adLockPessimistic, NULL ) ) )
      {
         m_g2antt.BeginUpdate();
         m_g2antt.SetColumnAutoResize( FALSE );
         m_g2antt.SetDataSource( spRecordset );
         m_g2antt.EndUpdate();
      }
   }
   catch ( _com_error& e )
   {
      AfxMessageBox( e.Description() );
   }
}
```

The #import statement imports definitions for ADODB type library, that's used to fill the control.

The following C# sample binds at runtime a table to the /NET component, using the DataSource property:

```
private DataTable GetTable()
{
   DataTable table = new DataTable();

   table.Columns.Add("Dosage", typeof(int));
   table.Columns.Add("Drug", typeof(String));
   table.Columns.Add("Patient", typeof(String));
   table.Columns.Add("DateIn", typeof(DateTime));
   table.Columns.Add("DateOut", typeof(DateTime));

   table.Rows.Add(25, "Indocin", "David", DateTime.Today, DateTime.Today.AddDays(2));
   table.Rows.Add(50, "Enebrel", "Sam", DateTime.Today.AddDays(2),
DateTime.Today.AddDays(4));
   table.Rows.Add(10, "Hydralazine", "Christoff", DateTime.Today.AddDays(4),
DateTime.Today.AddDays(8));
   table.Rows.Add(21, "Combivent", "Janet", DateTime.Today.AddDays(2),
DateTime.Today.AddDays(6));
```

```
    table.Rows.Add(100, "Dilantin", "Melanie", DateTime.Today.AddDays(1),
DateTime.Today.AddDays(3));

    return table;

}

private void Form1_Load(object sender, EventArgs e)
{
    Exgantt1.BeginUpdate();
    Exgantt1.DefaultItemHeight = 25;
    Exgantt1.DataSource = GetTable();
    Exgantt1.Chart.LevelCount = 2;
    Exgantt1.EndUpdate();
}
```

The following VB.NET sample binds at runtime a table to the /NET component, using the DataSource property:

```
Function GetTable() As DataTable

    Dim table As DataTable = New DataTable()

    table.Columns.Add("Dosage", GetType(Long))
    table.Columns.Add("Drug", GetType(String))
    table.Columns.Add("Patient", GetType(String))
    table.Columns.Add("DateIn", GetType(DateTime))
    table.Columns.Add("DateOut", GetType(DateTime))

    table.Rows.Add(25, "Indocin", "David", DateTime.Today, DateTime.Today.AddDays(2))
    table.Rows.Add(50, "Enebrel", "Sam", DateTime.Today.AddDays(2),
DateTime.Today.AddDays(4))
    table.Rows.Add(10, "Hydralazine", "Christoff", DateTime.Today.AddDays(4),
DateTime.Today.AddDays(8))
    table.Rows.Add(21, "Combivent", "Janet", DateTime.Today.AddDays(2),
DateTime.Today.AddDays(6))
    table.Rows.Add(100, "Dilantin", "Melanie", DateTime.Today.AddDays(1),
DateTime.Today.AddDays(3))
```

```
    GetTable = table
End Function


Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    With Exgantt1
        .BeginUpdate()
        .DefaultItemHeight = 25
        .DataSource = GetTable()
        .Chart.LevelCount = 2
        .EndUpdate()
    End With
End Sub
```

property G2antt.Debug as Boolean

Displays debug information.

Type	Description
Boolean	A Boolean expression that specifies whether debug information is displayed.

By default, the Debug property is False.

property G2antt.DefaultEditorOption(Name as EditorOptionEnum) as Variant

Specifies a default option for an editor.

Type	Description
Name as EditorOptionEnum	Specifies the name of the editor whose option is being changed.
Variant	A Variant expression that indicates the newly value for the option.

Use the DefaultEditorOption property to specify default option for the editors of a specified type. The DefaultEditorOption property affects the editors that are created after the calling the DefaultEditorOption method. The DefaultEditorOption property doesn't affect the editors being already created. Use the Option property to change the option for a particular editor. For instance, you can use the DefaultEditorOption property to localize the name of the months for all date editors using the exDateMonths option.

property G2antt.DefaultItemHeight as Long

Retrieves or sets a value that indicates the default item height.

Type	Description
Long	A long expression indicates the default item height.

The DefaultItemHeight property specifies the height of the items. Changing the property fails if the control contains already items. You can change the DefaultItemHeight property at design time, or at runtime, before adding any new items to the Items collection. Use the ItemHeight property to specify the height of a specified item. Use the ScrollBySingleLine property when using the items with different heights. Use the CellSingleLine property to specify whether the cell displays the caption using multiple lines. Use the ItemAllowSizing property to specify whether the user can resize the item at runtime. Use the Height property to specify the height for a bar.

property G2antt.Description(Type as DescriptionTypeEnum) as String

Changes descriptions for control objects.

Type	Description
Type as DescriptionTypeEnum	A DescriptionTypeEnum expression that indicates the part being changed.
String	A string value that indicates the part's description.

Use the Description property to customize the captions for control filter bar window. For instance, the Description(exFilterAll) = "(Include All)" changes the "(All)" item description in the filter bar window. Use the Description property to change the predefined strings in the filter bar window.

property G2antt.DetectAddNew as Boolean

Specifies whether the control detects when a new record is added to the bounded record set.

Type	Description
Boolean	A boolean expression that indicates whether the control detects when a new record is added to the bounded recordset

By default, the DetectAddNew property is False. The DetectAddNew property detects adding new records to a recordset (/COM). Use the DataSource property to bound the control to a table. If the DetectAddNew property is True, and user adds a new record to the bounded recordset, the control automatically adds a new item to the control. The DetectAddNew property has effect only if the control is bounded to an ADO, ADODB recordset, using the DataSource property.

Handling the AddNew method in the control, using the DAO recordset on MS Access

- *Insert a Button and the Control to a form, and name them as cmdAddNew and Grid1*
- *Add the Form_Load event of the form with the following code*:

```
Private Sub Form_Load()
  With G2antt1
    .BeginUpdate
      .DataSource = CurrentDb.OpenRecordset("Employees")
      .DetectAddNew = True
    .EndUpdate
  End With
End Sub
```

The code binds the control to a DAO recordset. Please notice, that the DetectAddNew property is set after calling the DataSource method. Setting the DetectAddNew property on True, makes the control associate new items with new records added during the AddItem event as shown bellow.

- *Add the Click event of the cmdAddNew button with the following code*

```
Private Sub cmdAddNew_Click()
  With G2antt1.Items
    .EnsureVisibleItem .AddItem
```

```
  End With
End Sub
```

The code adds a new item to the control and ensures that the new item fits the control's client area. The Items.AddItem call makes the control to fire the AddItem event, which will actually add the new record to the database, as in the following code

- *Add the AddItem event of the Control with the following code:*

```
Private Sub G2antt1_AddItem(ByVal Item As Long)
   With G2antt1
      If .DetectAddNew Then
         With .DataSource
            .AddNew
            !Lastname = "new"
            !FirstName = "new"
            .Update
         End With
      End If
   End With
End Sub
```

The code adds a new record to the bounded recordset. Here you need to insert or update the required fields so the new record is added to the DAO recordset. Once the event is finished, the new item is associated with the new record in the database, so from now on, any change to the item will be reflected in the recordset.

Handling the AddNew method in the control, using the ADO recordset in VB

- *Insert a Button and the Control to a form, and name them as cmdAddNew and Grid1*
- *Add the Form_Load event of the form with the following code*:

```
Private Sub Form_Load()
   With G2antt1
      Set rs = CreateObject("ADOR.Recordset")
      With rs
         .Open "Employees", "Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=\sample.accdb", 3, 3
      End With
      .DataSource = rs
```

```
        .DetectAddNew = True
    End With
End Sub
```

The code binds the control to an ADO recordset.

- *Add the Click event of the cmdAddNew button with the following code*

```
Private Sub cmdAddNew_Click()
    With G2antt1.DataSource
        .AddNew Array("FirstName", "LastName"), Array("new", "new")
        .Update
    End With
End Sub
```

The code adds a new record to the attached recordset, and the control will add a new associated item, because the DetectAddNew method is True.

The AddItem event occurs if the AddNew (method of the ADO.RecordSet object) is performed, if the control's DetectAddNew property is True. If using the CellValue properties during the AddItem event, you must be sure that they are available, or they have the proper values or expected values. For instance, let's say that we defined the AddItem event such as:

```
Private Sub G2antt1_AddItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
    With G2antt1.Items
        .AddBar Item, "Task", .CellValue(Item, 1), .CellValue(Item, 2)
    End With
End Sub
```

If using the r.AddNew method we MUST use the values to be added as parameters of the AddNew method as in the following sample:

```
r.AddNew Array(0, 1, 2), Array("Task", #1/3/2001#, #1/4/2001#)
```

instead using the following code:

```
r.AddNew
    r(0) = "Task"
    r(1) = #1/1/2001#
    r(2) = #1/2/2001#
```

```
r.Update
```

which is wrong as the AddItem event is called when the r.AddNew method is performed, and so during the AddItem event, the values for the cells are NOT yet available, as the r(0), r(1), r(2) are filled later then r.AddNew call.

property G2antt.DetectDelete as Boolean

Specifies whether the control detects when a record is deleted from the bounded recordset.

Type	Description
Boolean	A boolean expression that indicates whether the control detects when a record is deleted from the bounded recordset.

By default, the DetectDelete property is False. The property has effect only if the DataSource property points to an ADO recordset (/COM). If the DetectDelete property is True, the control is notified when a record is deleted, and the associated item is removed from the control's items collection.

Handling the Delete method in the control, using the DAO recordset on MS Access

- *Insert a Button and the Control to a form, and name them as cmdRemove and G2antt1*
- *Add the Form_Load event of the form with the following code:*

```
Private Sub Form_Load()
   With G2antt1
      .BeginUpdate
         .DataSource = CurrentDb.OpenRecordset("Employees")
         .DetectDelete = True
      .EndUpdate
   End With
End Sub
```

The code binds the control to a DAO recordset. The DetectDelete property on True, makes the control to move the current record on the item to be deleted, and to remove any reference to the record to be deleted.

- *Add the Click event of the cmdRemove button with the following code*

```
Private Sub cmdRemove_Click()
   With G2antt1.Items
      .RemoveItem .FocusItem
   End With
End Sub
```

The code removes the focused item. The Items.RemoveItem call makes the control to fire the RemoveItem event, which will actually delete the associated record in the database, as in the following code

- *Add the RemoveItem event of the Control with the following code:*

```
Private Sub G2antt1_RemoveItem(ByVal Item As Long)
   With G2antt1
      If .DetectDelete Then
         With .DataSource
            .Delete
         End With
      End If
   End With
End Sub
```

The code deletes the current record.

Handling the Delete method in the control, using the ADO recordset in VB

- *Insert a Button and the Control to a form, and name them as cmdRemove and G2antt1*
- *Add the Form_Load event of the form with the following code:*

```
Private Sub Form_Load()
   With G2antt1
      Set rs = CreateObject("ADOR.Recordset")
      With rs
         .Open "Employees", "Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=\sample.accdb", 3, 3
      End With
      .DataSource = rs
      .DetectDelete = True
   End With
End Sub
```

The code binds the control to an ADO recordset.

- *Add the Click event of the cmdRemove button with the following code*

```
Private Sub cmdRemove_Click()
   With G2antt1.DataSource
      .Delete
   End With
End Sub
```

The Delete method of the recordset removes the current record (select a new item to the control, and the current record is changed), and due DetectDelete the associated item is removed from the view.

method G2antt.DiscardValidateValue ()

Cancels the current validation process, and restores back the modified cells.

Type	Description

The DiscardValidateValue method has effect only if the CauseValidateValue property is not zero. The DiscardValidateValue method restores the values for modified cell during the validation. For instance, pressing the Cancel button during the ValidateValue event can restore the values for modified cells, using the DiscardValidateValue method. The DiscardValidateValue automatically closes the current editor. The EditClose method can be used to programmatically closes the focused editor.

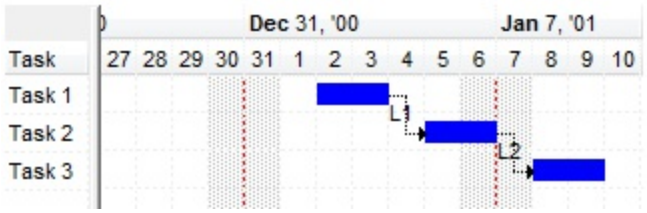
property G2antt.DrawGridLines as GridLinesEnum

Retrieves or sets a value that indicates whether the grid lines are visible or hidden.

Type	Description
GridLinesEnum	A GridLinesEnum expression that indicates whether the grid lines are visible or hidden.

Use the DrawGridLines property to add grid lines to the items list view. Use the GridLineColor property to specify the color for grid lines. Use the LinesAtRoot property specifies whether the control links the root items of the control. Use the HasLines property to specify whether the control draws the link between child items to their corresponding parent item. Use the DrawLevelSeperator property to draw lines between levels inside the chart's header. Use the DrawTickLines property to specify whether the grid lines between time units in the level are visible or hidden. Use the DrawGridLines property to specify whether the control draws the grid lines in the chart's area. Use the ColumnsAllowSizing property to allow resizing the columns, when the control's header bar is not visible.

The following screen shot shows the control using different style for gridlines:



In conclusion, the following properties are related to the control's gridlines:

- DrawGridLines specifies whether the gridlines are shown in the column/list part of the control. The gridlines in the chart part of the control are handled by the Chart.DrawGridLines property.
- GridLineColor specifies the color to show the horizontal grid line, and vertical grid lines for the columns/list part of the control. The color for vertical grid lines in the chart view part is handled by the Level.GridLineColor property.
- GridLineStyle specifies the style for horizontal grid lines and vertical grid lines in the columns/list part of the control. The Level.GridLineStyle property specifies the style for vertical grid lines in the chart area.
- Chart.DrawGridLines (belongs to Chart object) indicates whether gridlines are shown in the chart view.
- Level.DrawGridLines (belongs to Level object) specifies whether the level shows vertical gridlines in the chart part of the control.
- Level.GridLineColor (belongs to Level object) indicates the color for vertical gridlines in the chart view.
- Level.GridLineStyle (belongs to Level object) specifies the style to show the vertical

gridlines in the chart part area of the control.

property G2antt.DrawPartItem as HITEM

Indicates the handle of the item where the BeforeDrawPart / AfterDrawPart event occurs.

Type	Description
HITEM	A Long expression that specifies the handle of the item that hosts the "OwnerDraw" bar.

By default, the DrawPartItem property returns 0. The DrawPartItem property is read-only. The DrawPartItem property is valid during the BeforeDrawPart / AfterDrawPart event, while the Part parameter is exOwnerDrawBar. The DrawPartKey property indicates the key of the "OwnerDraw" being painted. Use the ItemBar property to access the properties of the "OwnerDraw" bar. The CellValue property specifies the cell's value giving the handle of the item and the index/key/name of the column.

property G2antt.DrawPartKey as Variant

Specifies the key of the owner bar to be painted during BeforeDrawPart / AfterDrawPart event.

Type	Description
Variant	A VARIANT expression that specifies the key of the bar to be painted.

By default, the DrawPartKey property returns empty value. The DrawPartKey property is read-only. The DrawPartKey property is valid during the BeforeDrawPart / AfterDrawPart event, while the Part parameter is exOwnerDrawBar. The DrawPartItem property indicates the handle of the item that hosts the "OwnerDraw" being painted. Use the ItemBar property to access the properties of the "OwnerDraw" bar. The CellValue property specifies the cell's value giving the handle of the item and the index/key/name of the column.

method G2antt.Edit ([Options as Variant])

Edits the focused cell.

Type	Description
Options as Variant	Optional. If missing, the control edits the focused cell. A long expression that indicates the handle of a locked item. Use the LockedItem property to retrieve the handle of a locked/fixed item.

The Edit method starts editing the focused cell, if the cell has an editor assigned. Use the Editor property of the Column object, or CellEditor property to assign an editor to a cell. The focused cell is determined by the intersection of the focused item and the focused column. Use the FocusItem property to get the handle of the focused item. Use the FocusColumnIndex property to determine the index of the focused column. The control fires the Edit event when the edit operation is about to start. The edit operation doesn't start if the control's ReadOnly property is True, or if the cell's editor is hidden (CellEditorVisible property is False). Use the Editing property to check whether the control is in edit mode, or to get the window's handle for the built-in editor that's visible and focused. The EditClose method closes the current editor. Use the ValidateValue event to validate the values that user enters.

The edit events are fired in the following order:

1. Edit event. Prevents editing cells, before showing the cell's editor.
2. EditOpen event. The edit operation started, the cell's editor is shown. The Editing property gives the window's handle of the built-in editor being started.
3. Change event. The Change event is fired only if the user types ENTER key, or the user selects a new value from a predefined data list.
4. EditClose event. The cell's editor is hidden and closed.

If the Options parameter is missing, the control edits the focused cell. The FocusItem and FocusColumnIndex properties indicates the focused cell. If the Options parameter is present, the control edits the item that Options parameter indicates.

For instance, the following VB sample edits a locked item when the user clicks a cell:

```
Private Sub G2antt1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
  With G2antt1
    If Button = 1 Then
```

```
        Dim h As EXG2ANTTLibCtl.HITEM, c As Long, hit As
EXG2ANTTLibCtl.HitTestInfoEnum
        h = .ItemFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY, c, hit)
        If Not h = 0 Then
           If (.Items.IsItemLocked(h)) Then
              .FocusColumnIndex = c
              .Edit h
           End If
        End If
     End If
  End With
End Sub
```

Call the DoEvents (Processes all Windows messages currently in the message queue) method after Edit method to start immediately the edit operation.

The following C++ sample edits edits a locked item when the user clicks a cell:

```
void OnMouseUpG2antt1(short Button, short Shift, long X, long Y)
{
   CItems items = m_g2antt.GetItems();
   long c = 0, hit = 0, h = m_g2antt.GetItemFromPoint( X, Y, &c, &hit);
   if ( h != 0 )
      if ( items.GetIsItemLocked( h ) )
      {
         m_g2antt.SetFocusColumnIndex( c );
         m_g2antt.Edit( COleVariant( h ) );
      }
}
```

The following VB.NET sample edits edits a locked item when the user clicks a cell:

```
Private Sub AxG2antt1_MouseUpEvent(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_MouseUpEvent) Handles AxG2antt1.MouseUpEvent
   With AxG2antt1
      Dim i As Integer, c As Integer, hit As EXG2ANTTLib.HitTestInfoEnum
      i = .get_ItemFromPoint(e.x, e.y, c, hit)
      If (Not (i = 0)) Then
         If (.Items.IsItemLocked(i)) Then
```

```
            .FocusColumnIndex = c
            .Edit(i)
        End If
    End If
  End With
End Sub
```

The following C# sample edits edits a locked item when the user clicks a cell:

```
private void axG2antt1_MouseUpEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_MouseUpEvent e)
{
   int c = 0;
   EXG2ANTTLib.HitTestInfoEnum hit;
   int i = axG2antt1.get_ItemFromPoint(e.x, e.y, out c, out hit);
   if (i != 0)
   {
      if ( axG2antt1.Items.get_IsItemLocked( i ) )
         {
            axG2antt1.FocusColumnIndex = c;
            axG2antt1.Edit(i);
         }
   }
}
```

The following VFP sample edits edits a locked item when the user clicks a cell:

```
*** ActiveX Control Event ***
LPARAMETERS button, shift, x, y

local c, hit
c = 0
hit = 0
with thisform.G2antt1
   .Items.DefaultItem = .ItemFromPoint( x, y, @c, @hit )
   if ( .Items.DefaultItem <> 0 )
      if ( .Items.IsItemLocked(0) )
         .FocusColumnIndex = c
```

```
        .Object.Edit(.Items.DefaultItem)
      endif
    endif
endwith
```

method G2antt.EditClose ()

Closes the current editor.

Type	Description

The EditClose method can be used to close programmatically the currently editor. The Edit method starts editing the focused cell, if the cell has an editor assigned. Use the Editor property of the Column object, or CellEditor property to assign an editor to a cell. The focused cell is determined by the intersection of the focused item and the focused column. Use the FocusItem property to get the handle of the focused item. Use the FocusColumnIndex property to determine the index of the focused column.

property G2antt.Editing as Long

Specifies the window's handle of the built-in editor while the control is running in edit mode.

Type	Description
Long	A long expression that indicates the window's handle for the built-in editor that's focused while the control is running in the edit mode.

Use the Editing property to check whether the control is in edit mode. Use the Editing property to get the window's handle for the built-in editor while editing. Use the Edit method to start editing the focused cell. Use the EditType property to define the column's editor. Use the ReadOnly property to make the control read only. Call the EditClose method to close the current editor. The EditingText property returns the caption being shown on the editor while the control runs in edit mode. The Editing property returns a not-zero value only if called during the EditOpen, Change or EditClose event.

The edit events are fired in the following order:

1. Edit event. Prevents editing cells, before showing the cell's editor.
2. EditOpen event. The edit operation started, the cell's editor is shown. The Editing property gives the window's handle of the built-in editor being started.
3. Change event. The Change event is fired only if the user types ENTER key, or the user selects a new value from a predefined data list.
4. EditClose event. The cell's editor is hidden and closed.

The following VB sample closes the current editor if the user presses the enter key:

```
Private Sub G2antt1_KeyDown(KeyCode As Integer, Shift As Integer)
  With G2antt1
    If Not (.Editing = 0) Then
      If (KeyCode = vbKeyReturn) Then
        .EditClose
        KeyCode = 0
      End If
    End If
  End With
End Sub
```

The following C++ sample closes the editor when user hits the enter key:

```
void OnKeyDownG2antt1(short FAR* KeyCode, short Shift)
{
    if ( *KeyCode == VK_RETURN )
        if ( m_g2antt.GetEditing() != 0 )
        {
            m_g2antt.EditClose();
            *KeyCode = 0;
        }
}
```

The following C# sample closes the editor when user hits the enter key:

```
private void axG2antt1_KeyDownEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_KeyDownEvent e)
{
    if (Convert.ToUInt32(e.keyCode) == Convert.ToUInt32(Keys.Enter))
        if (axG2antt1.Editing != 0)
        {
            axG2antt1.EditClose();
            e.keyCode = 0;
        }
}
```

The following VB.NET sample closes the editor when user hits the enter key:

```
Private Sub AxG2antt1_KeyDownEvent(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_KeyDownEvent) Handles AxG2antt1.KeyDownEvent
    If (Convert.ToUInt32(e.keyCode) = Convert.ToUInt32(Keys.Enter)) Then
        With AxG2antt1
            If Not (.Editing = 0) Then
                .EditClose()
                e.keyCode = 0
            End If
        End With
    End If
End Sub
```

The following VFP sample closes the editor when user hits the enter key:

```
*** ActiveX Control Event ***
LPARAMETERS keycode, shift

if ( keycode = 13 ) &&vkReturn
   with thisform.G2antt1.Object
      if ( .Editing() != 0 )
         .EditClose()
         keycode = 0
      endif
   endwith
endif
```

If your application still requires the string that user types into an text box inside the exG2antt control. you can use the following VB trick:

```
Private Sub G2antt1_Change(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal ColIndex As Long, NewValue As Variant)
   ' Finds the text inside the text box, in case that NewValue parameter is changed to a valid data
   Debug.Print getWndText(getEditWnd(G2antt1))
End Sub

Private Function getEditWnd(ByVal g As EXG2ANTTLibCtl.G2antt) As Long
   Dim h As Long
   h = GetWindow(g.hwnd, GW_CHILD)
   While Not (h = 0)
      If (getWndClass(h) = "HolderBuiltIn") Then
            getEditWnd = GetWindow(h, GW_CHILD)
         Exit Function
      End If
      h = GetWindow(h, GW_HWNDNEXT)
   Wend
   getEditWnd = 0
End Function

Private Function getWndText(ByVal h As Long) As String
```

```
    Dim s As String
    s = Space(1024)
    GetWindowText h, s, 1024
    getWndText = To0(s)
End Function

Private Function getWndClass(ByVal h As Long) As String
    Dim s As String
    s = Space(1024)
    GetClassName h, s, 1024
    getWndClass = To0(s)
End Function

Private Function To0(ByVal s As String) As String
    To0 = Left$(s, InStr(s, Chr$(0)) - 1)
End Function
```

The sample requires the following API declarations:

```
Private Declare Function GetWindow Lib "user32" (ByVal hwnd As Long, ByVal wCmd As
Long) As Long
Private Declare Function GetWindowText Lib "user32" Alias "GetWindowTextA" (ByVal
hwnd As Long, ByVal lpString As String, ByVal cch As Long) As Long
Private Declare Function GetClassName Lib "user32" Alias "GetClassNameA" (ByVal hwnd
As Long, ByVal lpClassName As String, ByVal nMaxCount As Long) As Long
Private Const GW_CHILD = 5
Private Const GW_HWNDNEXT = 2
```

The following C++ sample displays a message box with the caption that user types inside the text box of an editor:

```
HWND getEditWnd( HWND h )
{
    TCHAR szName[1024] = _T("");
    h = GetWindow( h, GW_CHILD );
    while ( !( h == 0 ) )
    {
```

```
        GetClassName( h, szName, 1024 );
        if ( _tcscmp( _T("HolderBuiltIn"), szName ) == 0 )
            return GetWindow( h, GW_CHILD );
        h = GetWindow( h, GW_HWNDNEXT );
    }
    return 0;
}

void OnChangeG2antt1(long Item, long ColIndex, VARIANT FAR* NewValue)
{
    HWND h = getEditWnd( m_g2antt.m_hWnd );
    if ( h )
    {
        TCHAR szText[1024] = _T("");
        ::GetWindowText( h, szText, 1024 );
        ::MessageBox( NULL, szText,NULL, NULL );
    }
}
```

property G2antt.EditingText as String

Specifies the caption of the editor during editing.

Type	Description
String	A String expression that specifies the caption of the field during editing mode.

By default, the EditingText property is "". The EditingText property returns the caption being shown on the editor while the control runs in edit mode. The control is in edit mode, if the Editing property returns a not-zero value. The EditingText property has effect only if called during the EditOpen, Change or EditClose event.

The edit events are fired in the following order:

1. Edit event. Prevents editing cells, before showing the cell's editor.
2. EditOpen event. The edit operation started, the cell's editor is shown. The Editing property gives the window's handle of the built-in editor being started.
3. Change event. The Change event is fired only if the user types ENTER key, or the user selects a new value from a predefined data list.
4. EditClose event. The cell's editor is hidden and closed.

property G2antt.Enabled as Boolean

Enables or disables the control.

Type	Description
Boolean	A boolean expression that indicates whether the control is enabled or disabled.

Use the Enabled property to disable the control. Use the ForeColor property to change the control's foreground color. Use the BackColor property to change the control's background color. Use the EnableItem to disable an item. Use the CellEnabled property to disable a cell. Use the Enabled property to disable a column. Use the SelectableItem property to specify whether an user can select an item.

method G2antt.EndUpdate ()

Resumes painting the control after painting is suspended by the BeginUpdate method.

Type	Description

The BeginUpdate and EndUpdate methods increases the speed of loading your items, by preventing painting the control when it suffers any change. Once that BeginUpdate method was called, you have to make sure that EndUpdate method will be called too. You can use the FreezeEvents method to prevent the control to fire any event.

The following VB sample prevents painting the control while adding data from a database:

```
Set rs = CreateObject("ADODB.Recordset")
rs.Open "Orders", "Provider=Microsoft.Jet.OLEDB.3.51;Data Source= D:\Program
Files\Microsoft Visual Studio\VB98\NWIND.MDB", 3 ' Opens the table using static mode

G2antt1.BeginUpdate
For Each f In rs.Fields
   G2antt1.Columns.Add f.Name
Next
G2antt1.PutItems rs.GetRows()
G2antt1.EndUpdate
```

The following VC sample prevents refreshing the control while adding columns and items from an ADODB recordset:

```
#include "Items.h"
#include "Columns.h"
#include "Column.h"

#pragma warning( disable : 4146 )
#import <msado15.dll> rename ( "EOF", "adoEOF" )
using namespace ADODB;

_RecordsetPtr spRecordset;
if ( SUCCEEDED( spRecordset.CreateInstance( "ADODB.Recordset") ) )
{
   // Builds the connection string.
   CString strTableName = "Employees", strConnection =
```

```
"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=";
   CString strPath = "D:\\Program Files\\Microsoft Visual Studio\\VB98\\NWIND.MDB";
   strConnection += strPath;
   try
   {
      // Loads the table
      if ( SUCCEEDED( spRecordset->Open(_variant_t( (LPCTSTR)strTableName ),
_variant_t((LPCTSTR)strConnection), adOpenStatic, adLockPessimistic, NULL ) ) )
      {
         m_g2antt.BeginUpdate();
         m_g2antt.SetColumnAutoResize( FALSE );
         CColumns columns = m_g2antt.GetColumns();
         long nCount = spRecordset->Fields->Count;
         if ( nCount > 0 )
         {
            // Adds the columns
            for ( long i = 0 ; i < nCount; i++ )
               columns.Add( spRecordset->Fields->Item[ i ]->Name );
            CItems items = m_g2antt.GetItems();
            // Adds the items
            while ( !spRecordset->adoEOF )
            {
               long j = 0;
               _variant_t vtI( items.AddItem( spRecordset->Fields->Item[ j ]->Value ) );
               for ( ++j ; j < nCount; j++ )
                  items.SetCellValue( vtI, _variant_t( j ), spRecordset->Fields->Item[ j ]->Value
);
               spRecordset->MoveNext();
            }
         }
         m_g2antt.EndUpdate();
      }
   }
   catch ( _com_error& e )
   {
      AfxMessageBox( e.Description() );
   }
```

```
}
```

The sample adds a column for each field in the recordset, and add a new items for each record. You can use the DataSource property to bind a recordset to the control. The #import statement imports definitions for ADODB type library, that's used to fill the control.

The following VB.NET sample prevents refreshing the control while adding columns and items:

```
With AxG2antt1
   .BeginUpdate()
   With .Columns
      .Add("Column 1")
      .Add("Column 2")
   End With
   With .Items
      Dim iNewItem As Integer
      iNewItem = .AddItem("Item 1")
      .CellValue(iNewItem, 1) = "SubItem 1"
      iNewItem = .AddItem("Item 2")
      .CellValue(iNewItem, 1) = "SubItem 2"
   End With
   .EndUpdate()
End With
```

The following C# sample prevents refreshing the control while adding columns and items:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Columns columns =axG2antt1.Columns;
columns.Add("Column 1");
columns.Add("Column 2");
EXG2ANTTLib.Items items = axG2antt1.Items;
int iNewItem = items.AddItem( "Item 1" );
items.set_CellValue( iNewItem, 1, "SubItem 1" );
items.InsertItem( iNewItem, "", "Child 1" );
iNewItem = items.AddItem( "Item 2" );
items.set_CellValue( iNewItem, 1, "SubItem 2" );
axG2antt1.EndUpdate();
```

The following VFP sample prevents refreshing the control while adding new columns and items:

```
thisform.G2antt1.BeginUpdate()
with thisform.G2antt1.Columns
    .Add("Column 1")
    .Add("Column 2")
endwith

with thisform.G2antt1.Items
    .DefaultItem = .AddItem("Item 1")
    .CellValue(0, 1) = "SubItem 1"
    .DefaultItem = .InsertItem(.DefaultItem,"","Child 1")
    .CellValue(0, 1) = "SubChild 1"
endwith
thisform.G2antt1.EndUpdate()
```

property G2antt.EnsureOnSort as Boolean

Specifies whether the control ensures that the focused item fits the control's client area, when the user sorts the items.

Type	Description
Boolean	A boolean expression that indicates whether the control ensures that the focused item fits the control's client area after sorting the items.

By default, the EnsureOnSort property is True. If the EnsureOnSort property is True, the control calls the EnsureVisibleItem method to ensure that the focused item (FocusItem property) fits the control's client area, once items get sorted. Use the SortOrder property to sort a column. The SortChildren method sorts child items of an item. The EnsureOnSort property prevents scrolling of the control when child items are sorted.

method G2antt.EnsureVisibleColumn (Column as Variant)

Scrolls the control's content to ensure that the column fits the client area.

Type	Description
Column as Variant	A long expression that indicates the index of the column, a string expression that indicates the column's caption or the column's key.

The EnsureVisibleColumn method ensures that the given column fits the control's client area. The EnsureVisibleColumn method has no effect if the column is hidded. Use the Visible property to show or hide a column. Use the Position property to change the column's position. Use the EnsureVisibleItem method to ensure that an item fits the control's client area. Use the ScrollBars property to hide the control's scroll bars. Use the Scroll method to programmatically scroll the control's content.

property G2antt.EventParam(Parameter as Long) as Variant

Retrieves or sets a value that indicates the current's event parameter.

Type	Description
Parameter as Long	A long expression that indicates the index of the parameter being requested ie 0 means the first parameter, 1 means the second, and so on. If -1 is used the EventParam property retrieves the number of parameters. If -2, the EventParam gives full information about the event, such as name, identifier, and parameters. Accessing an not-existing parameter produces an OLE error, such as invalid pointer (E_POINTER)
Variant	A VARIANT expression that specifies the parameter's value.

The EventParam method is provided to allow changing the event's parameters passed by reference, even if your environment does not support changing it (uniPaas 1.5 (formerly known as eDeveloper), DBase, and so on). For instance, Unipaas event-handling logic cannot update ActiveX control variables by updating the received arguments. The EventParam(0) retrieves the value of the first parameter of the event, while the EventParam(1) = 0, changes the value of the second parameter to 0 (the operation is successfully, only if the parameter is passed by reference). The EventParam(-1) retrieves the number of the parameters of the current event.

Let's take the event "event KeyDown (**KeyCode** as Integer, ByVal Shift as Integer)", where the KeyCode parameter is passed by reference. For instance, put the KeyCode parameter on 0, and the arrow keys are disabled while the control has the focus.

In most languages you will type something like:

```
Private Sub Control1_KeyDown(KeyCode As Integer, Shift As Integer)
   KeyCode = 0
End Sub
```

In case your environment does not support events with parameters by reference, you can use a code like follows:

```
Private Sub Control1_KeyDown(KeyCode As Integer, Shift As Integer)
   Control1.EventParam(0) = 0
End Sub
```

In other words, the EventParam property provides the parameters of the current event for reading or writing access, even if your environment does not allow changing parameters by reference.

Calling the EventParam property outside of an event produces an OLE error, such as pointer invalid, as its scope was designed to be used only during events.

method G2antt.ExecuteTemplate (Template as String)

Executes a template and returns the result.

Type	Description
Template as String	A Template string being executed

Return	Description
Variant	A Variant expression that indicates the result after executing the Template.

Use the ExecuteTemplate property to returns the result of executing a template file. Use the Template property to execute a template without returning any result. Use the ExecuteTemplate property to execute code by passing instructions as a string (template string).

For instance, the following sample retrieves the beginning date (as string) for the default bar in the first visible item:

```
Debug.Print G2antt1.ExecuteTemplate("Items.ItemBar(FirstVisibleItem(),``,1)")
```

Most of our UI components provide a Template page that's accessible in design mode. No matter what programming language you are using, you can have a quick view of the component's features using the WYSWYG Template editor.

- Place the control to your form or dialog.
- Locate the Properties item, in the control's context menu, in design mode. If your environment doesn't provide a Properties item in the control's context menu, please try to locate in the Properties browser.
- Click it, and locate the Template page.
- Click the Help button. In the left side, you will see the component, in the right side, you will see a x-script code that calls methods and properties of the control.

The control's Template page helps user to initialize the control's look and feel in design mode, using the x-script language that's easy and powerful. The Template page displays the control on the left side of the page. On the right side of the Template page, a simple editor is displayed where user writes the initialization code. The control's look and feel is automatically updated as soon as the user types new instructions. The Template script is saved to the container persistence (when Apply button is pressed), and it is executed when the control is initialized at runtime. Any component that provides a WYSWYG Template page, provides a Template property. The Template property executes code from a string (template string).

The Template or x-script is composed by lines of instructions. Instructions are separated by

"\n\r" (newline characters) or ";" character. The ; character may be available only for newer versions of the components.

An x-script instruction/line can be one of the following:

- **Dim** list of variables *Declares the variables. Multiple variables are separated by commas. (Sample: Dim h, h1, h2)*
- variable **=** property(list of arguments) *Assigns the result of the property to a variable. The "variable" is the name of a declared variable. The "property" is the property name of the object in the context. The "list or arguments" may include variables or values separated by commas. (Sample: h = InsertItem(0,"New Child"))*
- property**(** list of arguments **)** = value *Changes the property. The value can be a variable, a string, a number, a boolean value or a RGB value.*
- method**(** list of arguments **)** *Invokes the method. The "list or arguments" may include variables or values separated by commas.*
- **{** *Beginning the object's context. The properties or methods called between { and } are related to the last object returned by the property prior to { declaration.*
- **}** *Ending the object's context*
- object**.** property(list of arguments)**.**property(list of arguments)…. *The .(dot) character splits the object from its property. For instance, the Columns.Add("Column1").HeaderBackColor = RGB(255,0,0), adds a new column and changes the column's header back color.*

The x-script may uses constant expressions as follow:

- *boolean* expression with possible values as *True* or *False*
- *numeric* expression may starts with 0x which indicates a hexa decimal representation, else it should starts with digit, or +/- followed by a digit, and . is the decimal separator. *Sample: 13 indicates the integer 13, or 12.45 indicates the double expression 12,45*
- *date* expression is delimited by **#** character in the format #mm/dd/yyyy hh:mm:ss#. *Sample: #31/12/1971# indicates the December 31, 1971*
- *string* expression is delimited by **"** or ` characters. If using the ` character, please make sure that it is different than ' which allows adding comments inline. *Sample: "text" indicates the string text.*

Also , the template or x-script code may support general functions as follows:

- **Me** *property indicates the original object.*
- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the R G B values for the color being specified. For instance, the following code changes the control's background color to red: BackColor = RGB(255,0,0)*
- **LoadPicture(**file**)** *property loads a picture from a file or from BASE64 encoded strings, and returns a Picture object required by the picture properties.*

- **CreateObject(**progID**)** *property creates and retrieves a single uninitialized object of the class associated with a specified program identifier.*

property G2antt.ExpandOnDblClick as Boolean

Specifies whether the item is expanded or collapsed if the user dbl clicks the item.

Type	Description
Boolean	A boolean expression that indicates whether an item is expanded on dbl click.

Use the ExpandOnDblClick property to disable expanding or collapsing items when user dbl clicks an item. By default, the ExpandOnDblClick property is True. Use the ExpandOnKeys property to specify whether the control expands or collapses a node when user presses arrow keys. The ExpandOnSearch property specifies whether the control expands nodes when incremental searching is on (AutoSearch property is different than 0) and user types characters when the control has the focus. The control fires the DblClick event when user double clicks the control. Use the ExpandItem property to programmatically expand or collapse an item.

property G2antt.ExpandOnKeys as Boolean

Specifies a value that indicates whether the control expands or collapses a node when user presses arrow keys.

Type	Description
Boolean	A boolean expression that indicates whether the control expands or collapses a node when user presses arrow keys.

Use the ExpandOnKeys property to specify whether the control expands or collapses a node when user presses arrow keys. By default, the ExpandOnKeys property is True. Use the ExpandOnDblClick property to specify whether the control expands or collapses a node when user dbl clicks a node. The ExpandOnSearch property specifies whether the control expands nodes when incremental searching is on (AutoSearch property is different than 0) and user types characters when the control has the focus. If the ExpandOnKeys property is False, the user can't expand or collapse the items using the + or - keys on the numeric keypad. Use the ExpandItem property to programmatically expand or collapse an item.

The following VB sample expands or collapses the focused item if the user presses the + or - keys on the numeric keypad, and ExpandOnKeys property is False:

```
Private Sub G2antt1_KeyDown(KeyCode As Integer, Shift As Integer)
  With G2antt1.Items
    If (KeyCode = vbKeyAdd) Then
      .ExpandItem(.FocusItem) = True
    End If
    If (KeyCode = vbKeySubtract) Then
      .ExpandItem(.FocusItem) = False
    End If
  End With
End Sub
```

The following C++ sample expands or collapses the focused item if the user presses the + or - keys on the numeric keypad, and ExpandOnKeys property is False:

```
#include "Items.h"
void OnKeyDownG2antt1(short FAR* KeyCode, short Shift)
{
  CItems items = m_g2antt.GetItems();
  switch ( *KeyCode )
```

```
    {
        case VK_ADD:
        case VK_SUBTRACT:
        {
            items.SetExpandItem( items.GetFocusItem(), *KeyCode == VK_ADD ? TRUE : FALSE
);
            break;
        }
    }
}
```

The following VB.NET sample expands or collapses the focused item if the user presses the + or - keys on the numeric keypad, and ExpandOnKeys property is False:

```
Private Sub AxG2antt1_KeyDownEvent(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_KeyDownEvent) Handles AxG2antt1.KeyDownEvent
    Select Case (e.keyCode)
        Case Keys.Add
            With AxG2antt1.Items
                .ExpandItem(.FocusItem) = True
            End With
        Case Keys.Subtract
            With AxG2antt1.Items
                .ExpandItem(.FocusItem) = False
            End With
    End Select
End Sub
```

The following C# sample expands or collapses the focused item if the user presses the + or - keys on the numeric keypad, and ExpandOnKeys property is False:

```
private void axG2antt1_KeyDownEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_KeyDownEvent e)
{
    if ( ( e.keyCode == Convert.ToInt16(Keys.Add)  ) || (e.keyCode ==
Convert.ToInt16(Keys.Subtract) ) )
        axG2antt1.Items.set_ExpandItem( axG2antt1.Items.FocusItem, e.keyCode ==
Convert.ToInt16(Keys.Add) );
```

```
}
```

The following VFP sample expands or collapses the focused item if the user presses the + or - keys on the numeric keypad, and ExpandOnKeys property is False:

```
*** ActiveX Control Event ***
LPARAMETERS keycode, shift

with thisform.G2antt1.Items
   if ( keycode = 107 )
      .DefaultItem = .FocusItem
      .ExpandItem(0) = .t.
   else
      if ( keycode = 109 )
         .ExpandItem(0) = .f.
      endif
   endif
endwith
```

property G2antt.ExpandOnSearch as Boolean

Expands items automatically while user types characters to search for a specific item.

Type	Description
Boolean	A boolean expression that indicates whether the control expands items while user types characters to search for items.

Use the ExpandOnSearch property to expand items while user types characters to search for items using incremental search feature. Use the AutoSearch property to enable or disable incremental searching feature. Use the AutoSearch property of the Column object to specify the type of incremental searching being used within the column. The ExpandOnSearch property has no effect when the AutoSearch property is False. For instance, if the ExpandOnSearch property is True, the control fires the BeforeExpandItem event for items that have the ItemHasChildren property is True, when user types characters.

method G2antt.Export ([Destination as Variant], [Options as Variant])

Exports the control's data to a CSV or HTML format.

Type	Description
Destination as Variant	A String expression that specifies the file/format to be created. The Destination parameter indicates the format to be created as follows: • "array" indicates that the Export method returns the control's data as a two-dimensional array • if "htm" or "html", the control returns the HTML format (including CSS style) • Any file-name that ends on ".htm" or ".html" creates the file with the HTML format inside • missing, empty, or any other case the Export exports the control's data in CSV format. No error occurs, if the Export method can not create the file.
Options as Variant	A String expression that specifies the options to be used when exporting the control's data, as explained bellow.

Return	Description
Variant	The result of the Export method is a: • two-dimensional array, if the Destination is "array". For instance Export("array","vis") method exports the control's data as it is displayed into a two-dimensional array (zero-based). The result includes the columns headers into the first line, while the rest of lines contains the control's visible data. For instance, Export("array", "vis")(1, 5) returns the value of the cell on the second column and fifth row. • string, that indicates the format being exported. It could be CSV or HTML format based on the Destination parameter

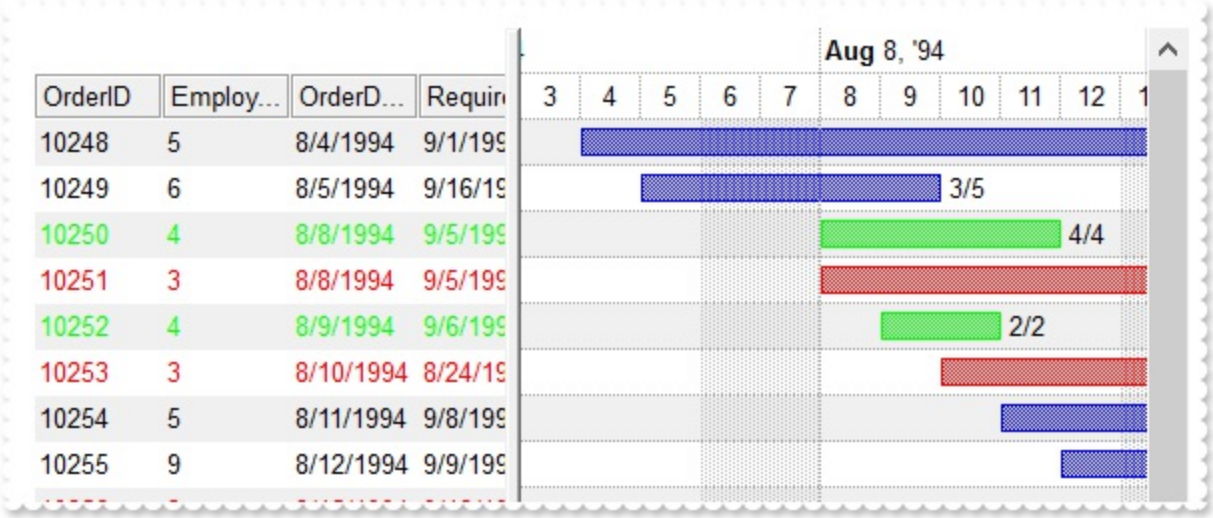
The Export method can export the control's DATA to a CSV or HTML format. The Export method can export a collection of columns from selected, visible, check or all items. By default, the control export all items, unless there is no filter applied on the control, where only visible items are exported. No visual appearance is saved in CSV format, instead the

HTML format includes the visual appearance in CSS style.

The following file samples, shows the format the Export method can export the control's DATA:

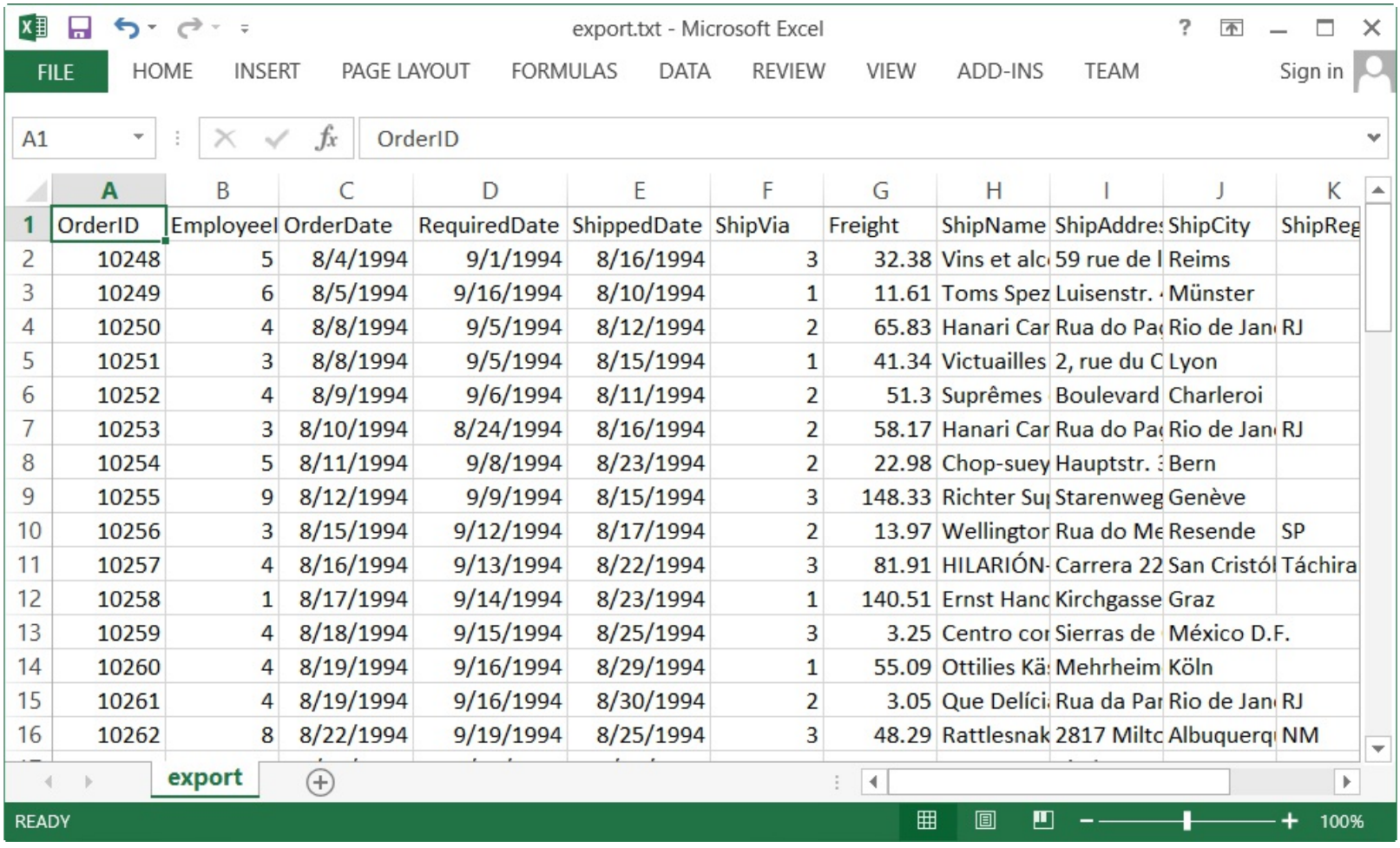
- CSV format
- HTML format

Let's say we have the following control's DATA:



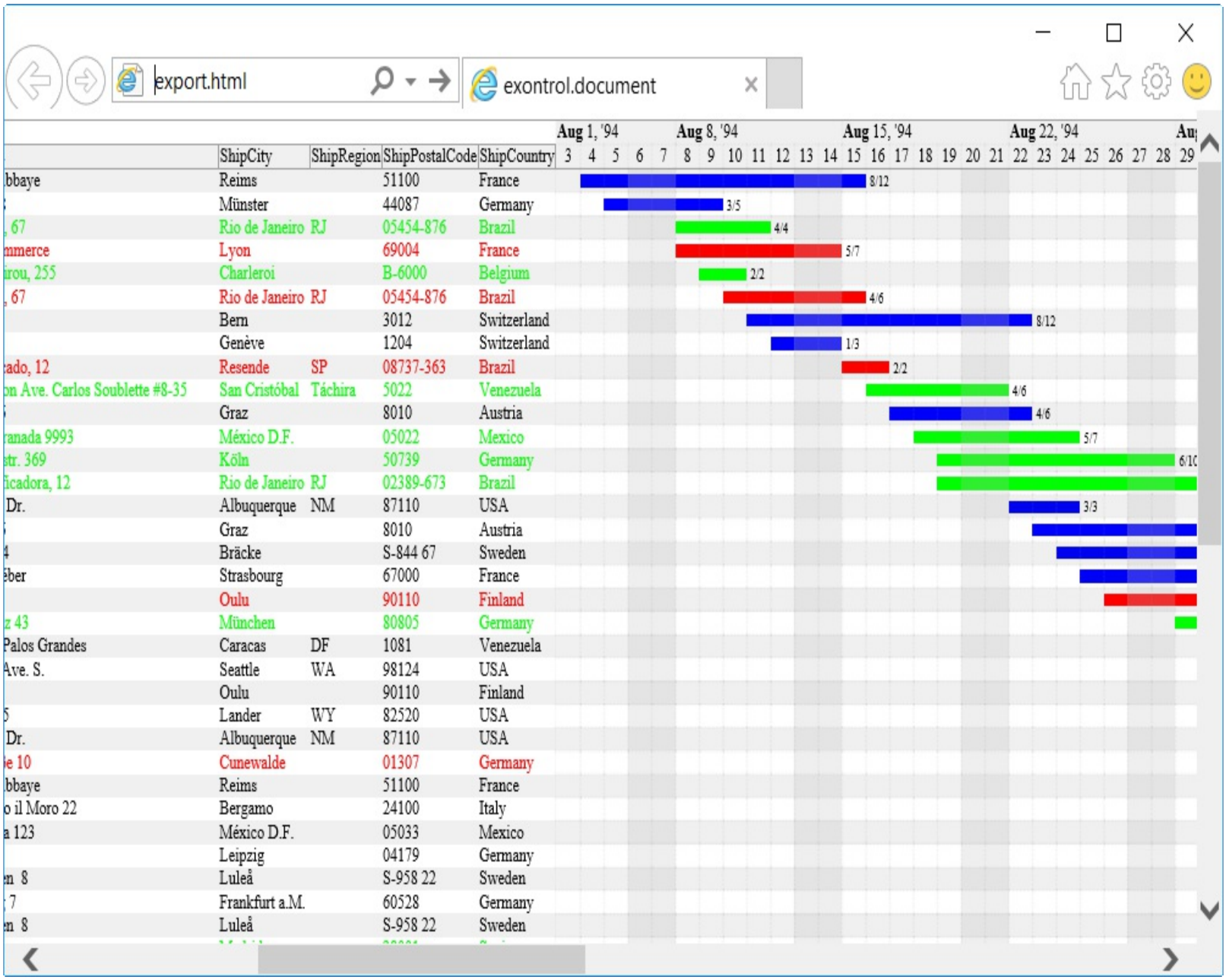
OrderID	Employ...	OrderD...	Requir
10248	5	8/4/1994	9/1/199
10249	6	8/5/1994	9/16/19
10250	4	8/8/1994	9/5/199
10251	3	8/8/1994	9/5/199
10252	4	8/9/1994	9/6/199
10253	3	8/10/1994	8/24/19
10254	5	8/11/1994	9/8/199
10255	9	8/12/1994	9/9/199

The following screen shot shows the control's DATA in CSV format:



	A	B	C	D	E	F	G	H	I	J	K
1	OrderID	EmployeeI	OrderDate	RequiredDate	ShippedDate	ShipVia	Freight	ShipName	ShipAddres	ShipCity	ShipReg
2	10248	5	8/4/1994	9/1/1994	8/16/1994	3	32.38	Vins et alc	59 rue de l	Reims	
3	10249	6	8/5/1994	9/16/1994	8/10/1994	1	11.61	Toms Spez	Luisenstr.	Münster	
4	10250	4	8/8/1994	9/5/1994	8/12/1994	2	65.83	Hanari Car	Rua do Pa	Rio de Jan	RJ
5	10251	3	8/8/1994	9/5/1994	8/15/1994	1	41.34	Victuailles	2, rue du C	Lyon	
6	10252	4	8/9/1994	9/6/1994	8/11/1994	2	51.3	Suprêmes	Boulevard	Charleroi	
7	10253	3	8/10/1994	8/24/1994	8/16/1994	2	58.17	Hanari Car	Rua do Pa	Rio de Jan	RJ
8	10254	5	8/11/1994	9/8/1994	8/23/1994	2	22.98	Chop-suey	Hauptstr.	Bern	
9	10255	9	8/12/1994	9/9/1994	8/15/1994	3	148.33	Richter Su	Starenweg	Genève	
10	10256	3	8/15/1994	9/12/1994	8/17/1994	2	13.97	Wellingtor	Rua do Me	Resende	SP
11	10257	4	8/16/1994	9/13/1994	8/22/1994	3	81.91	HILARIÓN-	Carrera 22	San Cristól	Táchira
12	10258	1	8/17/1994	9/14/1994	8/23/1994	1	140.51	Ernst Hanc	Kirchgasse	Graz	
13	10259	4	8/18/1994	9/15/1994	8/25/1994	3	3.25	Centro cor	Sierras de	México D.F.	
14	10260	4	8/19/1994	9/16/1994	8/29/1994	1	55.09	Ottilies Kä	Mehrheim	Köln	
15	10261	4	8/19/1994	9/16/1994	8/30/1994	2	3.05	Que Delíci	Rua da Par	Rio de Jan	RJ
16	10262	8	8/22/1994	9/19/1994	8/25/1994	3	48.29	Rattlesnak	2817 Milto	Albuquerq	NM

The following screen shot shows the control's DATA in HTML format:



	ShipCity	ShipRegion	ShipPostalCode	ShipCountry
bbaye	Reims		51100	France
	Münster		44087	Germany
, 67	Rio de Janeiro	RJ	05454-876	Brazil
mmerce	Lyon		69004	France
irou, 255	Charleroi		B-6000	Belgium
, 67	Rio de Janeiro	RJ	05454-876	Brazil
	Bern		3012	Switzerland
	Genève		1204	Switzerland
ado, 12	Resende	SP	08737-363	Brazil
on Ave. Carlos Soublette #8-35	San Cristóbal	Táchira	5022	Venezuela
	Graz		8010	Austria
ranada 9993	México D.F.		05022	Mexico
str. 369	Köln		50739	Germany
ficadora, 12	Rio de Janeiro	RJ	02389-673	Brazil
Dr.	Albuquerque	NM	87110	USA
	Graz		8010	Austria
4	Bräcke		S-844 67	Sweden
eber	Strasbourg		67000	France
	Oulu		90110	Finland
z 43	München		80805	Germany
Palos Grandes	Caracas	DF	1081	Venezuela
Ave. S.	Seattle	WA	98124	USA
	Oulu		90110	Finland
5	Lander	WY	82520	USA
Dr.	Albuquerque	NM	87110	USA
e 10	Cunewalde		01307	Germany
bbaye	Reims		51100	France
o il Moro 22	Bergamo		24100	Italy
a 123	México D.F.		05033	Mexico
	Leipzig		04179	Germany
n 8	Luleå		S-958 22	Sweden
7	Frankfurt a.M.		60528	Germany
n 8	Luleå		S-958 22	Sweden

The Options parameter consists a list of fields separated by | character, in the following order:

1. The first field could be **all**, **vis**, **sel** or **chk**, to export all, just visible, selected or checked items. The all option is used, if the field is missing. The **all** option displays all items, including the hidden or collapsed items. The **vis** option includes the visible items only, not including the child items of a collapsed item, or not-visible items (item's height is 0). The **sel** options lists the items being selected. The **chk** option lists all check and visible items. If chk option is used, the first column in the columns list should indicate the index of the column being queried for a check box state.
2. the second field indicates the column to be exported. All visible columns are exported, if missing. The list of columns is separated by , character, and indicates the index of the column to be shown on the exported data. The first column in the list indicates the column being queried, if the option **chk** is used.

3. the third field indicates the character to separate the fields inside each exported line [tab character-if missing]. This field is valid, only when exporting to a CSV format
4. the forth field could be **ansi** or **unicode**, which indicates the character-set to save the control's content to Destination. For instance, Export(Destination,"|||unicode") saves the control's content to destination in UNICODE format (two-bytes per character). By default, the Export method creates an ANSI file (one-byte character)

The Destination parameter indicates the file to be created where exported date should be saved. For instance, Export("c:\temp\export.html") exports the control's DATA to export.html file in HTML format, or Export("","sel|0,1|;") returns the cells from columns 0, 1 from the selected items, to a CSV format using the ; character as a field separator.

The "CSV" refers to any file that:

- CSV stands for Comma Separated Value
- is plain text using a character set such as ASCII, Unicode,
- consists of records (typically one record per line),
- with the records divided into fields separated by delimiters (typically a single reserved character such as tab, comma, or semicolon; sometimes the delimiter may include optional spaces),
- where every record has the same sequence of fields

The "HTML" refers to any file that:

- HTML stands for HyperText Markup Language.
- is plain text using a character set such as ASCII, Unicode
- It's the way web pages are encoded to handle things like bold, italics and even color text red.

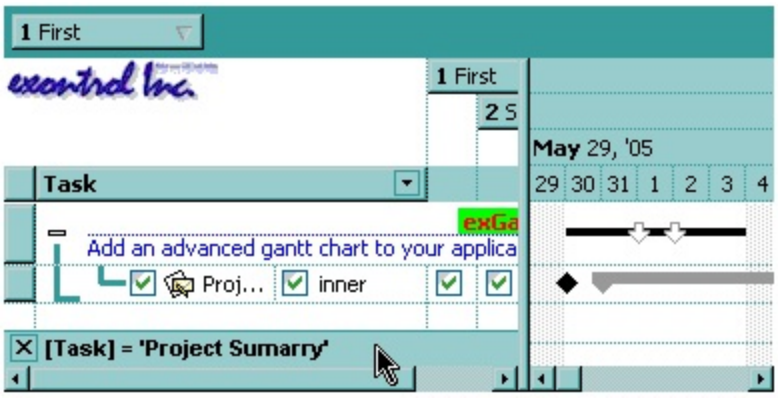
You can use the Copy/CopyTo to export the control's view to clipboard/EMF/BMP/JPG/PNG/GIF or PDF format.

property G2antt.FilterBarBackColor as Color

Specifies the background color of the control's filter bar.

Type	Description
Color	A color expression that defines the background color for description of the control's filter.

Use the FilterBarForeColor and FilterBarBackColor properties to define the colors used to paint the description for control's filter. Use the FilterBarHeight property to hide the control's filter bar header. Use the BackColor property to specify the control's background color. Use the BackColorLevelHeader property to specify the background color of the header when it displays multiple levels. Use the BackColorSortBar property to specify the background color of the control's sort bar.

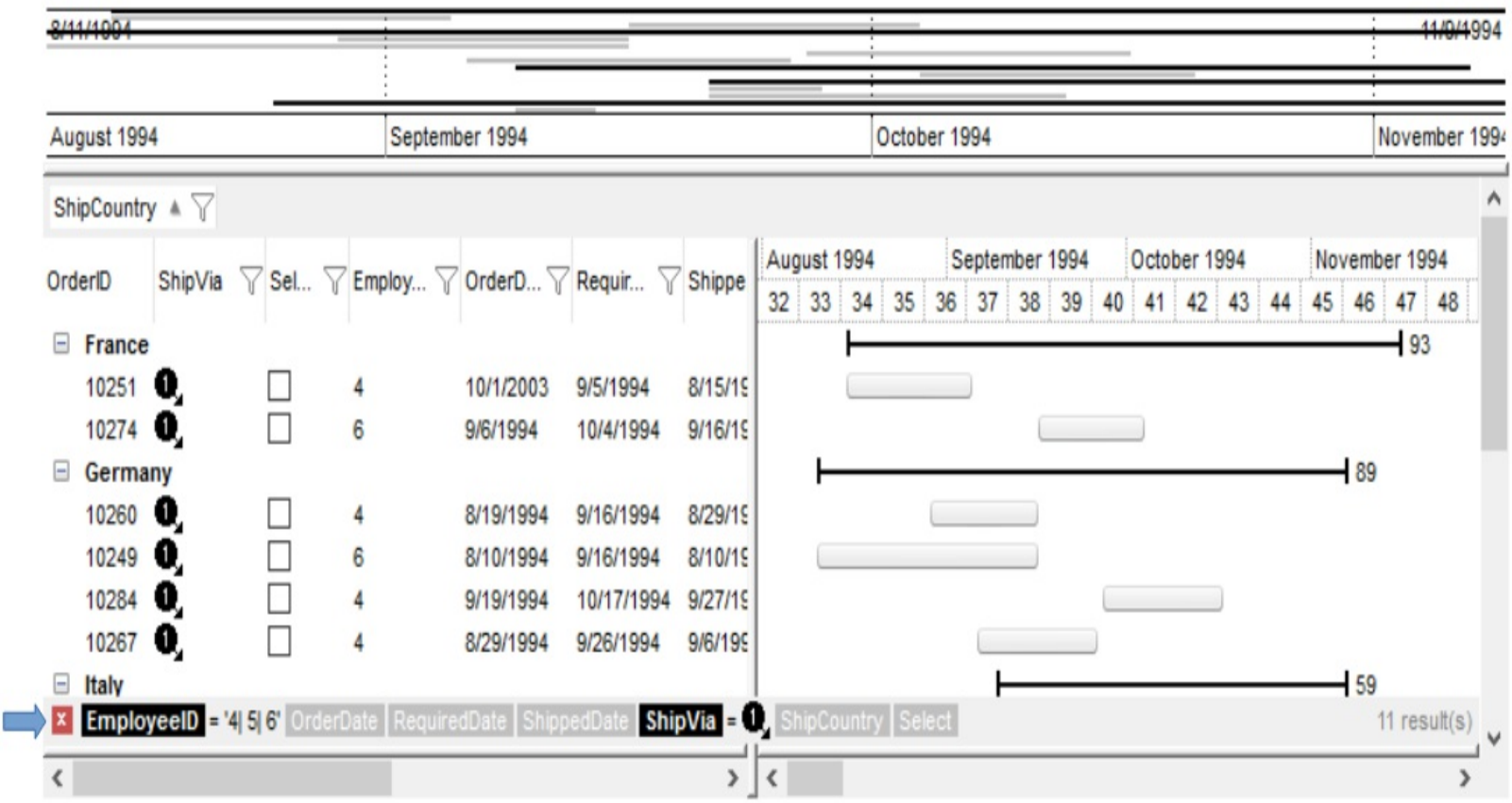


property G2antt.FilterBarCaption as String

Specifies the filter bar's caption.

Type	Description
String	A string value that defines the expression to display the control's filter bar.

By default, the FilterBarCaption property is empty. You can use the FilterBarCaption property to define the way the filter bar's caption is being displayed. The FilterBarCaption is displayed on the bottom side of the control where the control's filter bar is shown. While the FilterBarCaption property is empty, the control automatically builds the caption to be displayed on the filter bar from all columns that participates in the filter using its name and values. For instance, if the control filters items based on the columns "EmployeeID" and "ShipVia", the control's filter bar caption would appear such as "[EmployeeID] = '...' and [ShipVia] = '...'". The FilterBarCaption property supports expressions as explained bellow.

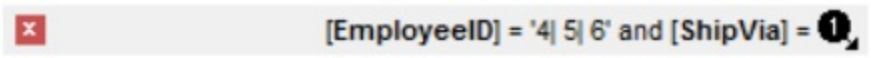


For instance:

- "no filter", shows no filter caption all the time

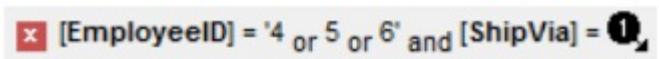
- "" displays no filter bar, if no filter is applied, else it displays the current filter

- "`<r>` + value", displays the current filter caption aligned to the right. You can include the exFilterBarShowCloseOnRight flag into the FilterBarPromptVisible property to display the close button aligned to the right



- "value replace ` and ` with `<fgcolor=FF0000> and </fgcolor>`", replace the AND keyword with a different foreground color

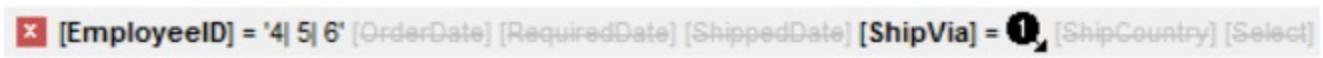
- "value replace ` and ` with `<off 4> and </off>` replace `|` with ` <off 4>or</off> ` replace ` ` with ` `", replaces the AND and | values



- "value replace `[` with `<bgcolor=000000><fgcolor=FFFFFF><b> ` replace `]` with ` </b></bgcolor></fgcolor>`", highlights the columns being filtered with a different background/foreground colors.

- "value + ` ` + available", displays the current filter, including all available columns to be filtered

- "allui" displays all available columns



- "((allui + `<fgcolor=808080>` + (matchitemcount < 0 ? ((len(allui) ? `` : ``) + `<r>` + abs(matchitemcount + 1) + ` result(s)`) : (`<r><fgcolor=808080>`+ itemcount + ` item(s)`))) replace `[<b>` with `<bgcolor=000000><fgcolor=FFFFFF><b> ` replace `</b>]` with ` </b></bgcolor></fgcolor>` replace `[<s>` with `<bgcolor=C0C0C0> <fgcolor=FFFFFF> ` replace `</s>]` with ` </bgcolor></fgcolor>`)" displays all available columns to be filtered with different background/foreground colors including the number of items/results

Use the FilterBarForeColor and FilterBarBackColor properties to define the colors used to paint the description for control's filter. Use the FilterBarHeight property to specify the height of the control's filter bar. Use the FilterBarFont property to specify the font for the control's filter bar. Use the Description property to define predefined strings in the filter bar caption. The VisibleItemCount property specifies the number of visible items in the list. The MatchItemCount property returns the number of matching items. The FilterBarPromptVisible property specifies whether how/where the control's filter/prompt is shown.

The FilterBarCaption method supports the following keywords, constants, operators and functions:

- **value** or **current** keyword returns the current filter as a string. At runtime the value may return a string such as "[EmployeeID] = '4| 5| 6' and [ShipVia] = <img>1</img>", so the control automatically applies HTML format, which you can change it. For instance, "upper(value)" displays the caption in uppercase or "value replace `<b>` with `<fgcolor=808080>` replace `</b>` with `</fgcolor>`" displays the column's name with a different foreground color.
- **itemcount** keyword returns the total number of items as indicated by ItemCount property. At runtime the itemcount is a positive integer that indicates the count of all items. For instance, "value + `<r><fgcolor=808080>Total: ` + itemcount" includes in the filter bar the number of items aligned to the right.
- **visibleitemcount** keyword returns the number of visible items as indicated by VisibleItemCount property. At runtime, the visibleitemcount is a positive integer if no filter is applied, and negative if a filter is applied. If positive, it indicates the number of visible items. The visible items does not include child items of a collapsed item. If negative, a filter is applied, and the absolute value minus one, indicates the number of visible items after filter is applied. 0 indicates no visible items, while -1 indicates that a filter is applied, but no item matches the filter criteria. For instance, "value + `<r> <fgcolor=808080>` + (visibleitemcount < 0 ? (`Result: ` + (abs(visibleitemcount) - 1)) : (`Visible: ` + visibleitemcount))" includes "Visible: " plus number of visible items, if no filter is applied or "Result: " plus number of visible items, if filter is applied, aligned to the right
- **matchitemcount** keyword returns the number of items that match the filter as indicated by MatchItemCount property. At runtime, the matchitemcount is a positive integer if no filter is applied, and negative if a filter is applied. If positive, it indicates the number of items within the control (ItemCount property). If negative, a filter is applied, and the absolute value minus one, indicates the number of matching items after filter is applied. A matching item includes its parent items, if the control's FilterInclude property allows including child items. 0 indicates no visible items, while -1 indicates that a filter is applied, but no item matches the filter criteria. For instance, "value + `<r> <fgcolor=808080>` + (matchitemcount < 0 ? (`Result: ` + (abs(matchitemcount) - 1)

) : (`Visible: ` + matchitemcount))" includes "Visible: " plus number of visible items, if no filter is applied or "Result: " plus number of macthing items, if filter is applied, aligned to the right

- **leafitemcount** keyword returns the number of leaf items. A leaf item is an item with no child items. At runtime, the leafitemcount is a positive number that computes the number of leaf items (expanded or collapsed). For instance, the "value + `<r> <fgcolor=808080><font ;6>` + leafitemcount" displays the number of leaf items aligned to the right with a different font and foreground color.
- **promptpattern** returns the pattern in the filter bar's prompt, as a string. The FilterBarPromptPattern specifies the pattern for the filter prompt. The control's filter bar prompt is visible, if the exFilterBarPromptVisible flag is included in the FilterBarPromptVisible property.
- **available** keyword returns the list of columns that are not currently part of the control's filter, but are available to be filtered. A column is available to be filtered, if the DisplayFilterButton property of the Column object, is True. At runtime, the available keyword may return a string such as "<fgcolor=C0C0C0>[<s>OrderDate</s>] <fgcolor> </fgcolor>[<s>RequiredDate</s>]<fgcolor> </fgcolor> [<s>ShippedDate</s>]<fgcolor> </fgcolor>[<s>ShipCountry</s>]<fgcolor> </fgcolor> [<s>Select</s>]</fgcolor>", so the control automatically applies HTML format, which you can change it. For instance, "value + ` ` + available", displays the current filter, including all available columns to be filtered. For instance, the "value + `<r>` + available replace `C0C0C0` with `FF0000`" displays the available columns aligned to the right with a different foreground color.
- **allui** keyword returns the list of columns that are part of the current filter and available columns to be filtered. A column is available to be filtered, if the DisplayFilterButton property of the Column object, is True. At runtime, the allui keyword may return a string such as "[EmployeeID] = '4| 5| 6'<fgcolor> </fgcolor><fgcolor=C0C0C0> [<s>OrderDate</s>]</fgcolor><fgcolor> </fgcolor><fgcolor=C0C0C0> [<s>RequiredDate</s>]</fgcolor><fgcolor> </fgcolor><fgcolor=C0C0C0> [<s>ShippedDate</s>]</fgcolor><fgcolor> </fgcolor>[ShipVia] = <img>1</img><fgcolor> </fgcolor><fgcolor=C0C0C0>[<s>ShipCountry</s>]</fgcolor> <fgcolor> </fgcolor><fgcolor=C0C0C0>[<s>Select</s>]</fgcolor>", so the control automatically applies HTML format, which you can change it. For instance, "allui", displays the current filter, including all available columns to be filtered. For instance, the "((allui + `<fgcolor=808080>` + (matchitemcount < 0 ? ((len(allui) ? `` : ``) + `<r>` + abs(matchitemcount + 1) + ` result(s)`) : (`<r><fgcolor=808080>`+ itemcount + ` item(s)`))) replace `[<b>` with `<bgcolor=000000><fgcolor=FFFFFF><b> ` replace `</b>]` with ` </b></bgcolor></fgcolor>` replace `[<s>` with `<bgcolor=C0C0C0> <fgcolor=FFFFFF> ` replace `</s>]` with ` </bgcolor></fgcolor>`)" displays all available columns to be filtered with different background/foreground colors including the number of items/results
- **all** keyword returns the list of all columns (visible or hidden) no matter if the DisplayFilterButton property is True or False. At runtime, the all keyword may return a

string such as "<fgcolor=C0C0C0>[<s>OrderID</s>]</fgcolor><fgcolor> </fgcolor> [EmployeeID] = '4| 5| 6'<fgcolor> </fgcolor><fgcolor=C0C0C0> [<s>OrderDate</s>]</fgcolor><fgcolor> </fgcolor><fgcolor=C0C0C0> [<s>RequiredDate</s>]</fgcolor><fgcolor>", so the control automatically applies HTML format, which you can change it. For instance, "all", displays the current filter, including all other columns. For instance, the "((all + `<fgcolor=808080>` + (matchitemcount < 0 ? ((len(allui) ? `` : ``) + `<r>` + abs(matchitemcount + 1) + ` result(s)`) : (`<r><fgcolor=808080>`+ itemcount + ` item(s)`))) replace `[<b>` with `<bgcolor=000000><fgcolor=FFFFFF><b> ` replace `</b>]` with ` </b></bgcolor> </fgcolor>` replace `[<s>` with `<bgcolor=C0C0C0><fgcolor=FFFFFF> ` replace `</s>]` with ` </bgcolor></fgcolor>`)" displays all columns with different background/foreground colors including the number of items/results

Also, the FilterBarCaption property supports predefined constants and operators/functions as described here.

Also, the FilterBarCaption property supports HTML format as described here:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.

  The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using <a ;exp=> or <a ;e64=> anchor tags. The exp/e64 field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption.

  - exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
  - e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.

- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{subscript}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

shadow

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:


outline anti-aliasing

property G2antt.FilterBarDropDownHeight as Double

Specifies the height of the drop down filter window proportionally with the height of the control's list.

Type	Description
Double	A double expression that indicates the height of the drop down filter window. The meaning of the value is explained bellow.

By default, the FilterBarDropDownHeight property is 0.5. It means, the height of the drop down filter window is half of the height of the control's list. Use the FilterBarDropDownHeight property to specify the height of the drop down window filter window. Use the DisplayFilterButton property to display a filter button to the column's caption. Use the FilterBarDropDownWidth property to specify the width of the drop down filter window. Use the Description property to define predefined strings in the filter bar. Use the FilterInclude property to specify whether the child items should be included to the list when the user applies the filter. Use the ShowFilter method to show programmatically the column's drop down filter window.

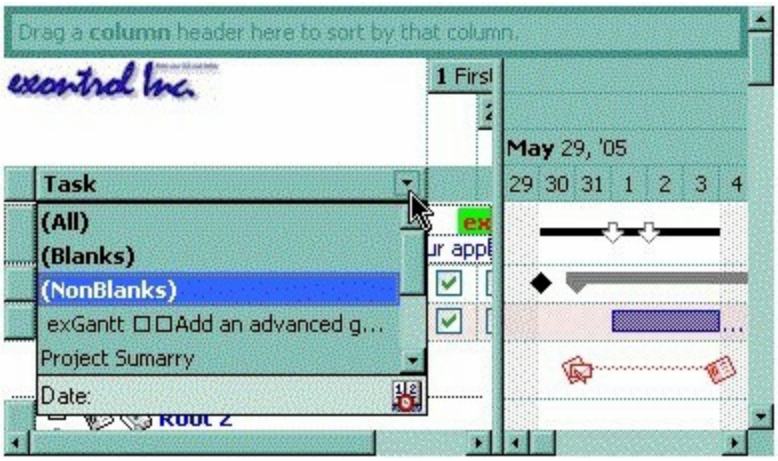
If the FilterBarDropDownHeight property is negative, the absolute value of the FilterBarDropDownHeight property indicates the height of the drop down filter window in pixels. In this case, the height of the drop down filter window is not proportionally with the height of the control's list area. For instance, the following sample specifies the height of the drop down filter window being 100 pixels:

```
With G2antt1
   .FilterBarDropDownHeight = -100
End With
```

If the FilterBarDropDownHeight property is greater than 0, it indicates the height of the drop down filter window proportionally with the height of the control's height list. For instance, the following sample specifies the height of the drop down filter window being the same with the height of the control's list area:

```
With G2antt1
   .FilterBarDropDownHeight = 1
End With
```

The drop down filter window always include an item.


Drag a column header here to sort by that column.
exontrol Inc.
1 First
May 29, '05
29 30 31 1 2 3 4
Task
(All)
(Blanks)
(NonBlanks)
exGantt ☐☐Add an advanced g...
Project Sumarry
Date:
Root 2

property G2antt.FilterBarFont as IFontDisp

Retrieves or sets the font for control's filter bar.

Type	Description
IFontDisp	A font object that indicates the font used to paint the description for control's filter

Use the FilterBarFont property to specify the font for the control's filter bar object. Use the Font property to set the control's font. Use the FilterBarHeight property to specify the height of the filter bar. Use the FilterBarCaption property to define the control's filter bar caption. Use the Refresh method to refresh the control.

property G2antt.FilterBarForeColor as Color

Specifies the foreground color of the control's filter bar.

Type	Description
Color	A color expression that defines the foreground color of the description of the control's filter.

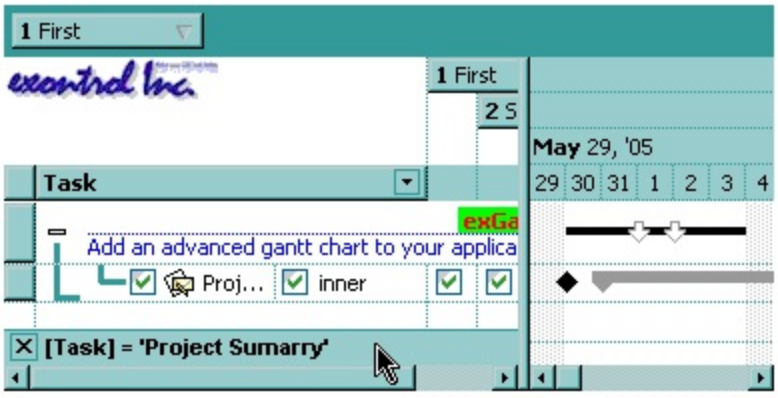
Use the FilterBarForeColor and FilterBarBackColor properties to define colors used to paint the description of the control's filter. Use the FilterBarFont property to specify the filter bar's font. Use the FilterBarCaption property to specify the caption of the control's filter bar.

property G2antt.FilterBarHeight as Long

Specifies the height of the control's filter bar. If the value is less than 0, the filter bar is automatically resized to fit its description.

Type	Description
Long	A long expression that indicates the height of the filter bar status.

The filter bar status defines the control's filter description. If the FilterBarHeight property is less than 0 the control automatically updates the height of the filter's description to fit in the control's client area. If the FilterBarHeight property is zero the filter's description is hidden. If the FilterBarHeight property is grater than zero it defines the height in pixels of the filter's description. Use the ClearFilter method to clear the control's filter. Use the FilterBarCaption property to define the control's filter bar caption. Use the FilterBarFont property to specify the font for the control's filter bar. Use the FilterBarDropDownWidth property to specify the width of the drop down filter window. Use the FilterBarDropDownHeight to specify the height of the drop down filter window. Use the ShowFilter method to show programmatically the column's drop down filter window.



property G2antt.FilterBarPrompt as String

Specifies the caption to be displayed when the filter pattern is missing.

Type	Description
String	A string expression that indicates the HTML caption being displayed in the filter bar, when filter prompt pattern is missing. The FilterBarPromptPattern property specifies the pattern to filter the list using the filter prompt feature.

By default, the FilterBarPrompt property is "<fgcolor=808080>Start Filter...</fgcolor>". The FilterBarPromptPattern property specifies the pattern to filter the list using the filter prompt feature. Changing the FilterBarPrompt property won't change the current filter. The FilterBarPromptColumns property specifies the list of columns to be used when filtering by prompt. The DisplayFilterButton property specifies whether the column's header displays a filter button. The VisibleItemCount property retrieves the number of visible items in the list. The control fires the FilterChanging event just before applying the filter, and FilterChange once the list gets filtered. Use the FilterBarCaption property to change the caption in the filter bar once a new filter is applied. The FilterBarFont property specifies the font to be used in the filter bar. The FilterBarBackColor property specifies the background color or the visual aspect of the control's filter bar. The FilterBarForeColor property specifies the foreground color or the control's filter bar.

The FilterBarPrompt property supports HTML format as described here:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.

  The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using <a ;exp=> or <a ;e64=> anchor tags. The exp/e64 field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption.

  - exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
  - e64, encodes in BASE64 the HTML text to be shown once the user clicks the

anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part

of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.

- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{subscript}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the

color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor></**sha**>*" gets:

The FilterBarPrompt property has effect only if:

- FilterBarPromptVisible property is True
- FilterBarPromptPattern property is Empty.

property G2antt.FilterBarPromptColumns as Variant

Specifies the list of columns to be used when filtering using the prompt.

Type	Description
Variant	A long expression that indicates the index of the column to apply the filter prompt, a string expression that specifies the list of columns (indexes) separated by comma to apply the filter prompt, or a safe array of long expression that specifies the indexes of the columns to apply the filter. The filter prompt feature allows you to filter the items as you type while the filter bar is visible on the bottom part of the list area.

By default, the FilterBarPromptColumns property is -1. If the FilterBarPromptColumns property is -1, the filter prompt is applied for all columns, visible or hidden. Use the FilterBarPromptColumns property to specify the list of columns to apply the filter prompt pattern. The FilterBarPromptVisible property specifies whether the filter prompt is visible or hidden. Use the FilterBarPrompt property to specify the HTML caption being displayed in the filter bar when the filter pattern is missing. The FilterBarPromptPattern property specifies the pattern to filter the list. Changing the FilterBarPromptPattern property does not require calling the ApplyFilter method to apply the new filter, only if filtering is required right a way. The FilterBarPromptType property specifies the type of filtering when the user edits the prompt in the filter bar.

property G2antt.FilterBarPromptPattern as String

Specifies the pattern for the filter prompt.

Type	Description
String	A string expression that specifies the pattern to filter the list.

By default, the FilterBarPromptPattern property is empty. If the FilterBarPromptPattern property is empty, the filter bar displays the FilterBarPrompt property, if the FilterBarPromptVisible property is True. The FilterBarPromptPattern property indicates the patter to filter the list. The pattern may include wild characters if the FilterBarPromptType property is exFilterPromptPattern. The FilterBarPromptColumns specifies the list of columns to be used when filtering. Changing the FilterBarPromptPattern property does not require calling the ApplyFilter method to apply the new filter, only if filtering is required right a way.

property G2antt.FilterBarPromptType as FilterPromptEnum

Specifies the type of the filter prompt.

Type	Description
FilterPromptEnum	A FilterPromptEnum expression that specifies how the items are being filtered.

By default, the FilterBarPromptType property is exFilterPromptContainsAll. The filter prompt feature allows you to filter the items as you type while the filter bar is visible on the bottom part of the list area. The Filter prompt feature allows at runtime filtering data on hidden columns too. Use the FilterBarPromptVisible property to show the filter prompt. Use the FilterBarPrompt property to specify the HTML caption being displayed in the filter bar when the filter pattern is missing. The FilterBarPromptPattern property specifies the pattern to filter the list. Changing the FilterBarPromptPattern property does not require calling the ApplyFilter method to apply the new filter, only if filtering is required right a way. The FilterBarPromptColumns property specifies the list of columns to be used when filtering by prompt. The DisplayFilterButton property specifies whether the column's header displays a filter button. The VisibleItemCount property retrieves the number of visible items in the list. The control fires the FilterChanging event just before applying the filter, and FilterChange once the list gets filtered. Use the FilterBarCaption property to change the caption in the filter bar once a new filter is applied.

The FilterBarPromptType property supports the following values:

- **exFilterPromptContainsAll**, The list includes the items that contains all specified sequences in the filter (FilterBarPromptPattern property). Can be combined with exFilterPromptCaseSensitive, exFilterPromptStartWords, exFilterPromptEndWords or exFilterPromptWords
- **exFilterPromptContainsAny**, The list includes the items that contains any of specified sequences in the filter (FilterBarPromptPattern property). Can be combined with exFilterPromptCaseSensitive, exFilterPromptStartWords, exFilterPromptEndWords or exFilterPromptWords
- **exFilterPromptStartWith**, The list includes the items that starts with any specified sequences in the filter (FilterBarPromptPattern property). Can be combined with exFilterPromptCaseSensitive, exFilterPromptStartWords, exFilterPromptEndWords or exFilterPromptWords
- **exFilterPromptEndWith**, The list includes the items that ends with any specified sequences in the filter (FilterBarPromptPattern property). Can be combined with exFilterPromptCaseSensitive, exFilterPromptStartWords, exFilterPromptEndWords or exFilterPromptWords
- **exFilterPromptPattern**, The filter indicates a pattern that may include wild characters to be used to filter the items in the list. The FilterBarPromptPattern property may

include wild characters as follows:

- '?' for any single character
- '*' for zero or more occurrences of any character
- '#' for any digit character
- ' ' space delimits the patterns inside the filter

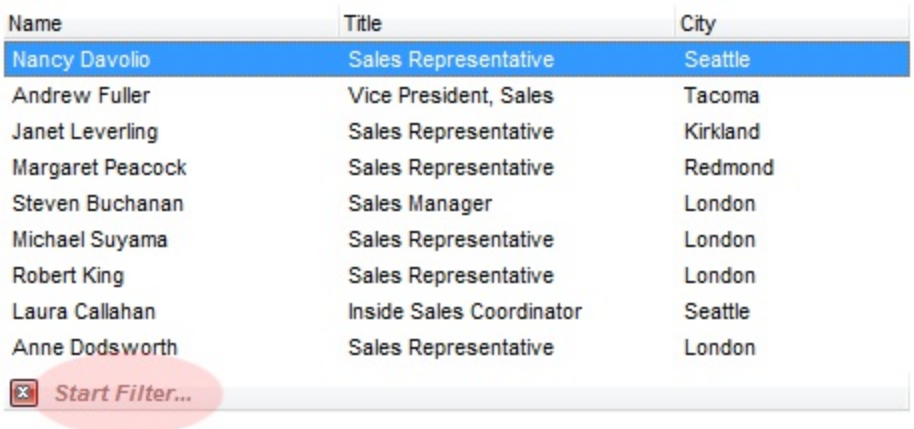
property G2antt.FilterBarPromptVisible as FilterBarVisibleEnum

Shows or hides the control's filter bar including filter prompt.

Type	Description
FilterBarVisibleEnum	A FilterBarVisibleEnum expression that defines the way the control's filter bar is shown.

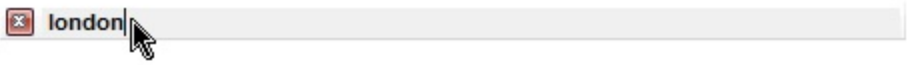
By default, The FilterBarPromptVisible property is exFilterBarHidden. The filter prompt feature allows you to filter the items as you type while the filter bar is visible on the bottom part of the list area. The Filter prompt feature allows at runtime filtering data on hidden columns too. Use the FilterBarPromptVisible property to show the filter prompt. Use the FilterBarPrompt property to specify the HTML caption being displayed in the filter bar when the filter pattern is missing. The FilterBarPromptPattern property specifies the pattern to filter the list. Changing the FilterBarPromptPattern property does not require calling the ApplyFilter method to apply the new filter, only if filtering is required right a way. The FilterBarCaption property defines the caption to be displayed on the control's filter bar. The FilterBarPromptType property specifies the type of filtering when the user edits the prompt in the filter bar. The FilterBarPromptColumns property specifies the list of columns to be used when filtering by prompt. The DisplayFilterButton property specifies whether the column's header displays a filter button. The VisibleItemCount property retrieves the number of visible items in the list. The control fires the FilterChanging event just before applying the filter, and FilterChange once the list gets filtered.

The following screen show shows the filter prompt:



The following screen show shows the list once the user types "london":

Name	Title	City
Steven Buchanan	Sales Manager	London
Michael Suyama	Sales Representative	London
Robert King	Sales Representative	London
Anne Dodsworth	Sales Representative	London



property G2antt.FilterCriteria as String

Retrieves or sets the filter criteria.

Type	Description
String	A string expression that indicates the filter criteria.

By default, the FilterCriteria property is empty. Use the FilterCriteria property to specify whether you need to filter items using OR, NOT operators between columns. If the FilterCriteria property is empty, or not valid, the filter uses the AND operator between columns. Use the FilterCriteria property to specify how the items are filtered.

The FilterCriteria property supports the following operators:

- **not** operator (unary operator)
- **and** operator (binary operator)
- **or** operator (binary operator)

Use the **(** and **)** parenthesis to define the order execution in the clause, if case. The operators are g2antted in their priority order. The **%** character precedes the index of the column (zero based), and indicates the column. For instance, **%0 or %1** means that OR operator is used when both columns are used, and that means that you can filter for values that are in a column or for values that are in the second columns. If a column is not g2antted in the FilterCriteria property, and the user filters values by that column, the AND operator is used by default. For instance, let's say that we have three columns, and FilterCriteria property is "%0 or %1". If the user filter for all columns, the filter clause is equivalent with (%0 or %1) and %2, and it means all that match the third column, and is in the first or the second column.

Use the Filter and FilterType properties to define a filter for a column. The ApplyFilter method should be called to update the control's content after changing the Filter or FilterType property, in code! Use the DisplayFilterButton property to display a drop down button to filter by a column.

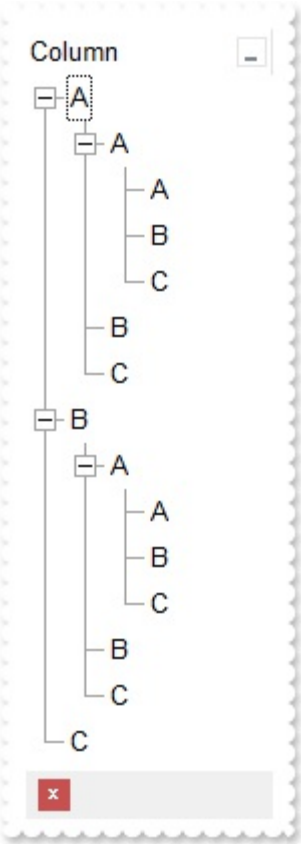
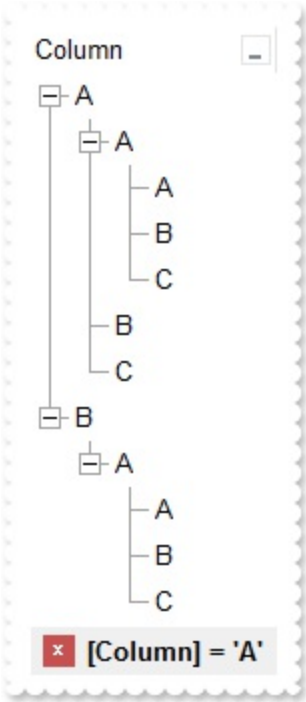
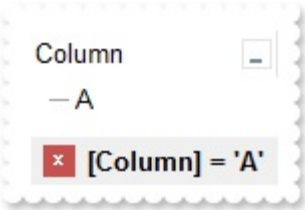
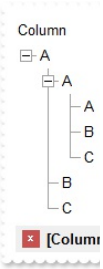
property G2antt.FilterInclude as FilterIncludeEnum

Specifies the items being included after the user applies the filter.

Type	Description
FilterIncludeEnum	A FilterIncludeEnum expression that indicates the items being included when the filter is applied.

By default, the FilterInclude property is exItemsWithoutChilds, which specifies that only items (and parent-items) that match the filter are being displayed. Use the FilterInclude property to specify whether the child- items should be displayed when the user applies the filter. Use the Filter property and FilterType property to specify the column's filter. Use the ApplyFilter to apply the filter at runtime. Use the ClearFilter method to clear the control's filter. Use the FilterCriteria property to filter items using the AND, OR and NOT operators. Use the FilterBarPromptVisible property to show the control's filter-prompt, that allows you to filter items as you type.

The following table shows items to display, when filter for "A" items, using different values for FilterInclude property:

no filter	exItemsWithoutChilds 0	exItemsWithChilds 1	exRootsWithoutChilds 2	exRootsW 3
				

property G2antt.FocusColumnIndex as Long

Specifies the index of focused column.

Type	Description
Long	A long expression that indicates the index of the focused column.

Use the FocusColumnIndex property to determine the focused column. Use the FocusItem property to determine the focused item. Use the TreeColumnIndex property to set the column that displays the hierarchy. Use the SearchColumnIndex property to set the index of the searching column. The SelectColumnInner property indicates the index of an inner cell that has the focus.

The control fires the FocusChanged event when the user changes:

- the focused item
- the focused column or an inner cell gets the focus.

property G2antt.Font as IFontDisp

Retrieves or sets the control's font.

Type	Description
IFontDisp	A Font object used to paint the items.

Use the Font property to change the control's font . Use the FilterBarFont property to assign a different font for the control's filter bar. Use the Refresh method to refresh the control. Use the BeginUpdate and EndUpdate method to maintain performance while adding new columns or items.

The following VB sample assigns by code a new font to the control:

```
With G2antt1
   With .Font
      .Name = "Tahoma"
   End With
   .Refresh
End With
```

The following C++ sample assigns by code a new font to the control:

```
COleFont font = m_g2antt.GetFont();
font.SetName( "Tahoma" );
m_g2antt.Refresh();
```

the C++ sample requires definition of COleFont class (#include "Font.h")

The following VB.NET sample assigns by code a new font to the control:

```
With AxG2antt1
   Dim font As System.Drawing.Font = New System.Drawing.Font("Tahoma", 10,
FontStyle.Regular, GraphicsUnit.Point)
   .Font = font
   .CtlRefresh()
End With
```

The following C# sample assigns by code a new font to the control:

```
System.Drawing.Font font = new System.Drawing.Font("Tahoma", 10, FontStyle.Regular);
```

```
axG2antt1.Font = font;
axG2antt1.CtlRefresh();
```

The following VFP sample assigns by code a new font to the control:

```
with thisform.G2antt1.Object
   .Font.Name = "Tahoma"
   .Refresh()
endwith
```

The following Template sample assigns by code a new font to the control:

```
Font
{
   Name = "Tahoma"
}
```

property G2antt.ForeColor as Color

Retrieves or sets a value that indicates the control's foreground color.

Type	Description
Color	A color expression that indicates the control's foreground color.

The ForeColor property changes the foreground color of the control's scrolled area. The ExG2antt control can group the columns into two categories: locked and unlocked. The Locked category contains all the columns that are fixed to the left area of the client area. These columns cannot be scrolled horizontally. Use the CountLockedColumns to specify the number of locked columns. The unlocked are contains the columns that can be scrolled horizontally. To change the background color of the control's locked area use BackColorLock property. Use the CellForeColor property to specify the cell's foreground color. Use the ItemForeColor property to specify the item's foreground color.

property G2antt.ForeColorHeader as Color

Specifies the header's foreground color.

Type	Description
Color	A color expression that indicates the foreground color for control's header.

Use the BackColorHeader and ForeColorHeader properties to customize the control's header. If the Def(exHeaderForeColor) property is not zero, it defines the foreground color to paint the column's caption in the header area. Use the Font property to change the control's font. Use the Add method to add new columns to the control. Use the HeaderVisible property to hide the control's header bar.

property G2antt.ForeColorLock as Color

Retrieves or sets a value that indicates the control's foreground color for the locked area.

Type	Description
Color	A color expression that indicates the control's foreground color for the locked area.

The ExG2antt control can group the control columns into two categories: locked and unlocked. The Locked category contains all the columns that are fixed to the left area of the client area. These columns cannot be scrolled horizontally. Use the CountLockedColumns to specify the number of locked columns. The unlocked are contains the columns that can be scrolled horizontally. To change the background color of the control's locked area use BackColorLock property.

property G2antt.ForeColorSortBar as Color

Retrieves or sets a value that indicates the sort bar's foreground color.

Type	Description
Color	A color expression that indicates the foreground color of the control's sort bar.

Use the ForeColorSortBar property to specify the foreground color of the caption in the control's sort bar. Use the SortBarVisible property to show the control's sort bar. Use the SortBarCaption property to specify the caption of the sort bar, when the control's sort bar contains no columns. Use the BackColorSortBar property to specify the background color of the control's sort bar. Use the BackColorSortBarCaption property to specify the caption's background color in the control's sort bar. Use the ForeColor property to specify the control's foreground color. Use the ForeColorHeader property to specify the background color of the control's header bar.

method G2antt.FormatABC (Expression as String, [A as Variant], [B as Variant], [C as Variant])

Formats the A,B,C values based on the giving expression and returns the result.

Type	Description
Expression as String	A String that defines the expression to be evaluated.
A as Variant	A VARIANT expression that indicates the value of the A keyword.
B as Variant	A VARIANT expression that indicates the value of the B keyword.
C as Variant	A VARIANT expression that indicates the value of the C keyword.
Return	**Description**
Variant	A VARIANT expression that indicates the result of the evaluation the G2antt.

The FormatABC method formats the A,B,C values based on the giving expression and returns the result.

For instance:

- "A + B + C", adds / concatenates the values of the A, B and C
- "value MIN 0 MAX 99", limits the value between 0 and 99
- "value format ``", formats the value with two decimals, according to the control's panel setting
- "date(`now`)" returns the current time as double

The FormatABC method supports the following keywords, constants, operators and functions:

- **A** or **value** keyword, indicates a variable A whose value is giving by the A parameter
- **B** keyword, indicates a variable B whose value is giving by the B parameter
- **C** keyword, indicates a variable C whose value is giving by the C parameter

This property/method supports predefined constants and operators/functions as described here.

property G2antt.FormatAnchor(New as Boolean) as String

Specifies the visual effect for anchor elements in HTML captions.

Type	Description
New as Boolean	Boolean expression that indicates whether to specify the anchors never clicked or anchors being clicked.
String	A String expression that indicates the HTMLformat to apply to anchor elements.

By default, the FormatAnchor(**True**) property is "<u><fgcolor=0000FF>#" that indicates that the anchor elements (that were never clicked) are underlined and shown in light blue. Also, the FormatAnchor(**False**) property is "<u><fgcolor=000080>#" that indicates that the anchor elements are underlined and shown in dark blue. *You can use the <a> anchor elements to insert hyperlinks to cells, bars or links.* Use the CellValue property to specify the cell's caption. Use the ItemBar(,,exBarCaption) property to specify the bar's caption. Use the Link(,exLinkText) property to specify a caption to be displayed on the link.

The visual effect is applied to the anchor elements, if the FormatAnchor property is not empty. For instance, if you want to do not show with a new effect the clicked anchor elements, you can use the FormatAnchor(**False**) = "", that means that the clicked or not-clicked anchors are shown with the same effect that's specified by FormatAnchor(**True**). An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link. The **<a>** element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the AnchorClick event to notify that the user clicks an anchor element. This event is fired only if prior clicking the control it shows the hand cursor. The AnchorClick event carries the identifier of the anchor, as well as application options that you can specify in the anchor element. The hand cursor is shown when the user hovers the mouse on the anchor elements.

method G2antt.FreezeEvents (Freeze as Boolean)

Prevents the control to fire any event.

Type	Description
Freeze as Boolean	A Boolean expression that specifies whether the control' events are froze or unfroze

The FreezeEvents(True) method freezes the control's events until the FreezeEvents(False) method is called. You can use the FreezeEvents method to improve performance of the control while loading data into it. For instance, the Change event is fired anytime the cell's value is changed (CellValue property), so during init time, you can call FreezeEvents(True) before, and FreezeEvents(False) after initialization is done.

The following samples show how you can lock the events while adding columns, items to the control:

```
' Change event - Occurs when the user changes the cell's content.
Private Sub G2antt1_Change(ByVal Item As EXG2ANTTLibCtl.HITEM,ByVal ColIndex As
Long,NewValue As Variant)
   With G2antt1
      Debug.Print( "Change event" )
   End With
End Sub

With G2antt1
   .FreezeEvents True
   .BeginUpdate
   With .Columns
      .Add("C1").Def(exCellHasCheckBox) = True
      .Add "C2"
   End With
   With .Items
      .CellValue(.AddItem("SubItem 1.1"),1) = "SubItem 1.2"
      .CellValue(.AddItem("SubItem 2.1"),1) = "SubItem 2.2"
   End With
   .EndUpdate
   .FreezeEvents False
End With
```

property G2antt.FullRowSelect as CellSelectEnum

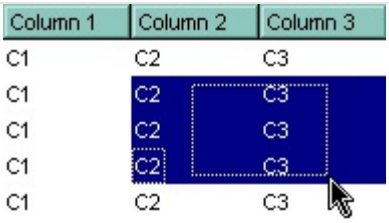
Enables full-row selection in the control.

Type	Description
CellSelectEnum	A CellSelectEnum expression that indicates whether the entire row is selected.

Use the FullRowSelect property to determine when the item or cell is selected. If the FullRowSelect property is exColumnSel, the SelectColumnIndex property determines the selected column. By default, the FullRowSelect property is exItemSel, and so the entire item is selected. If the FullRowSelect property is exRectSel property, the user can selects a range of cells by dragging. Use the Selected property to determine whether a cell is selected, if the FullRowSelect property is exRectSel. Use the SingleSel property to allow multiple items/cells in the selection. For instance, the FullRowSelect = True (boolean value) is the same as FullRowSelect = exItemSel, and FullRowSelect = False is the same as FullRowSelect = exColumnSel.

The following VB sample copies the selected cells to the clipboard, if the FullRowSelect property is exRectSel:

```
Private Sub G2antt1_SelectionChanged()
  Dim strData As String
  With G2antt1
    Dim i As Long, h As HITEM
    For i = 0 To .Items.SelectCount - 1
      h = .Items.SelectedItem(i)
      Dim c As Column
      For Each c In .Columns
        If (c.Selected) Then
          strData = strData + .Items.CellCaption(h, c.Index) + vbTab
        End If
      Next
      strData = strData + vbCrLf
    Next
  End With
  Clipboard.Clear
  Clipboard.SetText strData
End Sub
```



The following C++ sample copies the selected cells to the clipboard, if the FullRowSelect property is exRectSel:

```
#include "Column.h"
#include "Columns.h"
#include "Items.h"
void OnSelectionChangedG2antt1()
{
   CString strData;
   CColumns cols = m_g2antt.GetColumns();
   CItems items = m_g2antt.GetItems();
   for ( long i = 0; i < items.GetSelectCount(); i++ )
   {
      COleVariant vtItem( items.GetSelectedItem( i ) );
      for ( long j = 0; j < cols.GetCount(); j++ )
      {
         COleVariant vtColumn( j );
         if ( cols.GetItem( vtColumn ).GetSelected() )
            strData += items.GetCellCaption(vtItem, vtColumn ) + "\t";
      }
      strData += "\r\n";
   }
   if ( OpenClipboard() )
   {
      EmptyClipboard();
      HGLOBAL hGlobal = GlobalAlloc( GMEM_MOVEABLE | GMEM_DDESHARE,
strData.GetLength() );
      CopyMemory( GlobalLock( hGlobal ), strData.operator LPCTSTR(),
strData.GetLength() );
      GlobalUnlock( hGlobal );
      SetClipboardData( CF_TEXT, hGlobal );
      CloseClipboard();
   }
```

```
}
```

The following VB.NET sample copies the selected cells to the clipboard, if the FullRowSelect property is exRectSel:

```
Private Sub AxG2antt1_SelectionChanged(ByVal sender As Object, ByVal e As
System.EventArgs) Handles AxG2antt1.SelectionChanged
    Dim strData As String = ""
    With AxG2antt1
        Dim i As Integer, h As Integer, j As Integer
        For i = 0 To .Items.SelectCount - 1
            h = .Items.SelectedItem(i)
            For j = 0 To .Columns.Count - 1
                Dim c As EXG2ANTTLib.Column = .Columns(j)
                If (c.Selected) Then
                    strData = strData + .Items.CellCaption(h, c.Index) + vbTab
                End If
            Next
            strData = strData + vbCrLf
        Next
    End With
    Clipboard.Clear()
    Clipboard.SetText(strData)
End Sub
```

The following C# sample copies the selected cells to the clipboard, if the FullRowSelect property is exRectSel:

```
private void axG2antt1_SelectionChanged(object sender, System.EventArgs e)
{
    string strData = "";
    for (int i = 0; i < axG2antt1.Items.SelectCount; i++)
    {
        for ( int j = 0; j < axG2antt1.Columns.Count; j++ )
            if (axG2antt1.Columns[j].Selected)
            {
                string cellData =
axG2antt1.Items.get_CellCaption(axG2antt1.Items.get_SelectedItem(i), j);
```

```
            strData += cellData + "\t";
        }
    strData += "\r\n";
  }
  Clipboard.Clear();
  Clipboard.SetText(strData);
}
```

The following VFP sample copies the selected cells to the clipboard, if the FullRowSelect property is exRectSel (SelectionChanged event):

```
*** ActiveX Control Event ***

with thisform.G2antt1.Items
  local strData, i, j, cols
  strData = ""
  cols = thisform.G2antt1.Columns
  for i = 0 to .SelectCount - 1
    .DefaultItem = .SelectedItem( i )
    for j = 0 to cols.Count - 1
      if ( cols.Item(j).Selected )
        strData = strData + .CellCaption(0,j) + chr(9)
      endif
    next
    strData = strData + chr(13) + chr(10)
  next
  _CLIPTEXT = strData
endwith
```

method G2antt.GetItems (Options as Variant)

Gets the collection of items into a safe array,

Type	Description
Options as Variant	Specifies a long expression as follows: • if **0**, the result is a two-dimensional array with cell's values. The list includes the *collapsed* items, and the items are included as they are displayed (sorted, filtered). This option exports the values of cells. This option exports the values of the cells (CellValue property). • if **1**, the result the one-dimensional array of handles of items in the control as they are displayed (sorted, filtered). The list *does not include the collapsed items*. For instance, the first element in the array indicates the handle of the first item in the control, which can be different that FirstVisibleItem result, even if the control is vertically scrolled. This option exports the handles of the items. For instance, you can use the ItemToIndex property to get the index of the item based on its handle. • **else if other, and the number of columns is 1**, the result is a one-dimensional array that includes the items and its child items as they are displayed (sorted, filtered). In this case, the array may contains other arrays that specifies the child items. The list includes the *collapsed* items, and the items are included as they are displayed (sorted, filtered). This option exports the values of the cells (CellValue property) If missing, the Options parameter is 0. If the control displays no items, the result is an empty object (VT_EMPTY).

Return	Description
Variant	A safe array that holds the items in the control. If the control has a single column, the GetItems returns a single dimension array (object[]), else The safe array being returned has two dimensions (object[,]). The first

	dimension holds the collection of columns, and the second holds the cells.

The GetItems method gets a safe array that holds the items in the control. The GetItems method gets the items as they are displayed, sorted and filtered. If the Options parameter is 0, the GetItems method collect the child items as well, no matter if the parent item is collapsed or expanded. Use the PutItems method to load an array to the control. The method returns nothing if the control has no columns or items. Use the Items property to access the items collection. You can use the GetItems(1) method to get the list of handles for the items as they are displayed, sorted and filtered. The GetItems method returns an empty expression (VT_EMPTY), if there is no items in the result.

/NET Assembly:

The following C# sample converts the returned value to a object[] when the control contains a single column:

```
object[] Items = (object[])exg2antt1.GetItems()
```

or when the control contains multiple columns, the syntax is as follows:

```
object[,] Items = (object[,])exg2antt1.GetItems()
```

The following VB.NET sample converts the returned value to a Object() when the control contains a single column:

```
Dim Items As Object() = Exg2antt1.GetItems()
```

or when the control contains multiple columns, the syntax is as follows:

```
Dim Items As Object(,) = Exg2antt1.GetItems()
```

/COM version:

The following VB sample gets the items from a control and put them to the second one:

```
With G2antt2
  .BeginUpdate
    .Columns.Clear
    Dim c As EXG2ANTTLibCtl.Column
    For Each c In G2antt1.Columns
      .Columns.Add c.Caption
    Next
    .PutItems G2antt1.GetItems
```

```
    .EndUpdate
End With
```

The following C++ sample gets the items from a control an put to the second one:

```
#include "Items.h"
#include "Columns.h"
#include "Column.h"
m_g2antt2.BeginUpdate();
    CColumns columns = m_g2antt.GetColumns(), columns2 = m_g2antt2.GetColumns();
    for ( long i = 0; i < columns.GetCount(); i++ )
        columns2.Add( columns.GetItem( COleVariant( i ) ).GetCaption() );
    COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
    COleVariant vtItems = m_g2antt.GetItems( vtMissing );
    m_g2antt2.PutItems( &vtItems, vtMissing );
m_g2antt2.EndUpdate();
```

The following C# sample gets the items from a control and put them to a second one:

```
axG2antt2.BeginUpdate();
for (int i = 0; i < axG2antt1.Columns.Count; i++)
    axG2antt2.Columns.Add(axG2antt1.Columns[i].Caption);
object vtItems = axG2antt1.GetItems("");
axG2antt2.PutItems(ref vtItems);
axG2antt2.EndUpdate();
```

The following VB.NET sample gets the items from a control and put them to a second one:

```
With AxG2antt2
    .BeginUpdate()
    Dim j As Integer
    For j = 0 To AxG2antt1.Columns.Count - 1
        .Columns.Add(AxG2antt1.Columns(j).Caption)
    Next
    Dim vtItems As Object
    vtItems = AxG2antt1.GetItems("")
    .PutItems(vtItems)
    .EndUpdate()
End With
```

The following VFP sample gets the items from a control and put them to a second one:

```
local i
with thisform.G2antt2
   .BeginUpdate()
   for i = 0 to thisform.G2antt1.Columns.Count - 1
      .Columns.Add( thisform.G2antt1.Columns(i).Caption )
   next
   local array vtItems[1]
   vtItems = thisform.G2antt1.GetItems("")
   .PutItems( @vtItems )
   .EndUpdate()
endwith
```

property G2antt.GridLineColor as Color

Specifies the grid line color.

Type	Description
Color	A color expression that indicates the color of the grid lines.

Use the GridLineColor property to specify the color for grid lines. Use the DrawGridLines property to show the grid lines in the items area. The GridLineStyle property to specify the style for horizontal or/and vertical gridlines in the control. Use the DrawLevelSeperator property to draw lines between levels inside the chart's header. Use the DrawTickLines property to specify whether the grid lines between time units in the level are visible or hidden. Use the DrawGridLines property to specify whether the control draws the grid lines in the chart's area. Use the LinesAtRoot property specifies whether the control links the root items of the control. Use the HasLines property to specify whether the control draws the link between child items to their corresponding parent item.

property G2antt.GridLineStyle as GridLinesStyleEnum

Specifies the style for gridlines in the list part of the control.

Type	Description
GridLinesStyleEnum	A GridLinesStyleEnum expression that specifies the style to show the control's horizontal or vertical lines.

By default, the GridLineStyle property is exGridLinesDot. The GridLineStyle property has effect only if the DrawGridLines property is not zero. The GridLineStyle property can be used to specify the style for horizontal or/and vertical grid lines. Use the GridLineColor property to specify the color for grid lines. Use the LinesAtRoot property specifies whether the control links the root items of the control. Use the HasLines property to specify whether the control draws the link between child items to their corresponding parent item. The grid lines are shown only in the columns part of the controls, if you require the grid lines in the chart view use the DrawGridLines property of the Chart object.

The following VB sample shows dash style for horizontal gridlines, and solid style for vertical grid lines:

```
GridLineStyle = GridLinesStyleEnum.exGridLinesHDash Or
GridLinesStyleEnum.exGridLinesVSolid
```

The following VB/NET sample shows dash style for horizontal gridlines, and solid style for vertical grid lines:

```
GridLineStyle = exontrol.EXGANTTLib.GridLinesStyleEnum.exGridLinesHDash Or
exontrol.EXGANTTLib.GridLinesStyleEnum.exGridLinesVSolid
```

The following C# sample shows dash style for horizontal gridlines, and solid style for vertical grid lines:

```
GridLineStyle = exontrol.EXGANTTLib.GridLinesStyleEnum.exGridLinesHDash |
exontrol.EXGANTTLib.GridLinesStyleEnum.exGridLinesVSolid;
```

The following Delphi sample shows dash style for horizontal gridlines, and solid style for vertical grid lines:

```
GridLineStyle := Integer(EXGANTTLib.GridLinesStyleEnum.exGridLinesHDash) Or
Integer(EXGANTTLib.GridLinesStyleEnum.exGridLinesVSolid);
```

The following VFP sample shows dash style for horizontal gridlines, and solid style for vertical grid lines:

GridLineStyle = 36

method G2antt.Group ()

Forces the control to do a regrouping of the columns.

Type	Description

property G2antt.HasButtons as ExpandButtonEnum

Adds a button to the left side of each parent item.

Type	Description
ExpandButtonEnum	An ExpandButtonEnum expression that indicates whether the left side button of each parent item is visible or hidden.

The HasButtons property has effect only if the data is displayed as a tree. Use the InsertItem method to insert child items. The control displays a +/- button to parent items, if the HasButtons property is not zero, the ItemChild property is not empty, or the ItemHasChildren property is True. The user can click the +/- button to expand or collapse the child items as an alternative to double-clicking the parent item, in case the ExpandOnDblClick property is True. Use the ExpandItem property of Items object to programmatically expand/collapse an item. The HasButtonsCustom property specifies the index of icons being used for +/- signs on parent items, when HasButtons property is exCustom.

The following VB sample changes the +/- button appearance:

```
With G2antt1
   .HasButtons = ExpandButtonEnum.exWPlus
End With
```

The following C++ sample changes the +/- button appearance:

```
m_g2antt.SetHasButtons( 3 /*exWPlus*/ );
```

The following VB.NET sample changes the +/- button appearance:

```
With AxG2antt1
   .HasButtons = EXG2ANTTLib.ExpandButtonEnum.exWPlus
End With
```

The following C# sample changes the +/- button appearance:

```
axG2antt1.HasButtons = EXG2ANTTLib.ExpandButtonEnum.exWPlus;
```

The following VFP sample changes the +/- button appearance:

```
with thisform.G2antt1
   .HasButtons = 3 && exWPlus
```

endwith

property G2antt.HasButtonsCustom(Expanded as Boolean) as Long

Specifies the index of icons for +/- signs when the HasButtons property is exCustom.

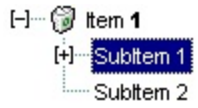
Type	Description
Expanded as Boolean	A boolean expression that indicates the sign being changed.
Long	A long expression that indicates the icon being used for +/- signs on the parent items. The last 7 bits in the high significant byte of the long expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part.

Use the HasButtonsCustom property to assign custom icons to the +/- signs on the parent items. The HasButtonsCustom property has effect only if the HasButtons property is exCustom. Use the Images, ReplaceIcon methods to add new icons to the control, at runtime.

The following VB sample specifies different (as in the screen shot) +/- signs for the control:

```
With G2antt1
   .BeginUpdate
      .Images
"gBJJgBAICAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olUrlktl0vmExm

      .LinesAtRoot = exLinesAtRoot
      .HeaderVisible = False
      .HasButtons = exCustom
      .HasButtonsCustom(False) = 1
      .HasButtonsCustom(True) = 2
      .Columns.Add "Column 1"
      With .Items
         Dim h As HITEM
         h = .AddItem("Item 1")
         .InsertItem h, , "SubItem 1"
```

```
        .InsertItem h, , "SubItem 2"
      End With
   .EndUpdate
End With
```



The following C++ sample specifies different (as in the screen shot) +/- signs for the control:

```
#include "Items.h"
#include "Columns.h"
#include "Column.h"
m_g2antt.BeginUpdate();
m_g2antt.Images( COleVariant(
"gBJJgBAICAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olUrlktl0vmExm
 ) );
m_g2antt.SetLinesAtRoot( -1 );
m_g2antt.SetHeaderVisible( FALSE );
m_g2antt.SetHasButtons( 4 /*exCustom*/ );
m_g2antt.SetHasButtonsCustom( FALSE, 1 );
m_g2antt.SetHasButtonsCustom( TRUE, 2 );
m_g2antt.GetColumns().Add( "Column 1" );
COleVariant vtMissing; V_VT( &vtMissing; ) = VT_ERROR;
CItems items = m_g2antt.GetItems();
long h = items.AddItem( COleVariant( "Item 1" ) );
items.InsertItem( h, vtMissing, COleVariant( "SubItem 1" ) );
items.InsertItem( h, vtMissing, COleVariant( "SubItem 2" ) );
m_g2antt.EndUpdate();
```

The following VB.NET sample specifies different (as in the screen shot) +/- signs for the control:

```
With AxG2antt1
   .BeginUpdate()

.Images("gBJJgBAICAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olUrlkt
```

```
    .LinesAtRoot = EXG2ANTTLib.LinesAtRootEnum.exLinesAtRoot
    .HeaderVisible = False
    .HasButtons = EXG2ANTTLib.ExpandButtonEnum.exCustom
    .set_HasButtonsCustom(False, 1)
    .set_HasButtonsCustom(True, 2)
    .Columns.Add("Column 1")
    With .Items
        Dim h As Long
        h = .AddItem("Item 1")
        .InsertItem(h, , "SubItem 1")
        .InsertItem(h, , "SubItem 2")
    End With
    .EndUpdate()
End With
```

The following C# sample specifies different (as in the screen shot) +/- signs for the control:

```
axG2antt1.BeginUpdate();
axG2antt1.Images("gBJJgBAICAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcb

axG2antt1.LinesAtRoot = EXG2ANTTLib.LinesAtRootEnum.exLinesAtRoot;
axG2antt1.HeaderVisible = false;
axG2antt1.HasButtons = EXG2ANTTLib.ExpandButtonEnum.exCustom;
axG2antt1.set_HasButtonsCustom(false, 1);
axG2antt1.set_HasButtonsCustom(true, 2);
axG2antt1.Columns.Add("Column 1");
int h = axG2antt1.Items.AddItem("Item 1");
axG2antt1.Items.InsertItem(h, "", "SubItem 1");
axG2antt1.Items.InsertItem(h, "", "SubItem 2");
axG2antt1.EndUpdate();
```

The following VFP sample specifies different (as in the screen shot) +/- signs for the control:

```
with thisform.G2antt1
    .BeginUpdate()
        local s
```

```
      s =
"gBJJgBAICAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olUrlktl0vmExm

      s = s +
"1ntFptVrtItt1vuFxuVzul1u13vF5vV7vl9v1/wGBwWDwmFw2HxGJxWLxmNx2PyGRyWTymV

      .Images(s)
      .LinesAtRoot = -1
      .HeaderVisible = .f.
      .HasButtons = 4 &&exCustom;
      local sT, sCR
      sCR = chr(13) + chr(10)
      sT = "HasButtonsCustom(True) = 2"+ sCR
      sT = sT + "HasButtonsCustom(False) = 1"+ sCR
      .Template = sT
      .Columns.Add("Column 1")
      With .Items
         local h
         h = .AddItem("Item 1")
         .InsertItem(h, , "SubItem 1")
         .InsertItem(h, , "SubItem 2")
      EndWith
   .EndUpdate()
endwith
```

property G2antt.HasLines as HierarchyLineEnum

Enhances the graphic representation of a tree control's hierarchy by drawing lines that link child items to their corresponding parent item.

Type	Description
HierarchyLineEnum	An HierarchyLinesEnum expression that indicates whether the control uses the lines to link the items of the hierarchy.

Use the HasLines property to hide the hierarchy lines. Use the LinesAtRoot property to allow control displays a line that links that root items of the control. Use the InsertItem method to insert new items to the control. Use HasButtons property to hide the buttons displayed at the left of each parent item. Use the DrawGridLines property to display grid lines. Use the InsertControlItem property to insert an ActiveX item.

property G2antt.HeaderAppearance as AppearanceEnum

Retrieves or sets a value that indicates the header's appearance.

Type	Description
AppearanceEnum	An AppearanceEnum expression that indicates the header's appearance.

Use the HeaderAppearance property to change the appearance of the control's header bar. Use the HeaderVisible property to hide the control's header bar. Use the Appearance property to specify the control's appearance.

property G2antt.HeaderEnabled as Boolean

Enables or disables the control's header.

Type	Description
Boolean	A boolean expression that specifies whether the control's header is enabled or disabled.

By default, the HeaderEnabled property is True. The HeaderEnabled property enables or disables the control's header (including the control's sort/groupby-bar). If the header is disabled, the user can't resize, sort or drag and drop any column. Also, if the header is disabled, the control's sort/groupby-bar is disabled as well. The HeaderVisible property shows or hides the control's header. The SortBarVisible property shows or hides the control's sort/groupby-bar.

property G2antt.HeaderHeight as Long

Retrieves or sets a value indicating the control's header height.

Type	Description
Long	A long expression that indicates the height of the control's header bar.

By default, the HeaderHeight property is 18 pixels. Use the HeaderHeight property to change the height of the control's header bar. Use the HeaderVisible property to hide the control's header bar. Use the LevelKey property to display the control's header bar using multiple levels. If the control displays the header bar using multiple levels the HeaderHeight property gets the height in pixels of a single level in the header bar. The control's header displays multiple levels if there are two or more neighbor columns with the same non empty level key. Use the HTMLCaption property to display multiple lines in the column's caption. Use the Add method to add new columns to the control. Use the LevelKey property to specify columns on the same level. Use the LevelCount property to specify the number of levels being displayed in the chart's header. *If the HeaderSingleLine property is False, the HeaderHeight property specifies the maximum height of the control's header when the user resizes the columns.*

The following VB sample displays a header bar using multiple lines:

```
With G2antt1
    .BeginUpdate
      .HeaderHeight = 32
      With .Columns.Add("Column 1")
        .HTMLCaption = "Line1<br>Line2"
      End With
      With .Columns.Add("Column 2")
        .HTMLCaption = "Line1<br>Line2"
      End With
    .EndUpdate
End With
```

The following C++ sample displays a header bar using multiple lines:

```
#include "Columns.h"
#include "Column.h"
m_g2antt.BeginUpdate();
m_g2antt.SetHeaderHeight( 32 );
```

```
m_g2antt.SetHeaderVisible( TRUE );
CColumn column1( V_DISPATCH( &m_g2antt.GetColumns().Add( "Column 1" ) ) );
   column1.SetHTMLCaption( "Line1<br>Line2" );
CColumn column2( V_DISPATCH( &m_g2antt.GetColumns().Add( "Column 2" ) ) );
   column2.SetHTMLCaption( "Line1<br>Line2" );
m_g2antt.EndUpdate();
```

The following VB.NET sample displays a header bar using multiple lines:

```
With AxG2antt1
   .BeginUpdate()
   .HeaderVisible = True
   .HeaderHeight = 32
   With .Columns.Add("Column 1")
      .HTMLCaption = "Line1<br>Line2"
   End With
   With .Columns.Add("Column 2")
      .HTMLCaption = "Line1<br>Line2"
   End With
   .EndUpdate()
End With
```

The following C# sample displays a header bar using multiple lines:

```
axG2antt1.BeginUpdate();
axG2antt1.HeaderVisible = true;
axG2antt1.HeaderHeight = 32;
EXG2ANTTLib.Column column1 = axG2antt1.Columns.Add("Column 1") as
EXG2ANTTLib.Column ;
column1.HTMLCaption = "Line1<br>Line2";
EXG2ANTTLib.Column column2 = axG2antt1.Columns.Add("Column 2") as
EXG2ANTTLib.Column;
column2.HTMLCaption = "Line1<br>Line2";
axG2antt1.EndUpdate();
```

The following VFP sample displays a header bar using multiple lines:

```
with thisform.G2antt1
   .BeginUpdate()
```

```
    .HeaderVisible = .t.
    .HeaderHeight = 32
    with .Columns.Add("Column 1")
       .HTMLCaption = "Line1<br>Line2"
    endwith
    with .Columns.Add("Column 2")
       .HTMLCaption = "Line1<br>Line2"
    endwith
    .EndUpdate()
endwith
```

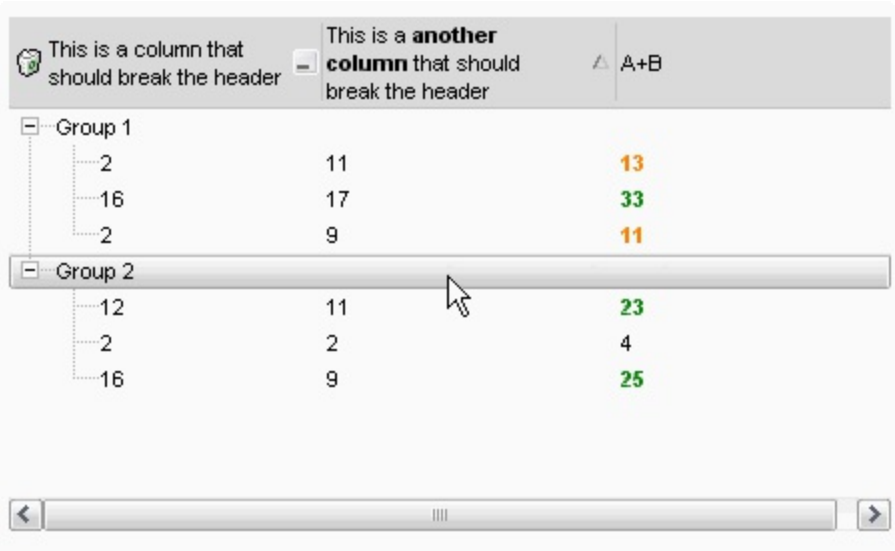
property G2antt.HeaderSingleLine as Boolean

Specifies whether the control resizes the columns header and wraps the captions in single or multiple lines.

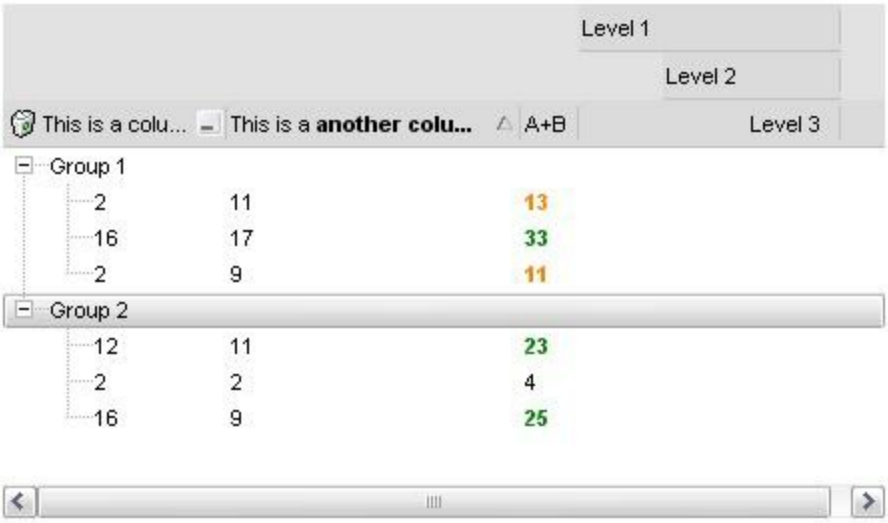
Type	Description
Boolean	A boolean expression that specifies whether the header displays single or multiple lines.

By defauly, the HeaderSingleLine property is True. If the HeaderSingleLine property is False the control breaks the column's caption as soon as the user resizes the column. **In this case the HeaderHeight property specifies the maximum height of the control's header.** The initial height is computed based on the control's Font property. The Caption property specifies the caption of the column being displayed in the control's header. The HTMLCaption property specifies the HTML caption of the column being displayed in the column's header. Use the LevelKey property to display the control's header on multiple levels.

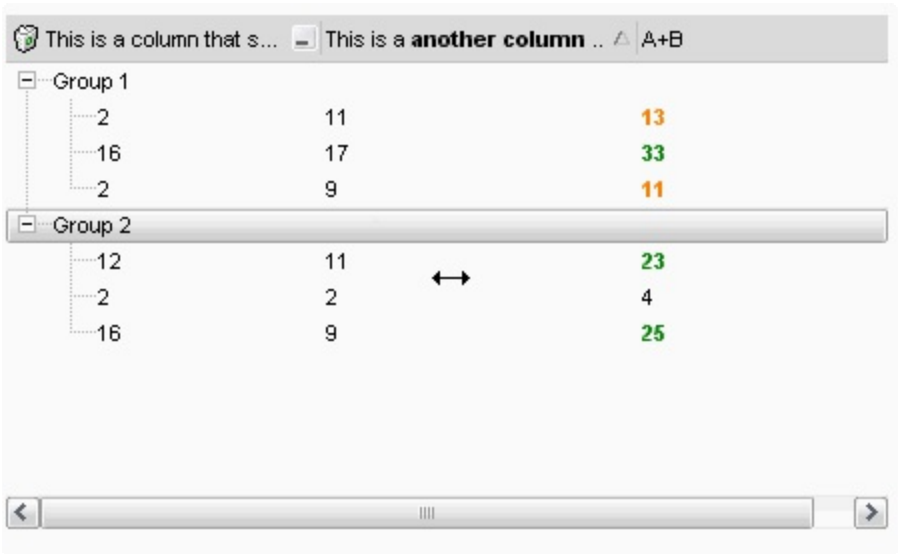
The following screen show shows the control's header while it displays a multiple lines (HeaderSingleLine = False):



The following screen shot shows the control's header on multiple levels using the LevelKey property:



The following screen show shows the control's header while it displays a single line (HeaderSingleLine = True):



property G2antt.HeaderVisible as HeaderVisibleEnum

Retrieves or sets a value that indicates whether the control's header is visible or hidden.

Type	Description
HeaderVisibleEnum	A HeaderVisibleEnum expression that specifies whether the control's header bar is visible or hidden.

By default, the HeaderVisible property is exHeaderVisible (True). Use the HeaderVisible property to hide the control's header bar. The control's header bar displays the levels in the chart area too. Use the LevelCount property to specify the number of levels being displayed in the chart's header. Use the Level property to access the level in the chart area. Use the Caption property to specify the column's caption being displayed in the control's header bar. Use the HeaderAppearance property to change the header bar's appearance. The HeaderEnabled property enables or disables the control's header. Use the BackColorHeader and ForeColorHeader properties to customize the control's header. Use the BackColorLevelHeader property to specify the background color of the header when it displays multiple levels. Use the HeaderHeight property to specify the height of the control's header bar. Use the SortBarVisible property to specify whether the control's sort bar is visible or it is hidden. Use the OverviewVisible property to show or hide the chart's overview area. Use the ColumnsAllowSizing property to allow resizing the columns, when the control's header bar is not visible. The HistogramHeaderVisible property to show the chart's header in bottom part of the histogram.

property G2antt.HideSelection as Boolean

Returns a value that determines whether selected item appears highlighted when a control loses the focus.

Type	Description
Boolean	A boolean expression that indicates whether the selected item appears highlighted when a control loses the focus.

By default, the HideSelection property is False. You can use this property to indicate which item is highlighted while another form or a dialog box has the focus. Use the SelForeColor and SelBackColor property to customize the colors for the selected items in the control. Use the SelectItem property to programmatically select an item. Use the SelectedItem and SelectCount property to retrieve the list of selected items. Use the SelectableItem property to specify whether an items can be selected.

property G2antt.HotBackColor as Color

Retrieves or sets a value that indicates the hot-tracking background color.

Type	Description
Color	A color expression that indicates the background color for item from the cursor (hovering the item). Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

By default, the HotBackColor property is 0, which means that the HotBackColor property has no effect. Use the HotBackColor property on a non-zero value to highlight the item from the cursor. The HotForeColor property specifies the foreground color to highlight the item from the cursor. The ItemFromPoint property gets the item from the cursor. The SelBackColor property specifies the selection background color. The SelBackMode property specifies the way the selected items are shown in the control.

The following sample displays a different background color mouse passes over an item.

VBA

```
With G2antt1
   .BeginUpdate
   .Columns.Add "Def"
   .HotBackColor = RGB(0,0,128)
   .HotForeColor = RGB(255,255,255)
   With .Items
      .AddItem "Item A"
      .AddItem "Item B"
      .AddItem "Item C"
   End With
   .EndUpdate
End With
```

VB6

```
With G2antt1
   .BeginUpdate
   .Columns.Add "Def"
```

```
   .HotBackColor = RGB(0,0,128)
   .HotForeColor = RGB(255,255,255)
   With .Items
      .AddItem "Item A"
      .AddItem "Item B"
      .AddItem "Item C"
   End With
   .EndUpdate
End With
```

VB.NET

```
With Exg2antt1
   .BeginUpdate()
   .Columns.Add("Def")
   .HotBackColor = Color.FromArgb(0,0,128)
   .HotForeColor = Color.FromArgb(255,255,255)
   With .Items
      .AddItem("Item A")
      .AddItem("Item B")
      .AddItem("Item C")
   End With
   .EndUpdate()
End With
```

VB.NET for /COM

```
With AxG2antt1
   .BeginUpdate()
   .Columns.Add("Def")
   .HotBackColor = RGB(0,0,128)
   .HotForeColor = RGB(255,255,255)
   With .Items
      .AddItem("Item A")
      .AddItem("Item B")
      .AddItem("Item C")
   End With
   .EndUpdate()
```

```
End With
```

C++

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)->GetControlUnknown();
spG2antt1->BeginUpdate();
spG2antt1->GetColumns()->Add(L"Def");
spG2antt1->PutHotBackColor(RGB(0,0,128));
spG2antt1->PutHotForeColor(RGB(255,255,255));
EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems();
   var_Items->AddItem("Item A");
   var_Items->AddItem("Item B");
   var_Items->AddItem("Item C");
spG2antt1->EndUpdate();
```

C++ Builder

```
G2antt1->BeginUpdate();
G2antt1->Columns->Add(L"Def");
G2antt1->HotBackColor = RGB(0,0,128);
G2antt1->HotForeColor = RGB(255,255,255);
Exg2anttlib_tlb::IItemsPtr var_Items = G2antt1->Items;
   var_Items->AddItem(TVariant("Item A"));
   var_Items->AddItem(TVariant("Item B"));
   var_Items->AddItem(TVariant("Item C"));
G2antt1->EndUpdate();
```

C#

```
exg2antt1.BeginUpdate();
```

```
exg2antt1.Columns.Add("Def");
exg2antt1.HotBackColor = Color.FromArgb(0,0,128);
exg2antt1.HotForeColor = Color.FromArgb(255,255,255);
exontrol.EXG2ANTTLib.Items var_Items = exg2antt1.Items;
   var_Items.AddItem("Item A");
   var_Items.AddItem("Item B");
   var_Items.AddItem("Item C");
exg2antt1.EndUpdate();
```

JavaScript

```
<OBJECT classid="clsid:CD481F4D-2D25-4759-803F-752C568F53B7" id="G2antt1">
</OBJECT>

<SCRIPT LANGUAGE="JScript">
   G2antt1.BeginUpdate()

   G2antt1.Columns.Add("Def")

   G2antt1.HotBackColor = 8388608

   G2antt1.HotForeColor = 16777215

   var var_Items = G2antt1.Items

      var_Items.AddItem("Item A")

      var_Items.AddItem("Item B")

      var_Items.AddItem("Item C")

   G2antt1.EndUpdate()

</SCRIPT>
```

C# for /COM

```
axG2antt1.BeginUpdate();
```

```
axG2antt1.Columns.Add("Def");
axG2antt1.HotBackColor = Color.FromArgb(0,0,128);
axG2antt1.HotForeColor = Color.FromArgb(255,255,255);
EXG2ANTTLib.Items var_Items = axG2antt1.Items;
   var_Items.AddItem("Item A");
   var_Items.AddItem("Item B");
   var_Items.AddItem("Item C");
axG2antt1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
   COM com_Items

   anytype var_Items




   super()


   exg2antt1.BeginUpdate()

   exg2antt1.Columns().Add("Def")

   exg2antt1.HotBackColor(WinApi::RGB2int(0,0,128))

   exg2antt1.HotForeColor(WinApi::RGB2int(255,255,255))

   var_Items = exg2antt1.Items()
    com_Items = var_Items

      com_Items.AddItem("Item A")

      com_Items.AddItem("Item B")
```

```
        com_Items.AddItem("Item C")

    exg2antt1.EndUpdate()

}
```

VFP

```
with thisform.G2antt1
    .BeginUpdate
    .Columns.Add("Def")
    .HotBackColor = RGB(0,0,128)
    .HotForeColor = RGB(255,255,255)
    with .Items
        .AddItem("Item A")
        .AddItem("Item B")
        .AddItem("Item C")
    endwith
    .EndUpdate
endwith
```

dBASE Plus

```
local oG2antt,var_Items

oG2antt = form.Activex1.nativeObject
oG2antt.BeginUpdate()
oG2antt.Columns.Add("Def")
oG2antt.HotBackColor = 0x800000
oG2antt.HotForeColor = 0xffffff
var_Items = oG2antt.Items
    var_Items.AddItem("Item A")
    var_Items.AddItem("Item B")
    var_Items.AddItem("Item C")
oG2antt.EndUpdate()
```

XBasic (Alpha Five)

```
Dim oG2antt as P
Dim var_Items as P

oG2antt = topparent:CONTROL_ACTIVEX1.activex
oG2antt.BeginUpdate()
oG2antt.Columns.Add("Def")
oG2antt.HotBackColor = 8388608
oG2antt.HotForeColor = 16777215
var_Items = oG2antt.Items
   var_Items.AddItem("Item A")
   var_Items.AddItem("Item B")
   var_Items.AddItem("Item C")
oG2antt.EndUpdate()
```

Delphi 8 (.NET only)

```
with AxG2antt1 do
begin
   BeginUpdate();
   Columns.Add('Def');
   HotBackColor := Color.FromArgb(0,0,128);
   HotForeColor := Color.FromArgb(255,255,255);
   with Items do
   begin
      AddItem('Item A');
      AddItem('Item B');
      AddItem('Item C');
   end;
   EndUpdate();
end
```

Delphi (standard)

```
with G2antt1 do
begin
   BeginUpdate();
   Columns.Add('Def');
   HotBackColor := RGB(0,0,128);
```

```
   HotForeColor := RGB(255,255,255);
   with Items do
   begin
      AddItem('Item A');
      AddItem('Item B');
      AddItem('Item C');
   end;
   EndUpdate();
end
```

Visual Objects

```
local var_Items as IItems

oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:Columns:Add("Def")
oDCOCX_Exontrol1:HotBackColor := RGB(0,0,128)
oDCOCX_Exontrol1:HotForeColor := RGB(255,255,255)
var_Items := oDCOCX_Exontrol1:Items
   var_Items:AddItem("Item A")
   var_Items:AddItem("Item B")
   var_Items:AddItem("Item C")
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
OleObject oG2antt,var_Items

oG2antt = ole_1.Object
oG2antt.BeginUpdate()
oG2antt.Columns.Add("Def")
oG2antt.HotBackColor = RGB(0,0,128)
oG2antt.HotForeColor = RGB(255,255,255)
var_Items = oG2antt.Items
   var_Items.AddItem("Item A")
   var_Items.AddItem("Item B")
   var_Items.AddItem("Item C")
oG2antt.EndUpdate()
```

property G2antt.HotForeColor as Color

Retrieves or sets a value that indicates the hot-tracking foreground color.

Type	Description
Color	A color expression that indicates the foreground color for item from the cursor (hovering the item).

By default, the HotForeColor property is 0, which means that the HotForeColor property has no effect. Use the HotForeColor property on a non-zero value to highlight the item from the cursor. The HotBackColor property specifies the background color to highlight the item from the cursor. The ItemFromPoint property gets the item from the cursor. The SelForeColor property specifies the selection foreground color.

The following sample displays a different background color mouse passes over an item.

VBA

```
With G2antt1
   .BeginUpdate
   .Columns.Add "Def"
   .HotBackColor = RGB(0,0,128)
   .HotForeColor = RGB(255,255,255)
   With .Items
      .AddItem "Item A"
      .AddItem "Item B"
      .AddItem "Item C"
   End With
   .EndUpdate
End With
```

VB6

```
With G2antt1
   .BeginUpdate
   .Columns.Add "Def"
   .HotBackColor = RGB(0,0,128)
   .HotForeColor = RGB(255,255,255)
   With .Items
      .AddItem "Item A"
```

```
    .AddItem "Item B"
    .AddItem "Item C"
  End With
  .EndUpdate
End With
```

VB.NET

```
With Exg2antt1
  .BeginUpdate()
  .Columns.Add("Def")
  .HotBackColor = Color.FromArgb(0,0,128)
  .HotForeColor = Color.FromArgb(255,255,255)
  With .Items
    .AddItem("Item A")
    .AddItem("Item B")
    .AddItem("Item C")
  End With
  .EndUpdate()
End With
```

VB.NET for /COM

```
With AxG2antt1
  .BeginUpdate()
  .Columns.Add("Def")
  .HotBackColor = RGB(0,0,128)
  .HotForeColor = RGB(255,255,255)
  With .Items
    .AddItem("Item A")
    .AddItem("Item B")
    .AddItem("Item C")
  End With
  .EndUpdate()
End With
```

C++

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
spG2antt1->GetColumns()->Add(L"Def");
spG2antt1->PutHotBackColor(RGB(0,0,128));
spG2antt1->PutHotForeColor(RGB(255,255,255));
EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems();
   var_Items->AddItem("Item A");
   var_Items->AddItem("Item B");
   var_Items->AddItem("Item C");
spG2antt1->EndUpdate();
```

C++ Builder

```
G2antt1->BeginUpdate();
G2antt1->Columns->Add(L"Def");
G2antt1->HotBackColor = RGB(0,0,128);
G2antt1->HotForeColor = RGB(255,255,255);
Exg2anttlib_tlb::IItemsPtr var_Items = G2antt1->Items;
   var_Items->AddItem(TVariant("Item A"));
   var_Items->AddItem(TVariant("Item B"));
   var_Items->AddItem(TVariant("Item C"));
G2antt1->EndUpdate();
```

C#

```
exg2antt1.BeginUpdate();
exg2antt1.Columns.Add("Def");
exg2antt1.HotBackColor = Color.FromArgb(0,0,128);
exg2antt1.HotForeColor = Color.FromArgb(255,255,255);
exontrol.EXG2ANTTLib.Items var_Items = exg2antt1.Items;
```

```
   var_Items.AddItem("Item A");
   var_Items.AddItem("Item B");
   var_Items.AddItem("Item C");
exg2antt1.EndUpdate();
```

JavaScript

```
<OBJECT classid="clsid:CD481F4D-2D25-4759-803F-752C568F53B7" id="G2antt1">
</OBJECT>

<SCRIPT LANGUAGE="JScript">
   G2antt1.BeginUpdate()

   G2antt1.Columns.Add("Def")

   G2antt1.HotBackColor = 8388608

   G2antt1.HotForeColor = 16777215

   var var_Items = G2antt1.Items

      var_Items.AddItem("Item A")

      var_Items.AddItem("Item B")

      var_Items.AddItem("Item C")

   G2antt1.EndUpdate()

</SCRIPT>
```

C# for /COM

```
axG2antt1.BeginUpdate();
axG2antt1.Columns.Add("Def");
axG2antt1.HotBackColor = Color.FromArgb(0,0,128);
axG2antt1.HotForeColor = Color.FromArgb(255,255,255);
EXG2ANTTLib.Items var_Items = axG2antt1.Items;
```

```
    var_Items.AddItem("Item A");
    var_Items.AddItem("Item B");
    var_Items.AddItem("Item C");
axG2antt1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    COM com_Items

    anytype var_Items




    super()


    exg2antt1.BeginUpdate()

    exg2antt1.Columns().Add("Def")

    exg2antt1.HotBackColor(WinApi::RGB2int(0,0,128))

    exg2antt1.HotForeColor(WinApi::RGB2int(255,255,255))

    var_Items = exg2antt1.Items()
     com_Items = var_Items

        com_Items.AddItem("Item A")

        com_Items.AddItem("Item B")

        com_Items.AddItem("Item C")

    exg2antt1.EndUpdate()
```

```
}
```

VFP

```
with thisform.G2antt1
   .BeginUpdate
   .Columns.Add("Def")
   .HotBackColor = RGB(0,0,128)
   .HotForeColor = RGB(255,255,255)
   with .Items
      .AddItem("Item A")
      .AddItem("Item B")
      .AddItem("Item C")
   endwith
   .EndUpdate
endwith
```

dBASE Plus

```
local oG2antt,var_Items

oG2antt = form.Activex1.nativeObject
oG2antt.BeginUpdate()
oG2antt.Columns.Add("Def")
oG2antt.HotBackColor = 0x800000
oG2antt.HotForeColor = 0xffffff
var_Items = oG2antt.Items
   var_Items.AddItem("Item A")
   var_Items.AddItem("Item B")
   var_Items.AddItem("Item C")
oG2antt.EndUpdate()
```

XBasic (Alpha Five)

```
Dim oG2antt as P
Dim var_Items as P
```

```
oG2antt = topparent:CONTROL_ACTIVEX1.activex
oG2antt.BeginUpdate()
oG2antt.Columns.Add("Def")
oG2antt.HotBackColor = 8388608
oG2antt.HotForeColor = 16777215
var_Items = oG2antt.Items
   var_Items.AddItem("Item A")
   var_Items.AddItem("Item B")
   var_Items.AddItem("Item C")
oG2antt.EndUpdate()
```

Delphi 8 (.NET only)

```
with AxG2antt1 do
begin
   BeginUpdate();
   Columns.Add('Def');
   HotBackColor := Color.FromArgb(0,0,128);
   HotForeColor := Color.FromArgb(255,255,255);
   with Items do
   begin
      AddItem('Item A');
      AddItem('Item B');
      AddItem('Item C');
   end;
   EndUpdate();
end
```

Delphi (standard)

```
with G2antt1 do
begin
   BeginUpdate();
   Columns.Add('Def');
   HotBackColor := RGB(0,0,128);
   HotForeColor := RGB(255,255,255);
   with Items do
   begin
```

```
    AddItem('Item A');
    AddItem('Item B');
    AddItem('Item C');
  end;
  EndUpdate();
end
```

Visual Objects

```
local var_Items as IItems

oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:Columns:Add("Def")
oDCOCX_Exontrol1:HotBackColor := RGB(0,0,128)
oDCOCX_Exontrol1:HotForeColor := RGB(255,255,255)
var_Items := oDCOCX_Exontrol1:Items
  var_Items:AddItem("Item A")
  var_Items:AddItem("Item B")
  var_Items:AddItem("Item C")
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
OleObject oG2antt,var_Items

oG2antt = ole_1.Object
oG2antt.BeginUpdate()
oG2antt.Columns.Add("Def")
oG2antt.HotBackColor = RGB(0,0,128)
oG2antt.HotForeColor = RGB(255,255,255)
var_Items = oG2antt.Items
  var_Items.AddItem("Item A")
  var_Items.AddItem("Item B")
  var_Items.AddItem("Item C")
oG2antt.EndUpdate()
```

property G2antt.HTMLPicture(Key as String) as Variant

Adds or replaces a picture in HTML captions.

Type	Description
Key as String	A String expression that indicates the key of the picture being added or replaced. If the Key property is Empty string, the entire collection of pictures is cleared.
Variant	The HTMLPicture specifies the picture being associated to a key. It can be one of the followings: • a string expression that indicates the path to the picture file, being loaded. • a string expression that indicates the base64 encoded string that holds a picture object, Use the eximages tool to save your picture as base64 encoded format. • A Picture object that indicates the picture being added or replaced. (A Picture object implements IPicture interface), If empty, the picture being associated to a key is removed. If the key already exists the new picture is replaced. If the key is not empty, and it doesn't not exist a new picture is added

The HTMLPicture property handles a collection of custom size picture being displayed in the HTML captions, using the <img> tags. By default, the HTMLPicture collection is empty. Use the HTMLPicture property to add new pictures to be used in HTML captions. For instance, the HTMLPicture("pic1") = "c:\winnt\zapotec.bmp", loads the zapotec picture and associates the pic1 key to it. Any "<img>pic1</img>" sequence in HTML captions, displays the pic1 picture. On return, the HTMLPicture property retrieves a Picture object (this implements the IPictureDisp interface).

The following sample shows how to put a custom size picture in the column's header:

```
<CONTROL>.HTMLPicture("pic1") = "c:/temp/editors.gif"
<CONTROL>.HTMLPicture("pic2") = "c:/temp/editpaste.gif"

<COLUMN1>.HTMLCaption = "A <img>pic1</img>"
<COLUMN2>.HTMLCaption = "B <img>pic2</img>"
<COLUMN3>.HTMLCaption = "A <img>pic1</img> + B <img>pic2</img>"
```


A
B
△
A
+ B

property G2antt.hWnd as Long

Retrieves the control's window handle.

Type	Description
Long	A long expression that indicates the control's window handle.

Use the hWnd property to get the control's main window handle. Use the ItemWindowHost property to get the handle of the container window that host an item's ActiveX Control. The Microsoft Windows operating environment identifies each form and control in an application by assigning it a handle, or hWnd. The hWnd property is used with Windows API calls. Many Windows operating environment functions require the hWnd of the active window as an argument.

property G2antt.HyperLinkColor as Color

Specifies the hyperlink color.

Type	Description
Color	A color expression that specifies the hyperlink color.

Use the HyperLinkColor property to specify the color used when the cursor is over the hyperlink cells. A hyperlink cell has the CellHyperLink property true. The control fires the HyperLinkClick property when user clicks a cell that has the CellHyperLink property on True.

method G2antt.Images (Handle as Variant)

Sets the control's image list at runtime.

Type	Description
Handle as Variant	The Handle parameter can be: • A string expression that specifies the ICO file to add. The ICO file format is an image file format for computer icons in Microsoft Windows. ICO files contain one or more small images at multiple sizes and color depths, such that they may be scaled appropriately. For instance, Images("c:\temp\copy.ico") method adds the sync.ico file to the control's Images collection *(string, loads the icon using its path)* • A string expression that indicates the BASE64 encoded string that holds the icons list. Use the Exontrol's ExImages tool to save/load your icons as BASE64 encoded format. In this case the string may begin with "gBJJ..." *(string, loads icons using base64 encoded string)* • A reference to a Microsoft ImageList control (mscomctl.ocx, MSComctlLib.ImageList type) that holds the icons to add *(object, loads icons from a Microsoft ImageList control)* • A reference to a Picture (IPictureDisp implementation) that holds the icon to add. For instance, the VB's LoadPicture (Function LoadPicture([FileName], [Size], [ColorDepth], [X], [Y]) As IPictureDisp) or LoadResPicture (Function LoadResPicture(id, restype As Integer) As IPictureDisp) returns a picture object (object, loads icon from a Picture object) • A long expression that identifies a handle to an Image List Control (the Handle should be of HIMAGELIST type). On 64-bit platforms, the Handle parameter must be a Variant of LongLong / LONG_PTR data type (signed 64-bit (8-byte) integers), saved under llVal field, as VT_I8 type. The LONGLONG / LONG_PTR is __int64, a 64-bit integer. For instance, in C++ you can use as Images(COleVariant((LONG_PTR)hImageList)) or Images(COleVariant((LONGLONG)hImageList)), where hImageList is of

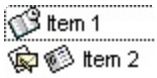
HIMAGELIST type. The GetSafeHandle() method of the CImageList gets the HIMAGELIST handle (long, loads icon from HIMAGELIST type)

The user can add images at design time, by drag and drop files to combo's image holder. The ImageSize property defines the size (width/height) of the icons within the control's Images collection. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection. Use the CellImage, CellImages properties to assign icons to a cell. Use the CellPicture property to assign a custom size picture to a cell. Use the CheckImage or RadioImage property to specify a different look for checkboxes or radio buttons in the cells.

The following VB sample adds the control's icons list from a BASE64 encoded string:

```
Dim s As String
With G2antt1
   .BeginUpdate
   s =
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olUrlktl0vmExn

   s = s + "Poyf5xoojKAg"
   .Images s

   .Columns.Add "Column 1"
   With .Items
      Dim h As HITEM
      h = .AddItem("Item 1")
      .CellImage(h, 0) = 1
      h = .AddItem("Item 2")
      .CellImages(h, 0) = "2,3"
   End With
   .EndUpdate
End With
```

If you run the sample you get:



The following VB sample loads images from a Microsoft Image List control:

```
G2antt1.Images ImageList1.hImageList
```

The following C++ sample loads icons from a BASE64 encoded string:

```
#include "Items.h"
#include "Columns.h"
#include "Column.h"
m_g2antt.BeginUpdate();
CString s =
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olUrlktl0vmExn

s +=
"/xoAw9ZiFdxbAAGVxM5yOTzkPy+MzGRpmdx2kl2epGY1WgxmZl+Yyery2yyGHyeirGoo+(

s +=
"NbDDLO2xz5PlBi3O8x0EsZD7zuG8T1vrCD5uZE7zxM+CXQNB78RKw8RRbF8Rwyu0UPS/U

s +=
"kSSAAkqUU2nE20gmp5oo6JwH+eZ31EjJwB+eBn1K/AHnBWIfvwAZwACYAHsMy9clMyFeE

m_g2antt.Images( COleVariant( s ) );
m_g2antt.GetColumns().Add( "Column 1" );
COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
CItems items = m_g2antt.GetItems();
long h = items.AddItem( COleVariant( "Item 1" ) );
items.SetCellImage( COleVariant( h ), COleVariant( (long) 0 ), 1 );
h = items.AddItem( COleVariant( "Item 2" ) );
items.SetCellImages( COleVariant( h ), COleVariant( (long) 0 ), COleVariant( "2,3" ) );
m_g2antt.EndUpdate();
```

The following C++ sample loads icons from a HIMAGELIST type:

```
SHFILEINFO sfi; ZeroMemory( &sfi, sizeof(sfi) );
 HIMAGELIST hSysImageList = (HIMAGELIST)SHGetFileInfo(_T("C:\\"),  0, &sfi, sizeof
(SHFILEINFO), SHGFI_SMALLICON | SHGFI_SYSICONINDEX  );
 m_g2antt.Images( _variant_t( (long)hSysImageList ) );
```

The following VB.NET sample loads icons from a BASE64 encoded string:

```
Dim s As String
With AxG2antt1
   .BeginUpdate()
   s =
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olUrlktl0vmExn

   s = s + "Poyf5xoojKAg"
   .Images(s)

   .Columns.Add("Column 1")
   With .Items
      Dim h As Integer
      h = .AddItem("Item 1")
      .CellImage(h, 0) = 1
      h = .AddItem("Item 2")
      .CellImages(h, 0) = "2,3"
   End With
   .EndUpdate()
End With
```

The following C# sample loads icons from a BASE64 encoded string:

```
axG2antt1.BeginUpdate();
string s =
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olUrlktl0vmExn

s = s + "Poyf5xoojKAg";
axG2antt1.Images(s);
axG2antt1.Columns.Add("Column 1");
int h = axG2antt1.Items.AddItem("Item 1");
axG2antt1.Items.set_CellImage(h, 0, 1 );
h = axG2antt1.Items.AddItem("Item 2");
axG2antt1.Items.set_CellImages(h, 0,"2,3");
axG2antt1.EndUpdate();
```

The following VFP sample loads icons from a BASE64 encoded string:

```
local s
```

```
With thisform.G2antt1
   .BeginUpdate()
   s =
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExn

   s = s +
"dr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul1u13vF5vV7vl9v1/wGBnqAQEZwmCxFhYGLib/xoAw9

   s = s +
"Goo+03mM02Jzee029y2Ewum2+FnOTlGezHNx0b3/C3U258a4mP5HVvOw52s2fg2vH6ml

   s = s +
"kicAJnCbsNbDDLO2xz5PlBi3O8x0EsZD7zuG8T1vrCD5uZE7zxM+CXQNB78RKw8RRbF8Rwy

   s = s +
"wi/8iNPJMjSo0clvjMLuTHLkJTNCqVTSms2Tkq8jTzOcVP/BsePUocQLDQ9AJ3LtFUbR1Hqaiw

   s = s + "Poyf5xoojKAg"
   .Images(s)

   .Columns.Add("Column 1")
   With .Items
      .DefaultItem = .AddItem("Item 1")
      .CellImage(0, 0) = 1
      .DefaultItem = .AddItem("Item 2")
      .CellImages(0, 0) = "2,3"
   EndWith
   .EndUpdate()
EndWith
```

property G2antt.ImageSize as Long

Retrieves or sets the size of icons the control displays.

Type	Description
Long	A long expression that defines the size of icons the control displays.

By default, the ImageSize property is 16 (pixels). The ImageSize property specifies the size of icons being loaded using the Images method. The control's Images collection is cleared if the ImageSize property is changed, so it is recommended to set the ImageSize property before calling the Images method. The ImageSize property defines the size (width/height) of the icons within the control's Images collection. For instance, if the ICO file to load includes different types the one closest with the size specified by ImageSize property is loaded by Images method. The ImageSize property does NOT change the height for the control's font.

The ImageSize property defines the size to display the following UI elements:

- any icon that a cell or column displays (<img>number</img> ex-html tag, CellImage, CellImages)
- check-box or radio-buttons (CellHasCheckBox, CellHasRadioButton)
- expand/collapse glyphs (HasButtons, HasButtonsCustom)
- header's sorting or drop down-filter glyphs

property G2antt.Indent as Long

Retrieves or sets the amount, in pixels, that child items are indented relative to their parent items.

Type	Description
Long	A long expression that indicates the amount, in pixels, that child items are indented relative to their parent items.

If the Indent property is 0, the child items are not indented relative to their parent item. Use HasLines and LinesAtRoot properties to show the hierarchy lines. Use the HasButtons property to define the +/- signs appearance. Use the TreeColumnIndex property to define the index of the column that displays the hierarchy. Use the InsertItem method to insert a child item. Use the InsertControlItem property to insert an ActiveX item.

property G2antt.IsGrouping as Boolean

Indicates whether the control is grouping the items.

Type	Description
Boolean	A Boolean expression that specifies whether the control is grouping or ungrouping the items.

The IsGrouping property determines whether the control is grouping/ungrouping the items. The AllowGroupBy property specifies whether the control supports Group-By feature. For instance, during grouping, the control may expand or collapse items, you can use the IsGrouping property to determine if the BeforeExpandItem/AfterExpandItem events occur due user interaction or control's grouping operation. The GroupItem property indicates the index of the column being grouped for specified grouping item. The Group/Ungroup method groups or ungroup the control's list. During execution any of these methods, the IsGrouping property returns True. The LayoutChanged event is fired when the user changes the layout of the control, including dragging a column to the sort bar. The SortBarColumnsCount property indicates the number of the columns being grouped. The SortBarColumn property indicates the column being sorted giving its position in the sort bar.

property G2antt.ItemFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS, ColIndex as Long, HitTestInfo as HitTestInfoEnum) as HITEM

Retrieves the item from the cursor.

Type	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in client coordinates.
ColIndex as Long	A long expression that indicates on return, the column where the point belongs. If the return value is zero, the ColIndex may indicate the handle of the cell (inner cell).
HitTestInfo as HitTestInfoEnum	A HitTestInfoEnum expression that determines on return, the position of the cursor within the cell.
HITEM	A long expression that indicates the item's handle where the point is.

Use the ItemFromPoint property to get the item from the point specified by the {X,Y}. The X and Y coordinates are expressed in client coordinates, so a conversion must be done in case your coordinates are relative to the screen or to other window.

The ItemFromPoint property returns:

- the **handle** of the item from the current cursor position, if the **X** and **Y** parameters are **-1**. The ItemFromPoint property returns 0, if not item is found.
- the **number** of rows from current cursor position to the last visible item, if the **X** is **0** and **Y** parameter is **-1**. The ItemFromPoint property returns 0, if the cursor hovers any item, else it returns a positive value, that indicate the number of items between the last visible item and the current cursor position. For instance, you can use this option, to add items to the cursor, once the user clicks the empty area of the items section of the control.

Use the ColumnFromPoint property to retrieve the column from cursor. Use the DateFromPoint property to specify the date from the cursor. Use the SelectableItem property to specify the user can select an item. Use the BarFromPoint property to get the bar from the point. Use the LinkFromPoint property to get the link from the point.

The following VB sample prints the cell's caption from the cursor (if the control contains no inner cells. Use the SplitCell property to insert inner cells) :

```
Private Sub G2antt1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
   On Error Resume Next
   ' Converts the container coordinates to client coordinates
   X = X / Screen.TwipsPerPixelX
   Y = Y / Screen.TwipsPerPixelY
   Dim h As HITEM
   Dim c As Long
   Dim hit As EXG2ANTTLibCtl.HitTestInfoEnum
   ' Gets the item from (X,Y)
   h = G2antt1.ItemFromPoint(X, Y, c, hit)
   If Not (h = 0) Then
      Debug.Print G2antt1.Items.CellValue(h, c) & " HT = " & hit
   End If
End Sub
```

The following VB sample displays the cell's caption from the cursor (if the control contains inner cells):

```
Private Sub G2antt1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
   On Error Resume Next
   ' Converts the container coordinates to client coordinates
   X = X / Screen.TwipsPerPixelX
   Y = Y / Screen.TwipsPerPixelY
   Dim h As HITEM
   Dim c As Long
   Dim hit As EXG2ANTTLibCtl.HitTestInfoEnum
   ' Gets the item from (X,Y)
   h = G2antt1.ItemFromPoint(X, Y, c, hit)
   If Not (h = 0) Or Not (c = 0) Then
      Debug.Print G2antt1.Items.CellValue(h, c) & " HT = " & hit
   End If
End Sub
```

The following VB sample displays the index of icon being clicked:

```
Private Sub G2antt1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Dim i As HITEM, h As HitTestInfoEnum, c As Long
   With G2antt1
      i = .ItemFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY, c, h)
   End With
   If (i <> 0) or ( c <> 0 ) Then
      If exHTCellIcon = (h And exHTCellIcon) Then
         Debug.Print "The index of icon being clicked is: " & (h And &HFFFF0000) / 65536
      End If
   End If
End Sub
```

The following C# sample displays the caption of the cell being double clicked (including the inner cells):

```
   EXG2ANTTLib.HitTestInfoEnum hit;
   int c = 0, h = axG2antt1.get_ItemFromPoint( e.x, e.y, out c, out hit );
   if ( ( h != 0 ) || ( c != 0 ) )
      MessageBox.Show( axG2antt1.Items.get_CellValue( h, c ).ToString() );
```

The following VC sample displays the caption of the cell being clicked:

```
#include "Items.h"

static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
   if ( pv )
   {
      if ( pv->vt == VT_ERROR )
         return szDefault;

      COleVariant vt;
      vt.ChangeType( VT_BSTR, pv );
      return V_BSTR( &vt );
   }
   return szDefault;
```

```
}

void OnMouseDownG2antt1(short Button, short Shift, long X, long Y)
{
   long c = 0, hit = 0, hItem = m_g2antt.GetItemFromPoint( X, Y, &c, &hit );
   if ( ( hItem != 0 ) || ( c != 0 ) )
   {
      CItems items = m_g2antt.GetItems();
      COleVariant vtItem( hItem ), vtColumn( c );
      CString strCaption = V2S( &items.GetCellValue( vtItem, vtColumn ) ), strOutput;
      strOutput.Format( "Cell: '%s', Hit = %08X\n", strCaption, hit );
      OutputDebugString( strOutput );
   }
}
```

The following VB.NET sample displays the caption from the cell being clicked:

```
Private Sub AxG2antt1_MouseDownEvent(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_MouseDownEvent) Handles
AxG2antt1.MouseDownEvent
   With AxG2antt1
      Dim i As Integer, c As Integer, hit As EXG2ANTTLib.HitTestInfoEnum
      i = .get_ItemFromPoint(e.x, e.y, c, hit)
      If (Not (i = 0) Or Not (c = 0)) Then
         Debug.WriteLine("Cell: " & .Items.CellValue(i, c) & " Hit: " & hit.ToString())
      End If
   End With
End Sub
```

The following C# sample displays the caption from the cell being clicked:

```
private void axG2antt1_MouseDownEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_MouseDownEvent e)
{
   int c = 0;
   EXG2ANTTLib.HitTestInfoEnum hit;
   int i = axG2antt1.get_ItemFromPoint( e.x, e.y, out c,out hit );
   if ( ( i != 0 ) || ( c != 0 ) )
```

```
    {
        string s = axG2antt1.Items.get_CellValue( i,c ).ToString();
        s = "Cell: " + s + ", Hit: " + hit.ToString();
        System.Diagnostics.Debug.WriteLine( s );
    }
}
```

The following VFP sample displays the caption from the cell being clicked (the code should be in the G2antt1.MouseDown event):

```
*** ActiveX Control Event ***
LPARAMETERS button, shift, x, y

local c, hit
c = 0
hit = 0
with thisform.G2antt1
    .Items.DefaultItem = .ItemFromPoint( x, y, @c, @hit )
    if ( .Items.DefaultItem <> 0 ) or ( c <> 0 )
        wait window nowait .Items.CellValue( 0, c ) + " " + Str( hit )
    endif
endwith
```

property G2antt.Items as Items

Retrieves the control's item collection.

Type	Description
Items	An Items object that holds the control's items collection.

Use the Items property to access the Items collection. Use the Items collection to add, remove or change the control items. Use the GetItems method to get the items collection into a safe array. Use the PutItems method to load items from a safe array. Use the Columns property to access the control's Columns collection. Use the AddItem, InsertItem or InsertControlItem method to add new items to the control. Use the DataSource to add new columns and items to the control. Adding new items fails if the control has no columns. Use the Chart object to access all properties and methods related to the G2antt chart. Use the AddBar method to add bars to the item. The bars are always shown in the chart area. Use the PaneWidth property to specify the width of the chart.

property G2antt.ItemsAllowSizing as ItemsAllowSizingEnum

Retrieves or sets a value that indicates whether a user can resize items at run-time.

Type	Description
ItemsAllowSizingEnum	An ItemsAllowSizingEnum expression that specifies whether the user can resize a single item at runtime, or all items, at once.

By default, the ItemsAllowSizing property is exNoSizing. Use the ItemsAllowSizing property to specify whether all items are resizable. Use the ItemAllowSizing property of the Items object to specify only when few items are resizable or not. Use the ItemHeight property to specify the height of the item. The CellSingleLine property specifies whether a cell displays its caption using multiple lines. The DefaultItemHeight property specifies the default height of the items. The DefaultItemHeight property affects only items that are going to be added. It doesn't affect items already added.

property G2antt.Layout as String

Saves or loads the control's layout, such as positions of the columns, scroll position, filtering values.

Type	Description
String	A String expression that specifies the control's layout.

You can use the Layout property to store the control's layout and to restore the layout later. For instance, you can save the control's Layout property to a file when the application is closing, and you can restore the control's layout when the application is loaded. The Layout property saves almost all of the control's properties that user can change at runtime (like changing the column's position by drag and drop). The Layout property does NOT save the control's data, so the Layout property should be called once you loaded the data from your database, xml or any other alternative. Once the data is loaded, you can call the Layout property to restore the View as it was saved. Before closing the application, you can call the Layout property and save the content to a file for reading next time the application is opened.

The Layout property saves/loads the following information:

- chart's FirstVisibleDate property, that indicates the first visible date in the chart section
- chart's UnitScale property, that specifies the scale to display the chart
- chart's UnitWidth property, that specifies the width of the time-unit to be displayed in the chart.
- panels width, through the PaneWidth property
- columns size and position
- current selection
- scrolling position and size
- expanded/collapsed items, if any
- sorting columns
- filtering options
- SearchColumnIndex property, indicates the focusing column, or the column where the user can use the control's incremental searching.
- TreeColumnIndex property, which indicates the index of the column that displays the hierarchy lines.

These properties are serialized to a string and encoded in BASE64 format.

The following movies show how Layout works:

- The Layout property is used to save and restore the control's view.

Generally, the Layout property can be used to save / load the control's layout (or as it is

displayed). Thought, you can benefit of this property to sort the control using one or more columns as follows:

- multiplesort="";singlesort="", removes any previously sorting
- multiplesort="C3:1", sorts ascending the column with the index 3 (and add it to the sort bar if visible)
- singlesort="C4:2", sorts descending the column with the index 4 (it is not added to sort bar panel)
- multiplesort="C3:1";singlesort="C4:2", sorts ascending the column with the index 3 (and add it to the sort bar if visible), and sorts descending the column with the index 4. In other words, it re-sort the control by columns 3 and 4.
- multiplesort="C3:1 C5:2";singlesort="C4:2", sorts ascending the column with the index 3 (and add it to the sort bar if visible), sorts descending the column with the index 5 (and add it to the sort bar if visible), and sorts descending the column with the index 4. In other words, it re-sort the control by columns 3, 5 and 4.

The format of the Layout in non-encoded form is like follows:

```
c0.filtertype=0
c0.position=0
c0.select=0
c0.visible=1
c0.width=96
....
columns=13
collapse="0-3 5-63 80-81 83"
filterprompt=""
focus=8
focuscolumnindex=0
hasfilter=1
hscroll=0
multiplesort="C12:1 C2:2"
searchcolumnindex=3
select="39 2 13 8"
selectcolumnindex=0
singlesort="C5:2"
treecolumnindex=0
vscroll=12
vscrolloffset=0
```

property G2antt.LinesAtRoot as LinesAtRootEnum

Link items at the root of the hierarchy.

Type	Description
LinesAtRootEnum	A LinesAtRootEnum expression that indicates whether the control link items at the root of the hierarchy.

The control paints the hierarchy lines to the right if the Column's Alignment property is RightAlignment. The TreeColumnIndex property specifies the index of column where the hierarchy lines are painted. Use the Indent property to increase or decrease the amount, in pixels, that child items are indented relative to their parent items. Use the HasLines property to enhances the graphic representation of a tree control's hierarchy by drawing lines that link child items to their corresponding parent item. Use the InsertItem method to insert a child item. Use the InsertControlItem property to insert an ActiveX item.

method G2antt.LoadXML (Source as Variant)

Loads an XML document from the specified location, using MSXML parser.

Type	Description
Source as Variant	An indicator of the object that specifies the source for the XML document. The object can represent a file name, a URL, an IStream, a SAFEARRAY, or an IXMLDOMDocument.

Return	Description
Boolean	A boolean expression that specifies whether the XML document is loaded without errors. If an error occurs, the method retrieves a description of the error occurred.

The LoadXML method uses the MSXML (MSXML.DOMDocument, XML DOM Document) parser to load XML documents, previously saved using the SaveXML method. The control is emptied when the LoadXML method is called, and so the columns and items collection are emptied before loading the XML document. The LoadXML method adds a new column for each **<column>** tag found in the **<columns>** collection. Properties like Caption, HTMLCaption, Image, Visible, LevelKey, DisplayFilterButton, DisplayFilterPattern, FilterType, Width and Position are fetched for each column found in the XML document. The control fires the AddColumn event for each found column. The **<items>** xml element contains a collection of **<item>** objects. Each <item> object holds information about an item in the control, including its cells, child items or bars. Each item contains a collection of **<cell>** objects that defines the cell for each column. The **<bars>** element contains a collection of **<bar>** each one is associated with the bars in the item. The Expanded attribute specifies whether an item is expanded or collapsed, and it carries the value of the ExpandItem property. The **<chart>** element contains data related to the chart data of the control. For instance, it includes the collection of levels being displayed in the chart, the first visible date, links and groups of bars. The **<levels>** element holds a collection of **<level>** objects each one being associated with an level in the chart area. The **<links>** element holds a collection of **<link>** objects each one indicating a link between two bars in the chart. The **<groups>** element holds a collection of **<group>** objects that indicates the bars that are grouped in the chart.

The XML format looks like follows:

```
- <Content Author Component Version ...>
  - <Chart FirstVisibleDate ...>
    - <Levels>
        <Level Label Unit Count />
```

```
        <Level Label Unit Count />
        ...
      </Levels>
    - <Links>
        <Link Key StartItem StartBar EndItem EndBar Visible StartPos EndPos Color Style
Width ShowDir Text ... />
        <Link Key StartItem StartBar EndItem EndBar Visible StartPos EndPos Color Style
Width ShowDir Text ... />
        ...
      </Links>
    - <Groups>
        <Group ItemA KeyA StartA ItemB KeyB StartB />
        <Group ItemA KeyA StartA ItemB KeyB StartB />
        ...
      </Groups>
  </Chart>
- <Columns>
    <Column Caption Position Width HTMLCaption LevelKey DisplayFilterButton
DisplayFilterPatter FilterType ... />
    <Column Caption Position Width HTMLCaption LevelKey DisplayFilterButton
DisplayFilterPatter FilterType ... />
    ...
  </Columns>
- <Items>
  - <Item Expanded ...>
      <Cell Value ValueFormat Images Image ... />
      <Cell Value ValueFormat Images Image ... />
      ...
    - <Bars>
        <Bar Name Start End Caption HAlignCaption VAlignCaption Key ... />
        <Bar Name Start End Caption HAlignCaption VAlignCaption Key ... />
        ...
      </Bars>
    - <Items>
      - <Item Expanded ...>
      - <Item Expanded ...>
      ....
```

```
        </Items>
      </Item>
    </Items>
  </Content>
```

property G2antt.MarkSearchColumn as Boolean

Retrieves or sets a value that indicates whether the searching column is marked or unmarked

Type	Description
Boolean	A boolean expression that indicates whether the searching column is marked or unmarked.

The control supports incremental search feature. The MarkSearchColumn property specifies whether the control highlights the searching column. Use the SearchColumnIndex property to specify the index of the searching column. The user can change the searching column by pressing the TAB ort Shift + TAB key. Use the AutoSearch property to specify whether the control enables the incremental searching feature. Use the AutoSearch property to specify the type of incremental searching the control supports within the column. Use the UseTabKey property to specify whether the control uses the TAB key.

property G2antt.MarkTooltipCells as Boolean

Retrieves or sets a value that indicates whether the control marks the cells that have tool tips.

Type	Description
Boolean	A boolean expression that indicates whether the control marks the cells that have tool tips.

By default, the MarkTooltipCells property is False. If the MarkTooltipCells property is True, the control paints on the right side of the cell a sign that indicates that the cell has associated a tool tip. Use the CellTooltip property to associate a tool tip to a cell. Use the TooltipCellsColor property to change the color used to sign cells that have tool tips. Use the MarkTooltipCellsImage property to assign a different look for signs to mark the cells that have tooltips.

property G2antt.MarkTooltipCellsImage as Long

Specifies a value that indicates the index of icon being displayed in the cells that have tooltips.

Type	Description
Long	A long expression that indicates the index of icon being displayed when a cell has tooltip assigned.

By default, the MarkTooltipCellsImage property is 0. The MarkTooltipCellsImage property has effect only if the MarkTooltipCells property is True. Use the Images method to assign new icons to the control. By default, if the control can't find the icon specified by the MarkTooltipCellsImage property, it paints the default sign to mark the cells that have the tooltips.

The following screen shot shows how the control marks by default, the cells that have tooltip:

The following screen shot shows how the control marks the cells that have tooltip, once that MarkTooltipCellsImage property points to an icon in the Images collection:

method G2antt.OLEDrag ()

Causes a component to initiate an OLE drag/drop operation.

Type	Description

Only for internal use.

property G2antt.OLEDropMode as exOLEDropModeEnum

Returns or sets how a target component handles drop operations

Type	Description
exOLEDropModeEnum	An exOLEDropModeEnum expression that indicates the OLE Drag and Drop mode.

By default, the OLEDropMode property is exOLEDropNone. Use the AutoDrag property to specify what the control does when the user clicks and drag the items.

Currently, the ExG2antt control supports only manual OLE Drag and Drop operation. Use the Background(exDragDropBefore) property to specify the visual appearance for the dragging items, before painting the items. Use the Background(exDragDropAfter) property to specify the visual appearance for the dragging items, after painting the items. Use the Background(exDragDropList) property to specify the graphic feedback for the item from the cursor, while the OLE drag and drop operation is running. See the OLEStartDrag and OLEDragDrop events for more details about implementing drag and drop operations into the ExG2antt control.

In the /NET Assembly, you have to use the AllowDrop property as explained here:

- *https://www.exontrol.com/sg.jsp?content=support/faq/net/#dragdrop*

property G2antt.OnResizeControl as OnResizeControlEnum

Specifies whether the list or the chart part is resized once the control is resized.

Type	Description
OnResizeControlEnum	An OnResizeControlEnum expression that specifies whether the list or the chart part of the control is resized, when the entire control is resized.

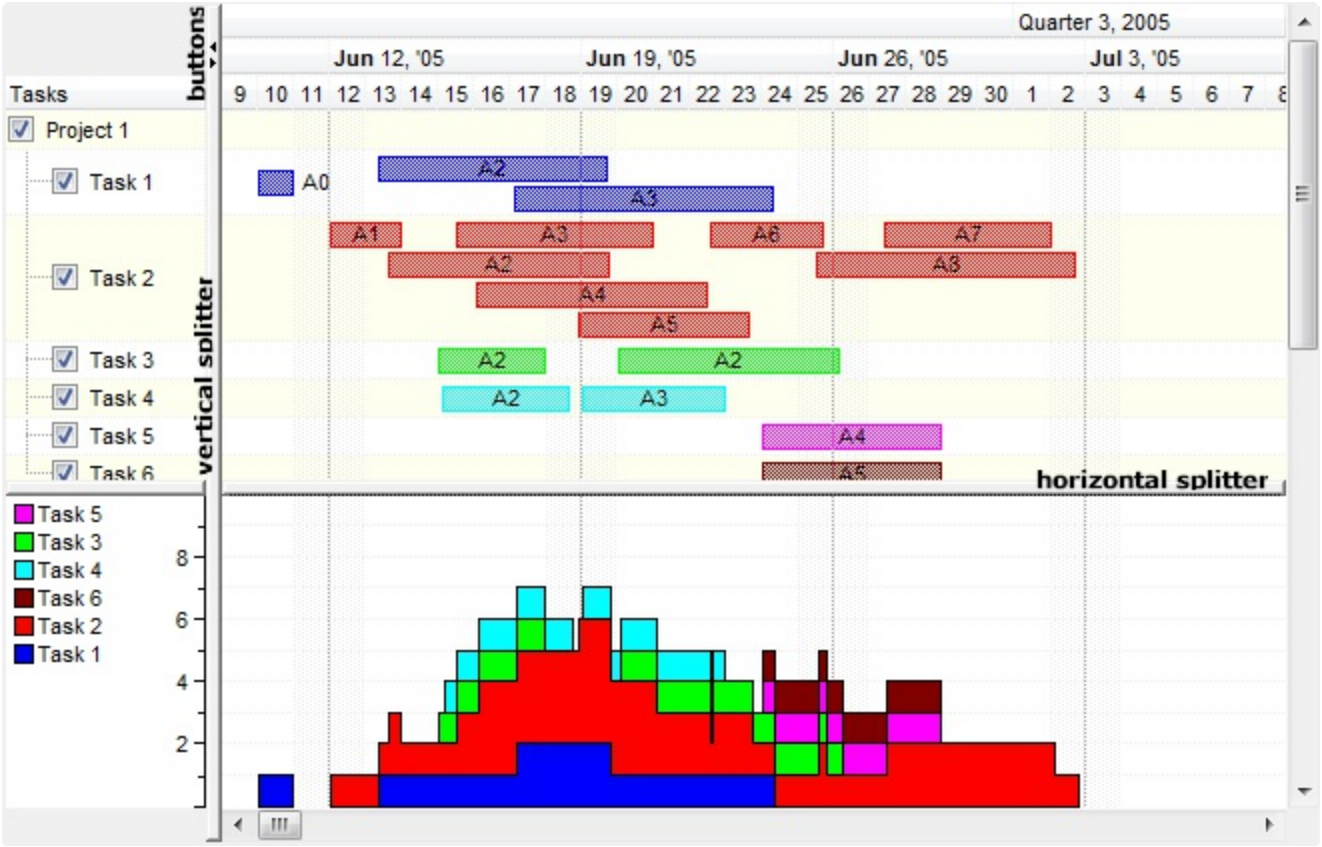
By default, the OnResizeControl property is exResizeList. In other words, the list part of the control (the part that lists the columns) gets resized, and the chart are stay fixed. Use the OnResizeControl to specify whether the chart area should be resized when the user resizes the control (whenever the chart is anchored to a form). Use the PaneWidth property to specify the width of the list or chart part of the control. The HistogramBoundsChanged event notifies your application when the location and the size of the chart's histogram is changed, so you can use it to add your legend for the histogram in a panel component. The controls vertical splitter is hidden if the OnControlResize property is exResizeChart + exDisableSplitter (129) and the PaneWidth(False) property is 0.

Use the OnResizeControl property to allow:

- resizing the list area when the control is resized (by default)
- resizing the chart area when the control is resized
- displaying the resizing buttons on the vertical splitter to allow quickly hide/show the items/chart area.
- hiding the vertical splitter

You can also use the OnResizeControl property to prevent:

- resizing the list/chart using the control's splitter.
- resizing the chart's histogram.



The following VB sample shows how can I disable the control's splitter so the user can't resize the chart area:

```
With G2antt1
   .OnResizeControl = exDisableSplitter
   .Chart.PaneWidth(1) = 60
End With
```

The following VB.NET sample shows how can I disable the control's splitter so the user can't resize the chart area:

```
With AxG2antt1
   .OnResizeControl = EXG2ANTTLib.OnResizeControlEnum.exDisableSplitter
   .Chart.PaneWidth(1) = 60
End With
```

The following C++ sample shows how can I disable the control's splitter so the user can't resize the chart area:

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'
```

```
   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->PutOnResizeControl(EXG2ANTTLib::exDisableSplitter);
spG2antt1->GetChart()->PutPaneWidth(1,60);
```

The following C# sample shows how can I disable the control's splitter so the user can't resize the chart area:

```
axG2antt1.OnResizeControl = EXG2ANTTLib.OnResizeControlEnum.exDisableSplitter;
axG2antt1.Chart.set_PaneWidth(1 != 0,60);
```

The following VFP sample shows how can I disable the control's splitter so the user can't resize the chart area:

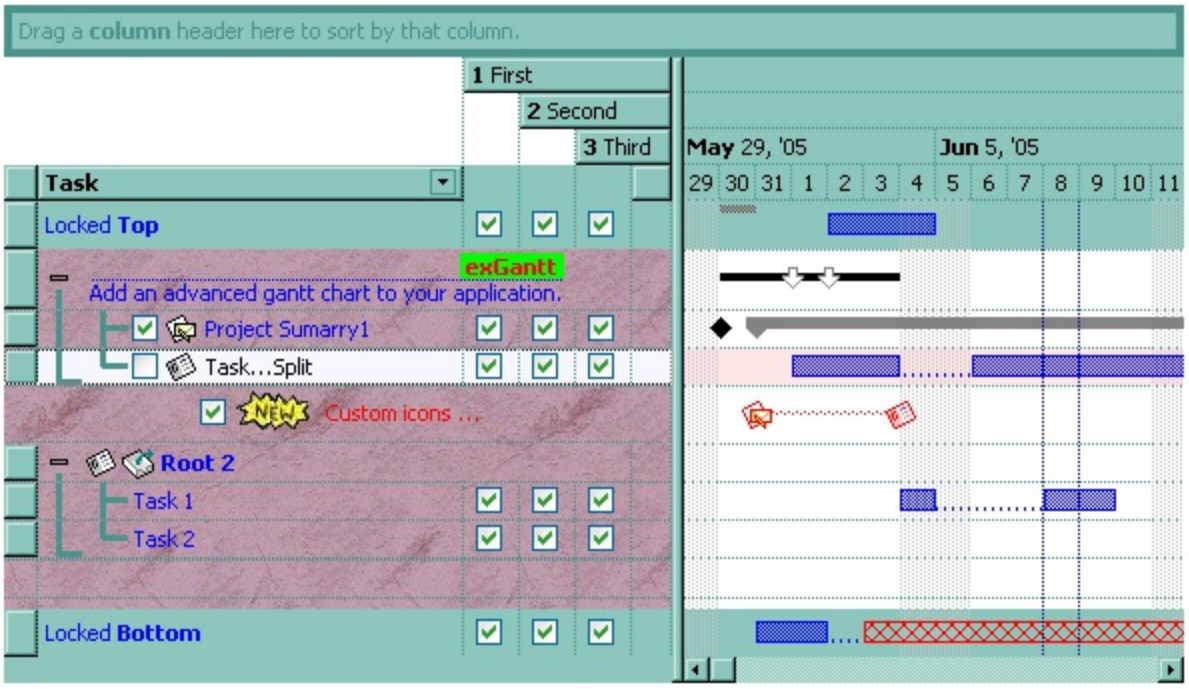
```
with thisform.G2antt1
   .OnResizeControl = 128
   .Chart.PaneWidth(1) = 60
endwith
```

property G2antt.Picture as IPictureDisp

Retrieves or sets a graphic to be displayed in the control.

Type	Description
IPictureDisp	A Picture object that's displayed on the control's background.

By default, the control has no picture associated. The control uses the PictureDisplay property to determine how the picture is displayed on the control's background. Use the PictureLevelHeader property to specify the picture on the control's levels header bar. Use the CellPicture property to assign a picture to a cell. Use the BackColor property to specify the control's background color. Use the Picture property to assign a picture to the chart area.



property G2antt.PictureDisplay as PictureDisplayEnum

Retrieves or sets a value that indicates the way how the graphic is displayed on the control's background

Type	Description
PictureDisplayEnum	A PictureDisplayEnum expression that indicates the way how the picture is displayed.

By default, the PictureDisplay property is exTile. The PictureDisplay property specifies how the Picture is displayed on the control's background. If the control has no picture associated the PictureDisplay property has no effect. Use the CellPicture property to assign a picture to a cell. Use the BackColor property to specify the control's background color.

property G2antt.PictureDisplayLevelHeader as PictureDisplayEnum

Retrieves or sets a value that indicates the way how the graphic is displayed on the control's header background.

Type	Description
PictureDisplayEnum	A PictureDisplayEnum expression that indicates the way how the picture is displayed on the control's header.

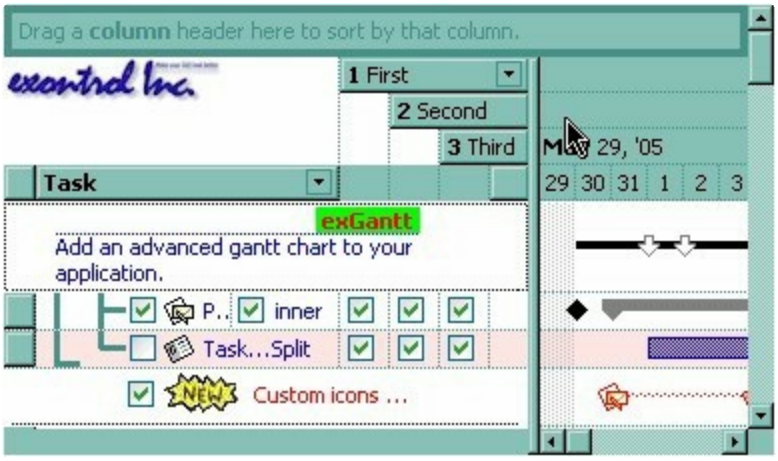
Use the PictureDisplayLevelHeader property to arrange the picture on the control's multiple levels header bar. Use the PictureLevelHeader property to load a picture on the control's header bar when it displays multiple levels. The control's header bar displays multiple levels if there are two or more neighbor columns with the same non empty level key. Use the LevelKey property to specify the control's level key.

property G2antt.PictureLevelHeader as IPictureDisp

Retrieves or sets a graphic to be displayed in the control's header when multiple levels is on.

Type	Description
IPictureDisp	A Picture object being displayed on the control's header bar when multiple levels is on.

Use the PictureLevelHeader property to display a picture on the control's header bar when it displays the columns using multiple levels. Use the PictureDisplayLevelHeader property to arrange the picture on the control's multiple levels header bar. The control's header bar displays multiple levels if there are two or more neighbor columns with the same non empty level key. Use the LevelKey property to specify the control's level key. Use the Picture property to display a picture on the control's list area. Use the BackColorLevelHeader property to specify the background color for parts of the control's header bar that are not occupied by column's headers.



method G2antt.PutItems (Items as Variant, [Parent as Variant])

Adds data to the control from a SafeArray containing numbers, strings, dates, or nested SafeArrays of numbers, strings, and dates, positioning them as child items of the specified parent item

Type	Description
Items as Variant	An array that control uses to fill with. The array can be one or two- dimensional. If the array is one-dimensional, the control requires one column being added before calling the PutItems method. If the Items parameter indicates a two-dimensional array, the first dimension defines the columns, while the second defines the number of items to be loaded. For instance, a(2,100) means 2 columns and 100 items. For instance: • PutItems Array("Item 1", "Item 2", "Item 3"), adds the rows at the end of the list • PutItems Array("Root", Array("Child 1", "Child 2")), adds data in a hierarchical structure, at the end of the list • PutItems rs.GetRows(), appends data from a recordset using the GetRows method of the Recordset • PutItems rs.GetRows(10), inserts the first 10 records from a Recordset using the GetRows method, at the end of the list where GetRows() method in ADO retrieves multiple records from a Recordset object and stores them in a two-dimensional array.
	Indicates one of the following: • missing, empty or 0 {number}, specifies that the data(Items) is being appended (added to the end of the list) • a long expression, that specifies the handle of the item where the array is being inserted • a string expression of of "parent;IDColumn;ParentIDColumn" format, where,

Parent as Variant	'parent' denotes the handle of the item where the data is being inserted, 'IDColumn' refers to the index of the column containing row identifiers, and 'ParentIDColumn' indicates the index of the column containing identifiers of parent rows. This way, you can insert data hierarchically using parent-id relationship. A parent-id relationship is a way of organizing data in a hierarchical structure where each element (or "child") is associated with a parent element. Please be aware that the rows of the data are inserted as they were provided by the Items parameter. Therefore, it is important that the data provided be sorted by the IDColumn so that the parent row referred to by the ParentIDColumn value is already present and can be used to insert the current row as a child of it. For instance: • PutItems Array("Item 1", "Item 2", "Item 3"), Items.ItemByIndex(2), inserts the rows as children of the item with index 2 • PutItems Array("Root", Array("Child 1", "Child 2")), Items.FirstVisibleItem, Inserts data as a hierarchical structure, placing it as a child of the first visible item • PutItems rs.GetRows(), Items.ItemByIndex(0), inserts the records from the recordset using the GetRows method of the Recordset, placing them as children of the item with index 0 • PutItems rs.GetRows(), ";0;3", inserts the records from the recordset using the GetRows method of the Recordset, utilizing parent-child relationships. The first column (index 0) contains the identifiers of the rows, while the fourth column (index 3) contains the keys of the parent rows. where GetRows() method in ADO retrieves multiple records from a Recordset object and stores them in a two-dimensional array.

The PutItems method loads items from a safe array. The PutItems method may raise one of the following exceptions:

- The array dimension exceeds 2 (In simpler terms, a two-dimensional array (or 2D array) is like a table with rows and columns. If an array exceeds 2 dimensions, it means it has three or more dimensions, such as a 3D array (which can be thought of as a collection of tables) or even higher dimensions) You need to provide a one-dimensional or two-dimensional array
- The number of columns does not match the array size (either the control has no columns or the number of columns is too small). You need to add more columns (Add property).
- The element type of the array is not valid (the type of the array is either unknown or not supported) You need to provide a valid type, which must be one of the following: Variant, String, Integer, Long, Double, Float, or Date.

The PutItems method performs:

1. **Insertion Order**: The data is inserted into the system in the same order as it is provided by the Items parameter. This means that the sequence of rows in the Items parameter directly affects how the data is inserted.
2. **Sorting Requirement**: To ensure correct insertion, it's crucial that the data is sorted by the IDColumn (when the Parent parameter is of "parent;IDColumn;ParentIDColumn" format). This sorting ensures that parent rows are inserted before their corresponding child rows.
3. **Parent-Child Relationship**: The sorting ensures that when a row refers to a parent row using the ParentIDColumn value (when the Parent parameter is of "parent;IDColumn;ParentIDColumn" format). The parent row is already present in the control. This allows the current row to be inserted as a child of the parent row without encountering errors or inconsistencies.

In essence, by sorting the data appropriately, you establish a clear hierarchy where parent rows are inserted before child rows, maintaining the integrity of the parent-child relationships within the dataset.

For instance, let's say we have the following data:

EmployeeID	EmployeeName	DepartmentID	ParentID
1	John	101	
2	Alice	102	1
3	Bob	101	1
4	Sarah	102	1
5	Emma	101	2
6	Mike	102	2

Each row represents an employee.

- EmployeeID uniquely identifies each employee (represents the column with the index 0)
- EmployeeName denotes the name of the employee (represents the column with the index 1)
- DepartmentID indicates the department to which the employee belongs (represents the column with the index 2)
- ParentID establishes the relationship between employees (represents the column with the index 3), where it references the EmployeeID of the parent employee. An empty value indicates the absence of a parent, typically representing the head of the department.

Having this data organized into a two-dimensional array, the statement PutItems d loads it as a flat table:

EmployeeID	EmployeeName	DepartmentID	ParentID
1	John	101	
2	Alice	102	1
3	Bob	101	1
4	Sarah	102	1
5	Emma	101	2
6	Mike	102	2

whereas PutItems d, ";0;3" loads it as a tree structure:

EmployeeID	EmployeeName	DepartmentID	ParentID
1	⊟ John	101	
2	⊟ Alice	102	1
5	Emma	101	2
6	Mike	102	2
3	Bob	101	1
4	Sarah	102	1

where d is an array as defined next:

```
Dim d(3, 5) As Variant
d(0, 0) = "1": d(1, 0) = "John": d(2, 0) = "101": d(3, 0) = ""
d(0, 1) = "2": d(1, 1) = "Alice": d(2, 1) = "102": d(3, 1) = "1"
d(0, 2) = "3": d(1, 2) = "Bob": d(2, 2) = "101": d(3, 2) = "1"
d(0, 3) = "4": d(1, 3) = "Sarah": d(2, 3) = "102": d(3, 3) = "1"
d(0, 4) = "5": d(1, 4) = "Emma": d(2, 4) = "101": d(3, 4) = "2"
d(0, 5) = "6": d(1, 5) = "Mike": d(2, 5) = "102": d(3, 5) = "2"
```

Use the GetItems method to get a safe array with the items in the control. The PutItems method fires AddItem event for each item added to Items collection. Use the Items property to access the items collection. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or

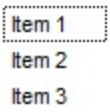
the value of a formula.

The following VB6 sample loads a flat array to a single column control (and shows as in the following picture):

```
With G2antt1
   .BeginUpdate
   .Columns.Add "Column 1"
   .PutItems Array("Item 1", "Item 2", "Item 3")
   .EndUpdate
End With
```

or similar for /NET Assembly version:

```
 With Exg2antt1
   .BeginUpdate()
   .Columns.Add("Column 1")
   .PutItems(New String() {"Item 1", "Item 2", "Item 3"})
   .EndUpdate()
End With
```



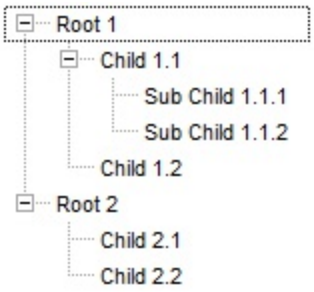
The following VB6 sample loads a hierarchy to a single column control (and shows as in the following picture):

```
With G2antt1
   .BeginUpdate
      .LinesAtRoot = exLinesAtRoot
      .Columns.Add ""
      .PutItems Array("Root 1", Array("Child 1.1", Array("Sub Child 1.1.1", "Sub Child 1.1.2"),
"Child 1.2"), "Root 2", Array("Child 2.1", "Child 2.2"))
   .EndUpdate
End With
```

or similar for /NET Assembly version:

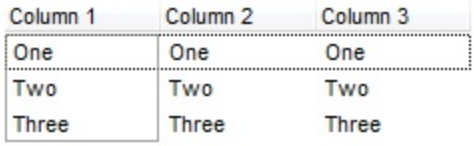
```
With Exg2antt1
   .BeginUpdate()
```

```
    .LinesAtRoot = exontrol.EXG2ANTTLib.LinesAtRootEnum.exLinesAtRoot
    .Columns.Add("")
    .PutItems(New Object() {"Root 1", New Object() {"Child 1.1", New String() {"Sub Child 1.1.1", "Sub Child 1.1.2"}, "Child 1.2"}, "Root 2", New String() {"Child 2.1", "Child 2.2"}})
    .EndUpdate()
End With
```



The following VB6 sample loads a list of items, in a three columns control (as shown in the following picture):

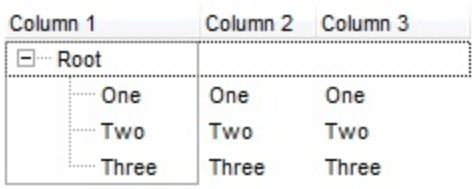
```
Dim v(2, 2) As String
v(0, 0) = "One": v(0, 1) = "Two": v(0, 2) = "Three"
v(1, 0) = "One": v(1, 1) = "Two": v(1, 2) = "Three"
v(2, 0) = "One": v(2, 1) = "Two": v(2, 2) = "Three"

With G2antt1
    .BeginUpdate
    .Columns.Add "Column 1"
    .Columns.Add "Column 2"
    .Columns.Add "Column 3"

    .PutItems v
    .EndUpdate
End With
```



The following VB6 sample loads a list of items, in a three columns control (as shown in the following picture):

```
Dim v(2, 2) As String
```

```
v(0, 0) = "One": v(0, 1) = "Two": v(0, 2) = "Three"
v(1, 0) = "One": v(1, 1) = "Two": v(1, 2) = "Three"
v(2, 0) = "One": v(2, 1) = "Two": v(2, 2) = "Three"

With G2antt1
   .BeginUpdate
   .Columns.Add "Column 1"
   .Columns.Add "Column 2"
   .Columns.Add "Column 3"

   .Items.AddItem "Root"

   .PutItems v, .Items.FirstVisibleItem
   .EndUpdate
End With
```



The following VB sample loads the collection of records from an ADO recordset:

```
Dim rs As Object
Const dwProvider = "Microsoft.Jet.OLEDB.4.0" ' OLE Data provider
Const nCursorType = 3 ' adOpenStatic
Const nLockType = 3 ' adLockOptimistic
Const nOptions = 2 ' adCmdTable
Const strDatabase = "D:\Program Files\Microsoft Visual Studio\VB98\NWIND.MDB"

'Creates an recordset and opens the "Employees" table, from NWIND database
Set rs = CreateObject("ADODB.Recordset")
rs.Open "Employees", "Provider=" & dwProvider & ";Data Source= " & strDatabase,
nCursorType, nLockType, nOptions
With G2antt1
   .BeginUpdate

      .ColumnAutoResize = False
```

```
        .MarkSearchColumn = False
        .DrawG2anttLines = True
        ' Adds a column for each field found
        With .Columns
            Dim f As Object
            For Each f In rs.Fields
                .Add f.Name
            Next
        End With

        ' Loads the collection of records
        .PutItems rs.GetRows()

        'Changes the editor of the "Photo" column
        .Columns("Photo").Editor.EditType = PictureType
    .EndUpdate
End With
```

The following C++ sample loads records from an ADO recordset, using the PutItems method:

```
#include "Items.h"
#include "Columns.h"
#include "Column.h"



#pragma warning( disable : 4146 )
#import <msado15.dll> rename ( "EOF", "adoEOF" )
using namespace ADODB;



_RecordsetPtr spRecordset;
if ( SUCCEEDED( spRecordset.CreateInstance( "ADODB.Recordset") ) )
{
   // Builds the connection string.
   CString strTableName = "Employees", strConnection =
"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=";
   CString strPath = "D:\\Program Files\\Microsoft Visual Studio\\VB98\\NWIND.MDB";
```

```
    strConnection += strPath;
    try
    {
        // Loads the table
        if ( SUCCEEDED( spRecordset->Open(_variant_t( (LPCTSTR)strTableName ),
_variant_t((LPCTSTR)strConnection), adOpenStatic, adLockPessimistic, NULL ) ) )
        {
            m_g2antt.BeginUpdate();
            m_g2antt.SetColumnAutoResize( FALSE );
            CColumns columns = m_g2antt.GetColumns();
            for ( long i = 0; i < spRecordset->Fields->Count; i++ )
                columns.Add( spRecordset->Fields->GetItem(i)->Name );
            COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
            m_g2antt.PutItems( &spRecordset->GetRows(-1), vtMissing );
            m_g2antt.EndUpdate();
        }
    }
    catch ( _com_error& e )
    {
        AfxMessageBox( e.Description() );
    }
}
```

The sample uses the #import statement to import ADODB recordset's type library. The sample enumerates the fields in the recordset and adds a new column for each field found. Also, the sample uses the GetRows method of the ADODB recordset to retrieves multiple records of a Recordset object into a safe array. Please consult the ADODB documentation for the GetRows property specification.

method G2antt.PutRes (ResHandle as Long, Type as PutResEnum)

The PutRes method associates an eXG2antt (Source) control with another eXG2antt (Target) control, using the Items.ItemBar(exBarResources).

Type	Description
ResHandle as Long	A Handle expression being returned by the ResHandle property.
Type as PutResEnum	A PutResEnnum expression that indicates whether the control loads or saves the bar's resources to or from another control.

The eXG2antt component (Target) can display resources being used by another eXG2antt component (Source). The PutRes method associates an eXG2antt (Source) control with another eXG2antt (Target) control, using the Items.ItemBar(exBarResources). The Source displays the bar and the resources to be used by each bar, while the Target shows the Resources being used by each bar found in the Source. A resource is identified by a name like R1, or a name followed by its usage in percent, like R1[20%], which indicates that the bar uses the resource R1 on 20% of its full capacity.

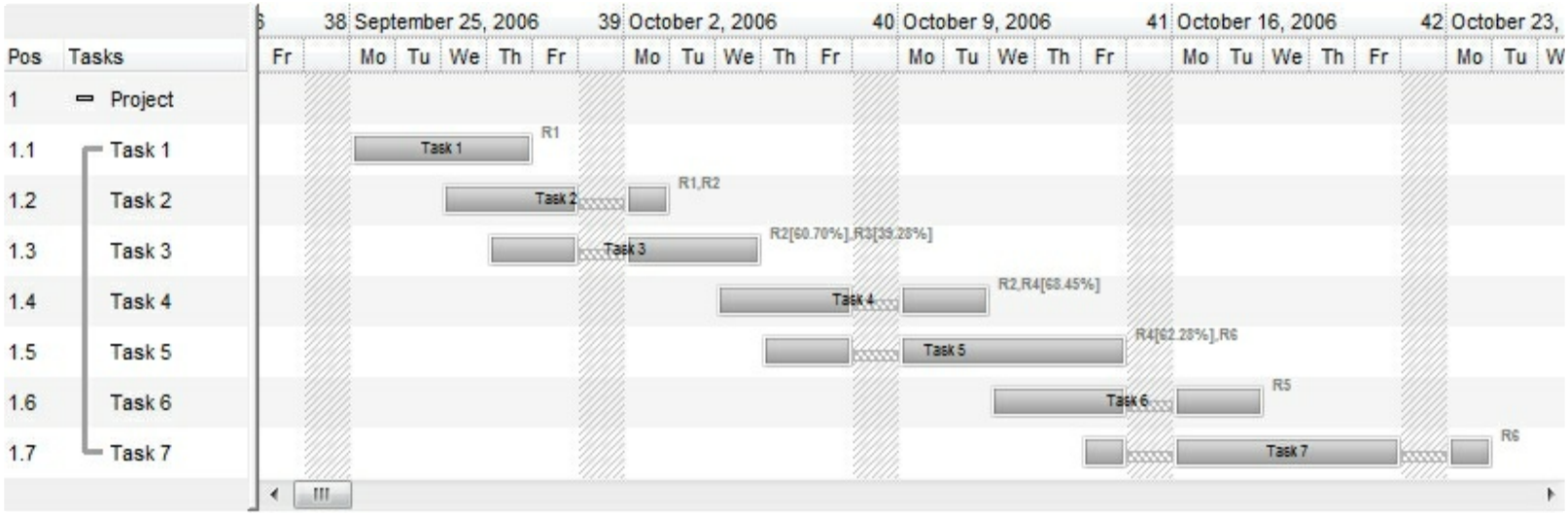
In order to use the PutRes method the Source control must:

- specify the activity/bar's resources using the ItemBar(exBarResources) property

Bellow you can find:

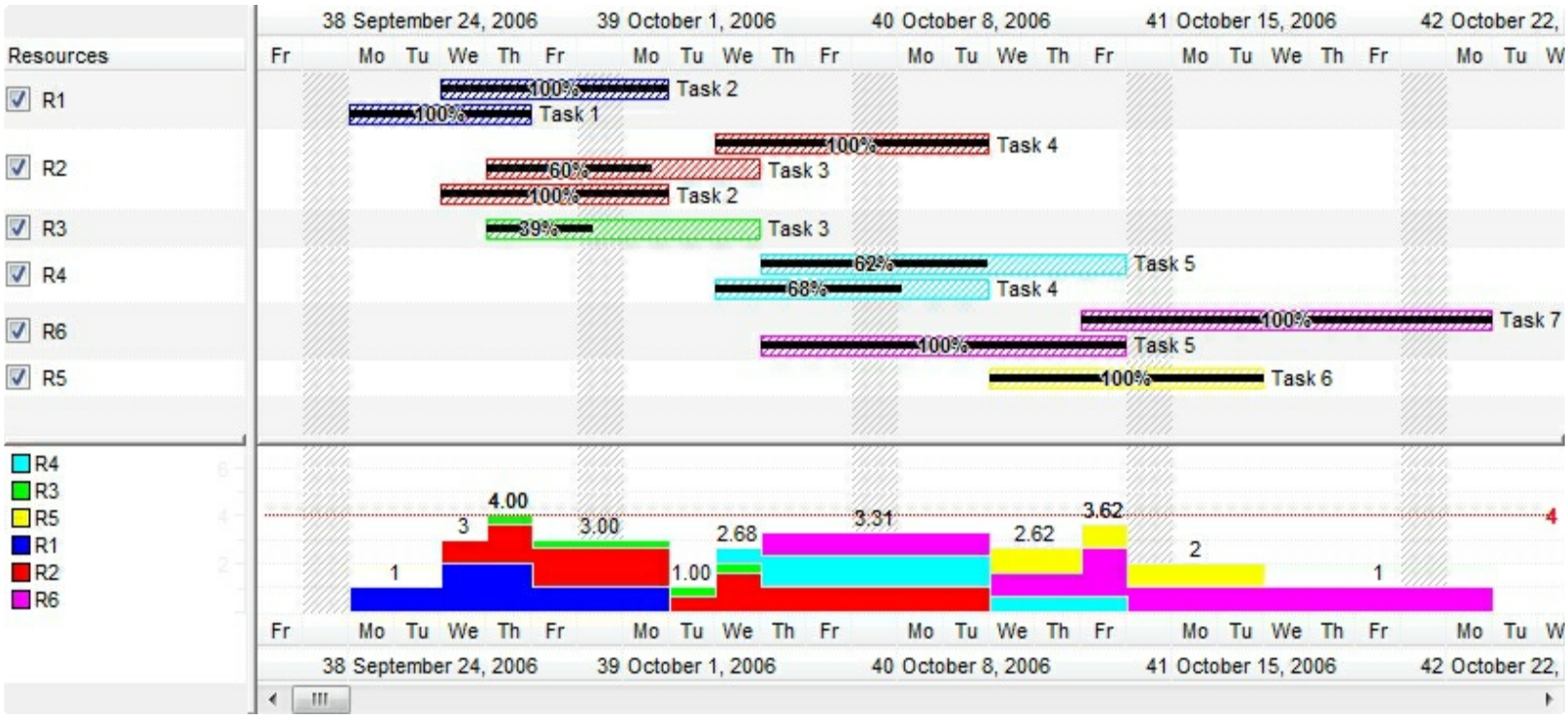
- How to display the resources from the Source to the Target control (exPutResLoad)
- How to display a histogram of using resources in the Target control
- How to update the Source from the Target once it has been updated (exPutResSave)

The following sample shows the bar's allocation of resources: (for instance the Task 1 uses the R1, Task 2 uses the R1 and R2, while Task 3 uses R2 on 60% and R3 on 39%, and so on)



(source, picture 1)

while the next picture shows the resource usage being taken from the first picture (for instance, the R1 is being used by Task 1 and Task 2, R2 is used by Task 2, Task 3 on 60% and Task 4, and so on) :



(target, picture 2)

The Source (picture 1) is the original control that displays your activities / bars. Use the Source.PutRes(Target.ResHandle, exPutResSave) to update the bar's resources in a Source control, from a Target control. IN a Source control, you can use the following properties to specify the usage of resources.

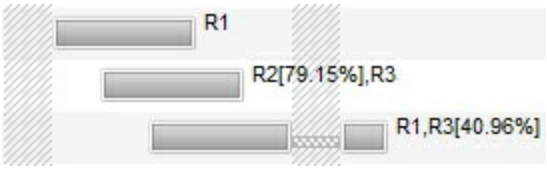
- **exBarResources** property (get/set) indicates the resources to be used by the current bar in the Source, as a string expression. The exBarResources property is a string

expression that indicate the list of resources (including its usage, or 100% if missing). The resources are separated by , (comma) character, while the usage is specified as a double expression (using the . dot character as a decimal separator). *For instance the "Resource1,Resource2,Resource3" indicates that the bar uses the Resource1,Resource2,Resource3, while "R1,R2[50%],R3[67.89%]" specifies that the bar uses the R1 on 100%, R2 on 50% and R3 on 67.89%.*

You can use the exBarCaption to display the bar's resources using the <%=formula%> format, like in the following VB sample:

```
With G2antt1.Chart.Bars("Task")
   .Def(EXG2ANTTLibCtl.ItemBarPropertyEnum.exBarCaption) = "<%=%" &
EXG2ANTTLibCtl.ItemBarPropertyEnum.exBarResources & "%>"
   .Def(EXG2ANTTLibCtl.ItemBarPropertyEnum.exBarHAlignCaption) = 18
End With
```

In other words, the sample allows you to display the bar's exBarResources property as shown bellow:



The set exBarResources property could be used in the following format based on the first character as listed:

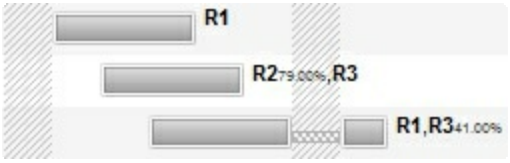
- If the first character is **+(plus)**, the rest of the expression indicates the resources to be assigned to the current bar. *For instance, if the current bar has the exBarResources property as "R1,R2" , and we call set exBarResources as "+R3", it means that the R3 is added to the bar's resources, and so the new exBarResources property is "R1,R2,R3".*
- If the first character is **-(minus)**, the rest of the expression indicates the resources to be removed from the current bar. *For instance, if the current bar has the exBarResources property as "R1,R2" , and we call set exBarResources as "-R2", it means that the R2 is removed from the bar's resources, and so the new exBarResources property is "R1".*
- If no +,- character, the new expression replaces the exBarResources property. *For instance, if the current bar has the exBarResources property as "R1,R2" , and we call set exBarResources as "R3,R4", it means that the new exBarResources property is "R3,R4".*

- **exBarResourceFormat** property (get/set) indicates the format or the expression to be

used if you need to display the bar's resource in a different format, in Source, as a string expression. The expression supports the **name** keyword which indicates the name of the resource, and the **percent** keyword to get the usage percent a a double expression between 0 and 1. The expression supports all predefined functions listed here. Use the exBarResource**s**Format (NOT exBarResourceFormat, whith no s) to get the HTML value of the formatted string using the bar's resources. *For instance, let's say we need to display the resource names in bold, and the usage percent in a smaller font and a different foreground color, so the Items.ItemBar(exBarResourceFormat) property could be "`<b>` + name + `</b><font ;5><fgcolor=404040>` + (percent = 1 ? `` : (round(100*percent) format ``) + `%`) + `</fgcolor></font>`", and so the VB sample could show as:*

```
With G2antt1.Chart.Bars("Task")
   .Def(exBarResourceFormat) = "`<b>` + name + `</b><font ;5>
<fgcolor=404040>` + (percent = 1 ? `` : (round(100*percent) format ``) + `%` ) +
`</fgcolor></font>`"
   .Def(EXG2ANTTLibCtl.ItemBarPropertyEnum.exBarCaption) = "<%=%" &
EXG2ANTTLibCtl.ItemBarPropertyEnum.exBarResourcesFormat & "%>"
   .Def(EXG2ANTTLibCtl.ItemBarPropertyEnum.exBarHAlignCaption) = 18
End With
```

In other words, the sample allows you to display the bar's exBarResources property as shown bellow:



- **exBarResourcesFormat** property (get only) returns formatted expression of the exBarResources using the **exBarResourceFormat** (no **s**, so it is exBarResourceFormat, not exBarResource**s**Format).
- **exBarResourcesNames** property (get only) returns the name of each resource to be used by the current bar, in the Source, as a string expression. This option returns no percent or usage of any resource. *For instance, if the exBarResources property is "R3[67.89%],R4[23.23%]", the exBarResourcesNames property gets the "R3,R4".*
- **exBarResourcesUsages** property (get only) returns the usage (double expression from 0 to 1) of each resource to be used by the current bar, in the Source, as a string expression. This option returns no name any resource. *For instance, if the exBarResources property is "R3[67.89%],R4[23.23%]", the exBarResourcesUsages*

property gets the "0.6789,0.2323".

The Target (picture 2) is the control that displays the Resources column, where all resources found are being added line by line, and its usage on items. Use the Target.PutRes(Source.ResHandle, exPutResLoad) to update the Target control from a Source control. IN a Target control, you can use the following properties to specify the usage of resources.

- **exBarPercent** property (get/set only) specifies the usage of the resource in the current bar/activity. The following VB sample updates the exBarEffort (so the histogram will be shown accordingly) once the user resizes a bar:

```
Private Sub G2antt1_BarResizing(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal
Key As Variant)
   With G2antt1.Items
      .ItemBar(Item, Key, EXG2ANTTLibCtl.ItemBarPropertyEnum.exBarEffort) =
.ItemBar(Item, Key, EXG2ANTTLibCtl.ItemBarPropertyEnum.exBarPercent)
   End With
End Sub
```

Now, let's display the Resources of the Source into the Target control (Target.PutRes Source.ResHandle, exPutResLoad)

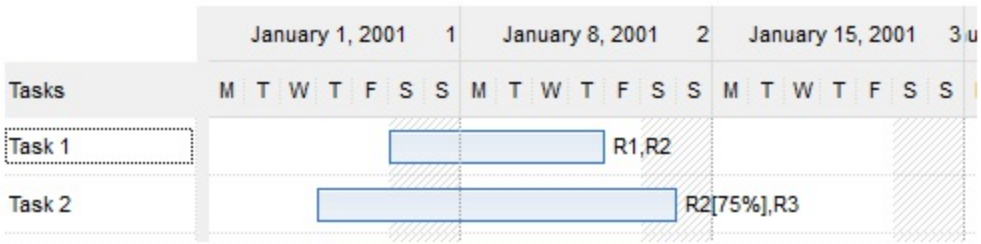
First step is specifying the ItemBar(exBarResources) property in the Source control. If the Source control has no bars with the ItemBar(exBarResources) property, no resource will be displayed in the target control.

The following sample adds a column and two activities/bars, with allocated resources:

```
With Source
  .BeginUpdate
  With .Chart
    .FirstVisibleDate = #1/1/2001#
    With .Bars("Task")
      .Def(exBarHAlignCaption) = 18
      .Def(exBarCaption) = "<%=%49%>"
    End With
  End With
  .Columns.Add "Tasks"
  Dim h As Long
  With .Items
```

```
    h = .AddItem("Task 1")
    .AddBar h, "Task", #1/6/2001#, #1/12/2001#, "K1"
    .ItemBar(h, "K1", exBarResources) = "R1,R2"
    h = .AddItem("Task 2")
    .AddBar h, "Task", #1/4/2001#, #1/14/2001#, "K2"
    .ItemBar(h, "K2", exBarResources) = "R2[75%],R3"
  End With
  .EndUpdate
End With
```

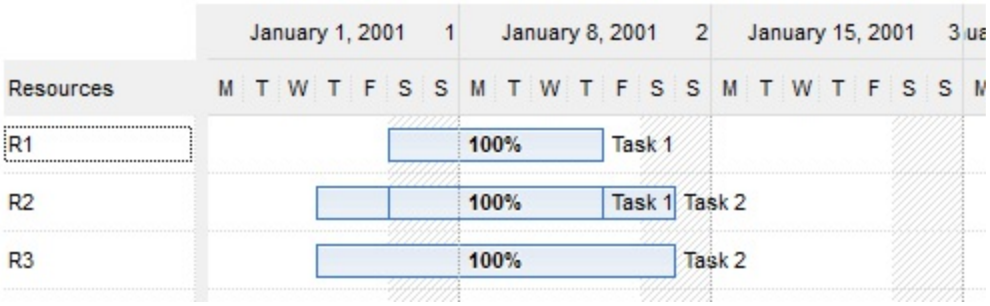
and the Source should show as:



The second step is calling the PutRes method as follows:

```
Target.PutRes Source.ResHandle, exPutResLoad
```

and the Target should show as:



The PutRes(exPutResLoad) method updates the Target as follows:

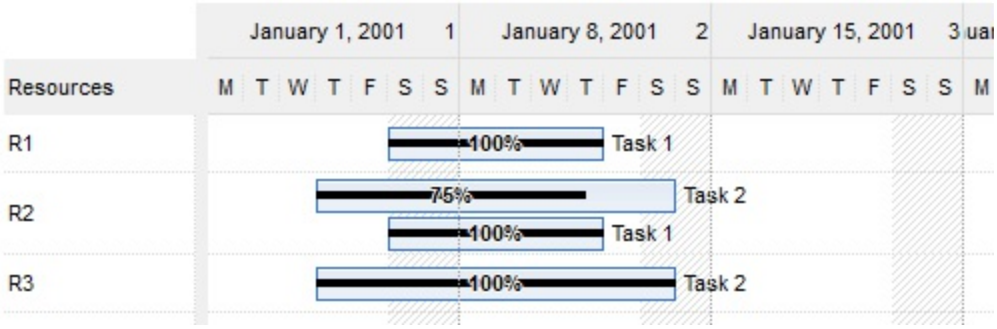
- adds the "Resources" column (nothing happens if the PutRes method was already called, or the Target control already contains a column with the Key "Resources")
- adds a new item with the name of the resource for each resource found (ItemBar(exBarResources)) in the Source control (R1, R2, ...)
- adds a new bar for each activity/bar in the Source control, that uses the specified resource, where the ItemBar(exBarPercent) and ItemBar(exBarEffort) properties indicate the usage of the resource (double expression between 0 and 1). The ItemBar(exBarEffort) property should be updated with the ItemBar(exBarPercent), during the BarResizing event, in case you provide a histogram view for the Target

control, as explained bellow.

As the Target control can display multiple activities/bars on the same row/item/resource we should make a few adjustments on the Target control as:

```
With Target
   With .Chart
      .FirstVisibleDate = Source.Chart.FirstVisibleDate
      With .Bars.Add("Task%Progress")
         .OverlaidType = exOverlaidBarsStack Or exOverlaidBarsStackAutoArrange
         .Shortcut = "Task"
      End With
   End With
End With
```

The code, defines the "Task" bar to display "Progress", and to be stacked on the same row. This code, should be called once, before calling the PutRes and so the Target should show as:



Now, let's display the histogram of Resources usage in the Target control The Target control represents a task into it's histogram only if:

- Bar.HistogramPattern or Bar.HistogramColor is specified. By default, none of these properties are set, so no bar is represented in the histogram

The control's histogram uses:

- ItemBar(exBarEffort) property specifies the effort to execute an unit in the task. By default, the ItemBar(exBarEffort) property is initialize with the ItemBar(exBarPercent) (resource usage percent)

The first step is to change the Target's code initialization as follows:

```
With Target
   With .Chart
```

```
    .FirstVisibleDate = Source.Chart.FirstVisibleDate
    With .Bars.Add("Task%Progress")
      .OverlaidType = exOverlaidBarsStack Or exOverlaidBarsStackAutoArrange
      .Shortcut = "Task"
      .HistogramPattern = exPatternShadow
      .HistogramCriticalColor = .HistogramColor
      .ShowHistogramValues = "1"
    End With
    .HistogramVisible = True
    .HistogramView = exHistogramCheckedItems
    .HistogramHeight = 164
  End With
  With .Columns.Add("Names")
    .Key = "Resources"
    .Def(exCellHasCheckBox) = True
  End With
End With
```

The code does the following:

- adds a column "Names", with the Key "Resources", that displays a check-box for each item, so next PutRes call won't add a new column
- change the bar's HistogramPattern so the "Task" will be displayed in the control's histogram
- display the control's histogram view

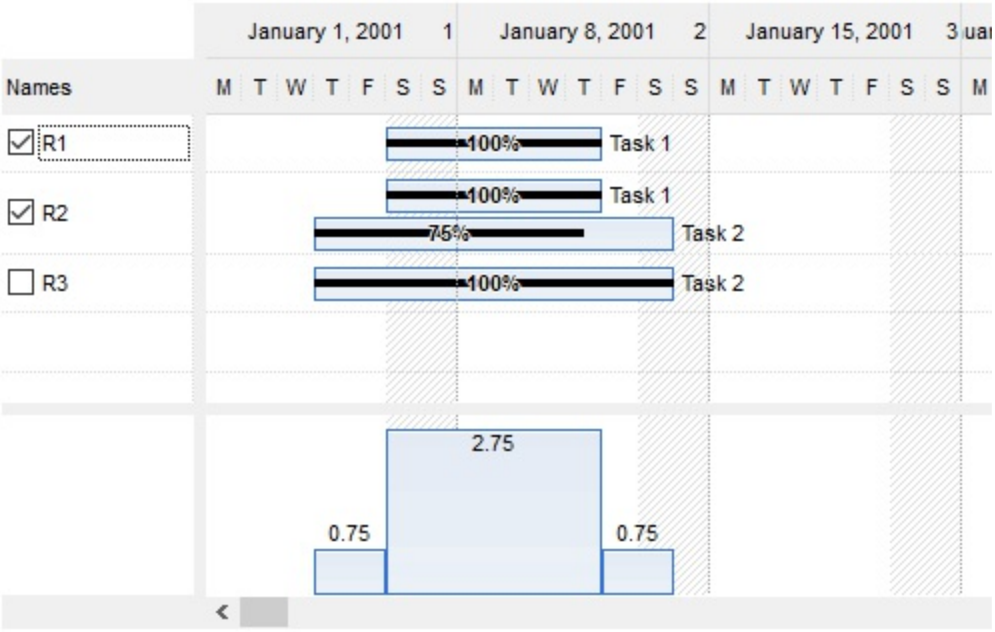
The second step is updating the exBarEffort with exBarPercent value, when the BarResizing event occurs:

```
Private Sub Target_BarResizing(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal Key As Variant)
  With Target.Items
    .ItemBar(Item, Key, exBarEffort) = .ItemBar(Item, Key, exBarPercent)
  End With
End Sub
```

The third step is calling the PutRes method as follows:

```
Target.PutRes Source.ResHandle, exPutResLoad
```

and the Target should show as:



Now, let's make the histogram displays cumulative-percents instead: So, the first step is to change the Target's code initialization as follows:

```
With Target
    With .Chart
        .FirstVisibleDate = Source.Chart.FirstVisibleDate
        With .Bars.Add("Task%Progress")
            .OverlaidType = exOverlaidBarsStack Or exOverlaidBarsStackAutoArrange
            .Shortcut = "Task"
            .HistogramType = exHistOverAllocation Or exHistCumulative
            .HistogramCumulativeColor(1) = RGB(255, 255, 0)
            .HistogramColor = RGB(196, 196, 196)
            .HistogramPattern = exPatternShadow
            .ShowHistogramValues = "value > 100 ? 255 : 1"
        End With
        .HistogramVisible = True
        .HistogramView = exHistogramCheckedItems
        .HistogramHeight = 96
    End With
    With .Columns.Add("Names")
        .Key = "Resources"
        .Def(exCellHasCheckBox) = True
    End With
End With
```

The code does the following:

- adds a column "Names", with the Key "Resources", that displays a check-box for each item, so next PutRes call won't add a new column
- change the bar's HistogramPattern so the "Task" will be displayed in the control's histogram
- display the control's histogram view

The second step is calling the PutRes method as follows:

```
Target.PutRes Source.ResHandle, exPutResLoad
```

The third step is updating the exBarEffort after PutRes call as follows:

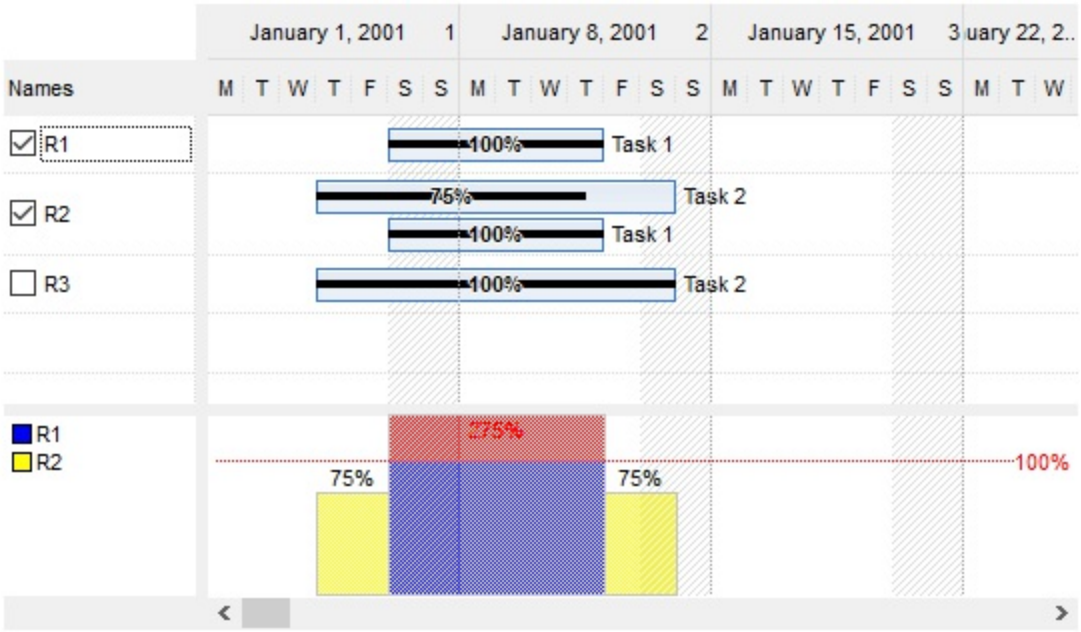
```
With Target.Items
   Dim Item As Variant
   For Each Item In Target.Items
      Dim Key As Variant
      Key = .FirstItemBar(Item)
      While Not IsEmpty(Key)
         .ItemBar(Item, Key, exBarEffort) = .ItemBar(Item, Key, exBarPercent) *
.ItemBar(Item, Key, exBarDuration)
         Key = .NextItemBar(Item, Key)
      Wend
   Next
End With
```

The code enumerates all the bars in the Target control, and changes the exBarEffort property to be exBarPecent of exBarDuration. This code is required as for exHistOverAllocation type the work-load for a task is computed as ItemBar(exBarEffort) / ItemBar(exBarDuration). The work-load for the task is the work effort / task duration. (i.e. If item.exBarEffort = 1 and the bar's length is 10 days, then the work-load = 0.1 or 10%). We also, should apply the change of exBarEffort when the BarResizing event, such as:

```
Private Sub Target_BarResizing(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal Key As
Variant)
   With Target.Items
      .ItemBar(Item, Key, exBarEffort) = .ItemBar(Item, Key, exBarPercent) *
.ItemBar(Item, Key, exBarDuration)
   End With
```

```
End Sub
```

and the Target should show as:



Now, let's update the Source control from the Target control (Source.PutRes Target.ResHandle, exPutResSave): The following code should be called to synchronize the Start/End/Resource-Usage from the Target to the Source

```
Source.PutRes Target.ResHandle, exPutResSave
```

The PutRes(exPutResSave) method updates the Source control as follows:

- updates the activity/bar's Start / ItemBar(exBarStart) and End / ItemBar(exBarEnd) date-time, in the Source control, according to its associated bar in the Target control
- updates the ItemBar(exBarResources) property in the Source control, with the new resource usage being indicated by ItemBar(exBarPercent) property in the Target control.

property G2antt.RadioImage(Checked as Boolean) as Long

Retrieves or sets a value that indicates the image used by cells of radio type.

Type	Description
Checked as Boolean	A boolean expression that indicates the radio's state. True means checked, and False means unchecked.
Long	A long expression that indicates the index of image used to paint the radio button. The last 7 bits in the high significant byte of the long expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part.

Use RadioImage and CheckImage properties to define the icons used for radio and check box cells. The RadioImage property defines the index of the icon being used by radio buttons. Use the CellHasRadioButton property to assign a radio button to a cell. Use the CellHasCheckBox property to assign a checkbox to a cell. Use the CellImage or CellImages property to assign one or multiple icons to a cell. Use the CellPicture property to assign a picture to a cell. Use the CellStateChanged event to notify your application when the cell's state is changed. Use the PartialCheck property to allow partial check feature within the column. Use the Images method to insert icons at runtime. The following samples require a control with icons, else nothing will be changed.

The following VB sample changes the default icon for the cells of radio type:

```
G2antt1.RadioImage(True) = 1        ' Sets the icon for cells of radio type that are checked
G2antt1.RadioImage(False) = 2       ' Sets the icon for cells of radio type that are unchecked
```

The G2antt1.RadioImage(True) = 0 makes the control to use the default icon for painting cells of radio type that are checked.

The following C++ sample changes the default icon for the cells of radio type:

```
m_g2antt.SetRadioImage( TRUE, 1 );
m_g2antt.SetRadioImage( FALSE, 2 );
```

The following VB.NET sample changes the default icon for the cells of radio type:

```
With AxG2antt1
   .set_RadioImage(True, 1)
   .set_RadioImage(False, 2)
End With
```

The following C# sample changes the default icon for the cells of radio type:

```
axG2antt1.set_RadioImage(true, 1);
axG2antt1.set_RadioImage(false, 2);
```

The following VFP sample changes the default icon for the cells of radio type:

```
with thisform.G2antt1
   local sT, sCR
   sCR = chr(13) + chr(10)
   sT = "RadioImage(True) = 1"+ sCR
   sT = sT + "RadioImage(False) = 2"+ sCR
   .Template = sT
endwith
```

The VFP considers the RadioImage call as being a call for an array, so an error occurs if the method is called directly, so we built a template string that we pass to the Template property.

property G2antt.RClickSelect as Boolean

Retrieves or sets a value that indicates whether an item is selected using right mouse button.

Type	Description
Boolean	A boolean expression that indicates whether an item is selected using the right mouse button.

Use the RClickSelect property to allow users select items using the right click. By default, the RClickSelect property is False. The control fires the SelectionChanged event when user selects an item. Use the SelectItem property to select programmatically select an item. Use the SelectCount property to get the number of selected items. Use the SelectedItem property to get the selected item. Use the FocusItem property to get the focused item. Use the ItemFromPoint property to retrieve an item from the point.

property G2antt.ReadOnly as ReadOnlyEnum

Retrieves or sets a value that indicates whether the control is read only.

Type	Description
ReadOnlyEnum	A ReadOnlyEnum expression that indicates whether the control is read only.

The ReadOnly property makes the control read only. Use the Enabled property to disable the control. Use the Locked property to lock an editor. If the control is read only, the Edit or Change event is not fired. Use the CellEditorVisible property to hide the cell's editor. Use the SelectableItem property to specify the user can select an item.

The ReadOnly property of the control mostly refers ot the List/Columns section of the control, and it is not applied to the Chart section.

You can use the following properties to prevent changes in the Chart section.

- AllowCreateBar property on False, prevents the user to create new bars in the chart panel.
- BarsAllowSizing property on False, prevents the user to move or resize any bar on the chart.
- AllowLinkBars property on False, prevents the user to link bars, at runtime

method G2antt.Refresh ()

Refreshes the control's content.

Type	Description

The Refresh method forces repainting the control. Use the BeginUpdate and EndUpdate methods to maintain the control's performance while adding multiple items or columns. Use the hWnd property to get the handle of the control's window.

The following VB sample calls the Refresh method:

```
G2antt1.Refresh
```

The following C++ sample calls the Refresh method:

```
m_g2antt.Refresh();
```

The following VB.NET sample calls the Refresh method:

```
AxG2antt1.CtlRefresh()
```

In VB.NET the System.Windows.Forms.Control class has already a Refresh method, so the CtlRefresh method should be called.

The following C# sample calls the Refresh method:

```
axG2antt1.CtlRefresh();
```

In C# the System.Windows.Forms.Control class has already a Refresh method, so the CtlRefresh method should be called.

The following VFP sample calls the Refresh method:

```
thisform.G2antt1.Object.Refresh()
```

method G2antt.RemoveSelection ()

Removes the selected links/bars from the chart if exists, else it removes the selected items (including the descendents)

Type	Description

The RemoveSelection method removes the selected links/bars from the chart if exists, else it removes the selected items (including the descendents). The UnselectAll method unselects all items. The RemoveSelection method removes the selected items (including the descendents). The RemoveSelection method removes the selected objects (bars or links) within the chart.

method G2antt.ReplaceIcon ([Icon as Variant], [Index as Variant])

Adds a new icon, replaces an icon or clears the control's image list.

Type	Description
Icon as Variant	A long expression that indicates the icon's handle.
Index as Variant	A long expression that indicates the index where icon is inserted.
Return	**Description**
Long	A long expression that indicates the index of the icon in the images collection

Use the ReplaceIcon property to add, remove or replace an icon in the control's images collection. Also, the ReplaceIcon property can clear the images collection. Use the Images method to attach a image list to the control. The ImageSize property defines the size (width/height) of the icons within the control's Images collection.

The following VB sample adds a new icon to control's images list:

```
i = ExG2antt1.ReplaceIcon( LoadPicture("d:\icons\help.ico").Handle), i specifies the index where the icon is added
```

The following VB sample replaces an icon into control's images list::

```
i = ExG2antt1.ReplaceIcon( LoadPicture("d:\icons\help.ico").Handle, 0), i is zero, so the first icon is replaced.
```

The following VB sample removes an icon from control's images list:

```
ExG2antt1.ReplaceIcon 0,  i, i specifies the index of icon removed.
```

The following VB clears the control's icons collection:

```
ExG2antt1.ReplaceIcon 0,  -1
```

property G2antt.ResHandle as Long

The ResHandle property indicates the handle to be used for ResHandle parameter of the PutRes method.

Type	Description
Long	A Long expression that indicates the unique resource handle, when using the PutRes method.

The ResHandle property indicates the handle to be used for ResHandle parameter of the PutRes method. The PutRes method associates an eXG2antt (Source) control with another eXG2antt (Target) control, using the Items.ItemBar(exBarResources). The Source displays the bar and the resources to be used by each bar, while the Target shows the Resources being used by each bar found in the Source.

property G2antt.RightToLeft as Boolean

Indicates whether the component should draw right-to-left for RTL languages.

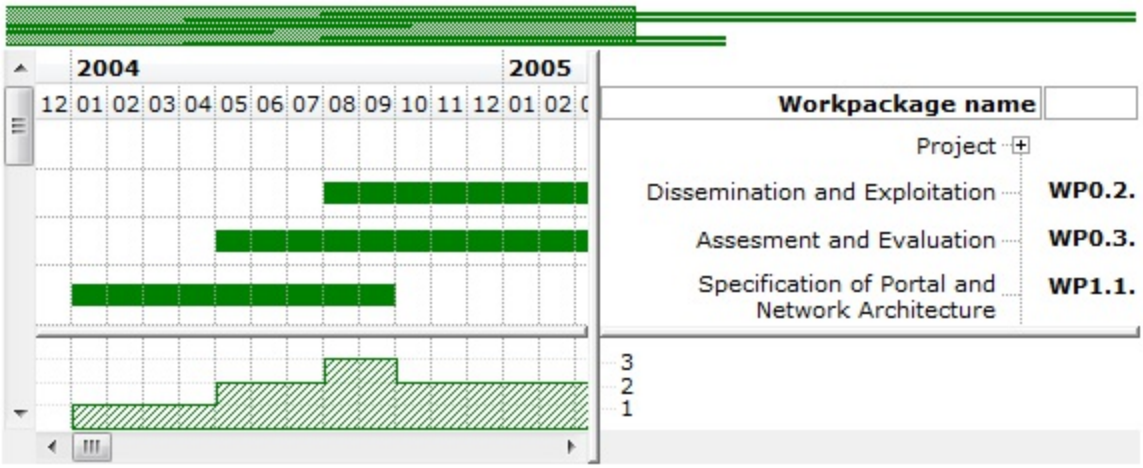
Type	Description
Boolean	A boolean expression that specifies whether the control is drawn from right to left or from left to right.

By default, the RightToLeft property is False. The RightToLeft gets or sets a value indicating whether control's elements are aligned to right or left. The RightTolLeft property affects all columns, and future columns being added.

Changing the RightToLeft property on True does the following:

- flips the panels, so the chart panel is displayed on the left side (ChartOnLeft property)
- displays the vertical scroll bar on the left side of the control (Scrollbars property)
- flips the order of the columns (Position property)
- change the column's alignment to right, if the column is not centered (Alignment property, HeaderAlignment property, HeaderImageAlignment property)
- reverse the order of the drawing parts in the cells (Def(exCellDrawPartsOrder) property to "caption,picture,icons,icon,check")
- aligns the locked columns to the right (CountLockedColumns property)
- aligns the control's group-by bar / sort bar to the right (SortBarVisible property)
- the control's filter bar/prompt/close is aligned to the right (FilterBarPromptVisible property)

The following screen shot shows how the control looks if the RightToLeft property is True:

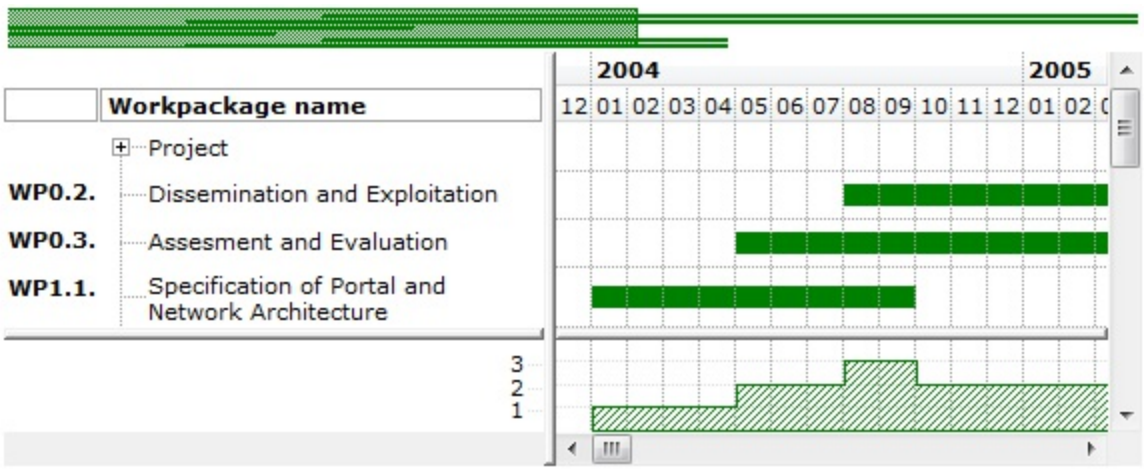


(By default) Changing the RightToLeft property on False does the following:

- flips the panels, so the chart panel is displayed on the right side (ChartOnLeft property)

- displays the vertical scroll bar on the right side of the control (Scrollbars property)
- flips the order of the columns (Position property)
- change the column's alignment to left, if the column is not centered (Alignment property, HeaderAlignment property, HeaderImageAlignment property)
- reverse the order of the drawing parts in the cells (Def(exCellDrawPartsOrder) property to "check,icon,icons,picture,caption")
- aligns the locked columns to the left (CountLockedColumns property)
- aligns the control's group-by bar / sort bar to the left (SortBarVisible property)
- the control's filter bar/prompt/close is aligned to the left (FilterBarPromptVisible property)

The following screen shot shows how the control looks if the RightToLeft property is False:



The following VB sample flips the order of the control's elements from right to left

```
With G2antt1
   .BeginUpdate
   .ScrollBars = exDisableBoth
   .LinesAtRoot = exLinesAtRoot
   With .Columns.Add("P1")
      .Def(exCellHasCheckBox) = True
      .PartialCheck = True
   End With
   With .Items
      h = .AddItem("Root")
      .InsertItem h,0,"Child 1"
      .InsertItem h,0,"Child 2"
      .ExpandItem(h) = True
   End With
   .RightToLeft = True
```

```
   .EndUpdate
End With
```

The following VB.NET sample flips the order of the control's elements from right to left

```
Dim h
With AxG2antt1
   .BeginUpdate
   .ScrollBars = EXG2ANTTLib.ScrollBarsEnum.exDisableBoth
   .LinesAtRoot = EXG2ANTTLib.LinesAtRootEnum.exLinesAtRoot
   With .Columns.Add("P1")
      .Def(EXG2ANTTLib.DefColumnEnum.exCellHasCheckBox) = True
      .PartialCheck = True
   End With
   With .Items
      h = .AddItem("Root")
      .InsertItem h,0,"Child 1"
      .InsertItem h,0,"Child 2"
      .ExpandItem(h) = True
   End With
   .RightToLeft = True
   .EndUpdate
End With
```

The following C++ sample flips the order of the control's elements from right to left

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
spG2antt1->PutScrollBars(EXG2ANTTLib::exDisableBoth);
spG2antt1->PutLinesAtRoot(EXG2ANTTLib::exLinesAtRoot);
```

```
EXG2ANTTLib::IColumnPtr var_Column = ((EXG2ANTTLib::IColumnPtr)(spG2antt1->GetColumns()->Add(L"P1")));
   var_Column->PutDef(EXG2ANTTLib::exCellHasCheckBox,VARIANT_TRUE);
   var_Column->PutPartialCheck(VARIANT_TRUE);
EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems();
   long h = var_Items->AddItem("Root");
   var_Items->InsertItem(h,long(0),"Child 1");
   var_Items->InsertItem(h,long(0),"Child 2");
   var_Items->PutExpandItem(h,VARIANT_TRUE);
spG2antt1->PutRightToLeft(VARIANT_TRUE);
spG2antt1->EndUpdate();
```

The following C# sample flips the order of the control's elements from right to left

```
axG2antt1.BeginUpdate();
axG2antt1.ScrollBars = EXG2ANTTLib.ScrollBarsEnum.exDisableBoth;
axG2antt1.LinesAtRoot = EXG2ANTTLib.LinesAtRootEnum.exLinesAtRoot;
EXG2ANTTLib.Column var_Column = (axG2antt1.Columns.Add("P1") as EXG2ANTTLib.Column);
   var_Column.set_Def(EXG2ANTTLib.DefColumnEnum.exCellHasCheckBox,true);
   var_Column.PartialCheck = true;
EXG2ANTTLib.Items var_Items = axG2antt1.Items;
   int h = var_Items.AddItem("Root");
   var_Items.InsertItem(h,0,"Child 1");
   var_Items.InsertItem(h,0,"Child 2");
   var_Items.set_ExpandItem(h,true);
axG2antt1.RightToLeft = true;
axG2antt1.EndUpdate();
```

The following VFP sample flips the order of the control's elements from right to left

```
with thisform.G2antt1
   .BeginUpdate
   .ScrollBars = 15
   .LinesAtRoot = -1
   with .Columns.Add("P1")
      .Def(0) = .T.
      .PartialCheck = .T.
```

```
   endwith
   with .Items
      h = .AddItem("Root")
      .InsertItem(h,0,"Child 1")
      .InsertItem(h,0,"Child 2")
      .DefaultItem = h
      .ExpandItem(0) = .T.
   endwith
   .RightToLeft = .T.
   .EndUpdate
endwith
```

The following Delphi sample flips the order of the control's elements from right to left

```
with AxG2antt1 do
begin
   BeginUpdate();
   ScrollBars := EXG2ANTTLib.ScrollBarsEnum.exDisableBoth;
   LinesAtRoot := EXG2ANTTLib.LinesAtRootEnum.exLinesAtRoot;
   with (Columns.Add('P1') as EXG2ANTTLib.Column) do
   begin
      Def[EXG2ANTTLib.DefColumnEnum.exCellHasCheckBox] := TObject(True);
      PartialCheck := True;
   end;
   with Items do
   begin
      h := AddItem('Root');
      InsertItem(h,TObject(0),'Child 1');
      InsertItem(h,TObject(0),'Child 2');
      ExpandItem[h] := True;
   end;
   RightToLeft := True;
   EndUpdate();
end
```

method G2antt.SaveXML (Destination as Variant)

Saves the control's content as XML document to the specified location, using the MSXML parser.

Type	Description
Destination as Variant	This object can represent a file name, reference to a string member, an XML document object, or a custom object that supports persistence as follows: • String - Specifies the file name. Note that this must be a file name, rather than a URL. The file is created if necessary and the contents are entirely replaced with the contents of the saved document. For example: `G2antt1.SaveXML("sample.xml")` • Reference to a String member - Saves the control's content to the string member. Note that the string member must be empty, before calling the SaveXML method. For example: `Dim s As String` `G2antt1.SaveXML s` In VB.NET for /NET assembly, you should call such as : `Dim s As String = String.Empty` `Exg2antt1.SaveXML(s)` In C# for /NET assembly, you should call such as : `string s = string.Empty;` `exg2antt1.SaveXML(ref s);` • XML Document Object. For example: `Dim xmldoc as Object` `Set xmldoc = CreateObject("MSXML.DOMDocument")` `G2antt1.SaveXML(xmldoc)` • Custom object supporting persistence - Any other custom COM object that supports **QueryInterface** for **IStream**, **IPersistStream**, or **IPersistStreamInit** can also be provided here and the document will be saved accordingly. In the **IStream** case, the **IStream::Write**

method will be called as it saves the document; in the **IPersistStream** case, **IPersistStream::Load** will be called with an **IStream** that supports the **Read**, **Seek**, and **Stat** methods.

Return	Description
Boolean	A Boolen expression that specifies whether saving the XML document was ok.

The SaveXML method uses the MSXML (MSXML.DOMDocument, XML DOM Document) parser to save the control's data in XML documents. The LoadXML method loads XML documents being created with SaveXML method. The SaveXML method saves each column in **<column>** elements under the **<columns>** collection. Properties like Caption, HTMLCaption, Image, Visible, LevelKey, DisplayFilterButton, DisplayFilterPattern, FilterType, Width and Position are saved for each column in the control. The **<items>** xml element saves a collection of **<item>** objects. Each <item> object holds information about an item in the control, including its cells, child items or bars. Each item saves a collection of **<cell>** objects that defines the cell for each column. The **<bars>** element saves a collection of **<bar>** each one is associated with the bars in the item. The Expanded attribute specifies whether an item is expanded or collapsed, and it carries the value of the ExpandItem property. The **<chart>** element saves data related to the chart data of the control. For instance, it includes the collection of levels being displayed in the chart, the first visible date, links and groups of bars. The **<levels>** element holds a collection of **<level>** objects each one being associated with an level in the chart area. The **<links>** element holds a collection of **<link>** objects each one indicating a link between two bars in the chart. The **<groups>** element holds a collection of **<group>** objects that indicates the bars that are grouped in the chart.

The control saves the control's data in XML format like follows:

```
- <Content Author Component Version ...>
  - <Chart FirstVisibleDate ...>
    - <Levels>
        <Level Label Unit Count />
        <Level Label Unit Count />
        ...
      </Levels>
    - <Links>
        <Link Key StartItem StartBar EndItem EndBar Visible StartPos EndPos Color Style
Width ShowDir Text ... />
```

```
        <Link Key StartItem StartBar EndItem EndBar Visible StartPos EndPos Color Style
Width ShowDir Text ... />
        ...
      </Links>
    - <Groups>
        <Group ItemA KeyA StartA ItemB KeyB StartB />
        <Group ItemA KeyA StartA ItemB KeyB StartB />
        ...
      </Groups>
   </Chart>
  - <Columns>
      <Column Caption Position Width HTMLCaption LevelKey DisplayFilterButton
DisplayFilterPatter FilterType ... />
      <Column Caption Position Width HTMLCaption LevelKey DisplayFilterButton
DisplayFilterPatter FilterType ... />
      ...
   </Columns>
  - <Items>
    - <Item Expanded ...>
        <Cell Value ValueFormat Images Image ... />
        <Cell Value ValueFormat Images Image ... />
        ...
      - <Bars>
          <Bar Name Start End Caption HAlignCaption VAlignCaption Key ... />
          <Bar Name Start End Caption HAlignCaption VAlignCaption Key ... />
          ...
        </Bars>
      - <Items>
        - <Item Expanded ...>
        - <Item Expanded ...>
        ....
        </Items>
      </Item>
   </Items>
 </Content>
```

The following C# sample saves the control's data to testing.xml file:

```
object xml = "c:\\testing.xml";
axG2antt1.SaveXML(ref xml);
```

The following VB.NET sample saves the control's data to testing.xml file:

```
Dim xml As String = "testing.xml"
AxG2antt1.SaveXML(xml)
```

method G2antt.Scroll (Type as ScrollEnum, [ScrollTo as Variant])

Scrolls the control's content.

Type	Description
Type as ScrollEnum	A ScrollEnum expression that indicates type of scrolling being performed.
ScrollTo as Variant	A long expression that indicates the position where the control is scrolled when Type is exScrollVTo or exScrollHTo. If the ScrollTo parameter is missing, 0 value is used.

Use the Scroll method to scroll the control's content by code. Use the EnsureVisibleItem method to ensure that a specified item fits the control's client area. Use the ScrollPos property to get the control's scroll position. Use the EnsureVisibleColumn method to ensure that a specified column fits the control's client area. If the Type parameter is exScrollLeft, exScrollRight or exScrollHTo the Scroll method scrolls horizontally the control's content pixel by pixel, if the ContinueColumnScroll property is False, else the Scroll method scrolls horizontally the control's content column by column. Use the ScrollTo method to ensure that a specified date fits the chart's client area. The FirstVisibleDate property specifies the first visible date.

If the Scroll(exScrollVTo) does not work please check if the ScrollBars property includes the exVScrollOnThumbRelease, and use a code like follows:

```
With G2antt1
    .ScrollBars = .ScrollBars And Not exVScrollOnThumbRelease
    .Scroll exScrollVTo, 10000
    .ScrollBars = .ScrollBars Or exVScrollOnThumbRelease
End With
```

The code removes temporary the exVScrollOnThumbRelease flag from the ScrollBars property, performs the scrolling (jump to row 10000) , and restore back the exVScrollOnThumbRelease flag.

The following VB sample scrolls the control's content to the first item (scrolls to the top):

```
G2antt1.Scroll exScrollVTo, 0
```

The following C++ sample scrolls the control's content to the top:

```
m_g2antt.Scroll( 2 /*exScrollVTo*/, COleVariant( (long)0 ) );
```

The following C# sample scrolls the control's content to the top:

```
axG2antt1.Scroll(EXG2ANTTLib.ScrollEnum.exScrollVTo, 0);
```

The following VB.NET sample scrolls the control's content to the top:

```
AxG2antt1.Scroll(EXG2ANTTLib.ScrollEnum.exScrollVTo, 0)
```

The following VFP sample scrolls the control's content to the top:

```
with thisform.G2antt1
    .Scroll( 2, 0 ) && exScrollVTo
endwith
```

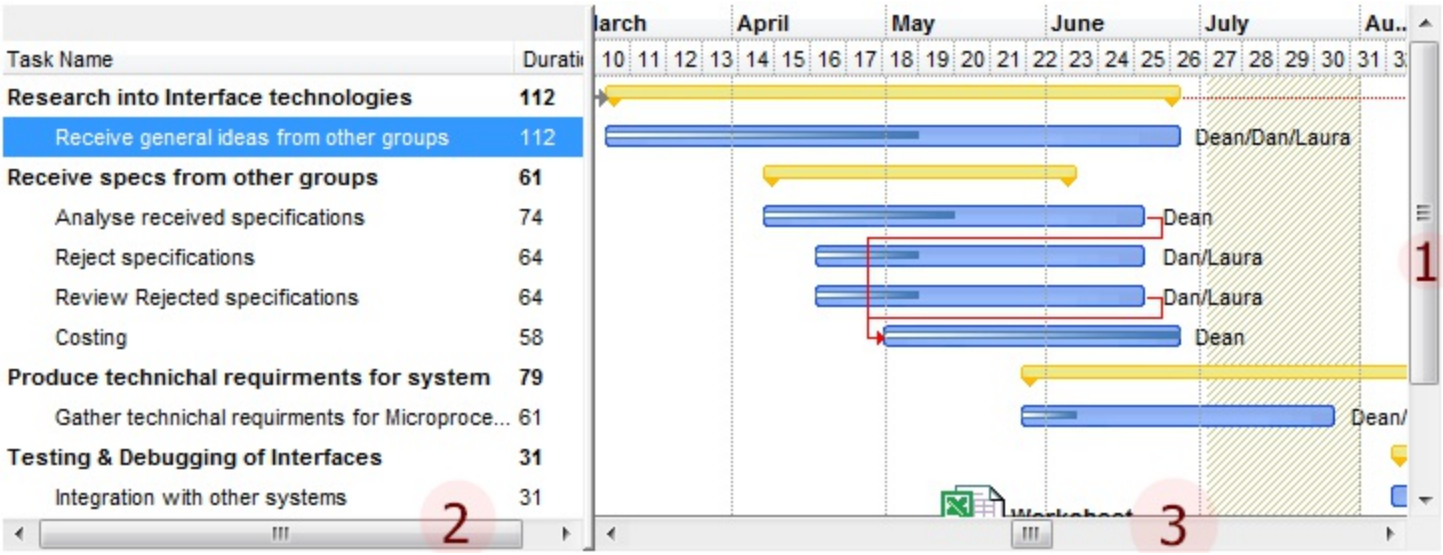
property G2antt.ScrollBars as ScrollBarsEnum

Returns or sets a value that determines whether the control has horizontal and/or vertical scroll bars.

Type	Description
ScrollBarsEnum	A ScrollBarsEnum expression that identifies which scroll bars are visible.

Use the ScrollBars property to show, enable or disable the control's scroll bars. By default, the ScrollBars property is exBoth, so both scroll bars are used if necessarily. For instance, if the ScrollBars property is exNone the control displays no scroll bars. Use the ScrollPos property to get the control's scroll position. Use the EnsureVisibleItem method to ensure that an item fits the control's client area. Use the EnsureVisibleColumn method to ensure that a specified column fits the control's client area. Use the Scroll method to scroll programmatically the control. Use the ScrollOrderParts property to customize the order of the buttons in the scroll bar. The RightToLeft property flips the order of the control's elements from right to left. Use the FirstVisibleDate property to scroll the chart's date.

The following screen shots shows all 3 scrollbars that can be visible in the control:



The control provides up to 3 scroll bars. The vertical scroll bar is displayed on the right side of the control (RightToLeft property is False) and it scrolls the items, rows (nodes). The other 2 scroll bars are for scrolling the columns section and the chart area. The ScrollBar property of the Chart object specifies whether the chart displays the horizontal scroll bar.

1. *vertical scroll bar*. Use the methods such us: Scroll, ScrollPos, EnsureVisibleItem to scroll vertically the control (scroll vertically the rows, items or nodes). The vertical scroll bar is displayed only if it required, if the ScrollBars property is exBoth or exVertical. The vertical scroll bar is always visible if the ScrollBars property is exDisableBoth or exDisableNoVertical.

2. *horizontal scroll bar*. Use the methods such us: Scroll, ScrollPos, EnsureVisibleColumn to scroll horizontally the control (scrolls horizontally the columns section). The horizontal scroll bar is displayed only if it required, if the ScrollBars property is exBoth or exHorizontal. The horizontal scroll bar is always visible if the ScrollBars property is exDisableBoth or exDisableNoHorizontal. The ColumnAutoResize property specifies whether all visible columns fit the control's area (so no horizontal scroll bar is shown).
3. *chart scroll bar*. Use the methods such us: Chart.ScrollTo, FirstVisibleDate to browse for a new date (scrolls horizontally the chart area, so a new range of dates are being displayed)

If the Scroll(exScrollVTo) does not work please check if the ScrollBars property includes the exVScrollOnThumbRelease, and use a code like follows:

```
With G2antt1
    .ScrollBars = .ScrollBars And Not exVScrollOnThumbRelease
    .Scroll exScrollVTo, 10000
    .ScrollBars = .ScrollBars Or exVScrollOnThumbRelease
End With
```

The code removes temporary the exVScrollOnThumbRelease flag from the ScrollBars property, performs the scrolling (jump to row 10000) , and restore back the exVScrollOnThumbRelease flag.

The following VB sample scrolls vertically the control to row 100:

```
G2antt1.Scroll exScrollVTo, 100
```

property G2antt.ScrollButtonHeight as Long

Specifies the height of the button in the vertical scrollbar.

Type	Description
Long	A long expression that defines the height of the button in the vertical scroll bar.

By default, the ScrollButtonHeight property is -1. If the ScrollButtonHeight property is -1, the control uses the default height (from the system) for the buttons in the vertical scroll bar. Use the ScrollButtonWidth property to specify the width of the buttons in the horizontal scroll bar. Use the ScrollWidth property to specify the width of the vertical scroll bar. Use the ScrollBars property to specify which scroll bar is visible or hidden in the control. Use the ScrollHeight property to specify the height of the horizontal scroll bar. Use the ScrollPartVisible property to specify the visible parts in the control's scroll bar. Use the ScrollThumbSize property to define a fixed size for the scrollbar's thumb.

property G2antt.ScrollButtonWidth as Long

Specifies the width of the button in the horizontal scrollbar.

Type	Description
Long	A long expression that defines the width of the button in the horizontal scroll bar.

By default, the ScrollButtonWidth property is -1. If the ScrollButtonWidth property is -1, the control uses the default width (from the system) for the buttons in the horizontal scroll bar. Use the ScrollButtonHeight property to specify the height of the buttons in the vertical scroll bar. Use the ScrollWidth property to specify the width of the vertical scroll bar. Use the ScrollBars property to specify which scroll bar is visible or hidden in the control. Use the ScrollHeight property to specify the height of the horizontal scroll bar. Use the ScrollPartVisible property to specify the visible parts in the control's scroll bar. Use the ScrollThumbSize property to define a fixed size for the scrollbar's thumb.

property G2antt.ScrollBySingleLine as Boolean

Retrieves or sets a value that indicates whether the control scrolls the lines to the end, item by item.

Type	Description
Boolean	A boolean expression that indicates whether the control scrolls the lines to the end, item by item.

By default, the ScrollBySingleLine property is False. We recommend to set the ScrollBySingleLine property on True if you have one of the following:

- If you have at least a cell that has CellSingleLine property on exCaptionWordWrap / exCaptionBreakWrap / False, or a column with Def(exCellSingleLine) on exCaptionWordWrap / exCaptionBreakWrap / False
- If your control contains at least an item that hosts an ActiveX control. See InsertControlItem property.
- If the control displays items with different height. Use the ItemHeight property to specify the item's height.
- If the chart displays bars with the OverlaidType property on exOverlaidBarsStack.

In conclusion, If the ScrollBySingleLine property is

- **False**, the first visible item can not be partially visible. The False value is recommended when all items has the same height.
- **True**, the first visible item can be partially visible, and clicking the up or down buttons on the vertical scroll bar makes the control to scroll vertically pixel by pixel (The DefaultItemHeight property indicates the number of pixels to scroll at once). You can set the AutoDrag property on exAutoDragScroll, and so the user can scroll the control's content by clicking the control and dragging the cursor up or down. The True value is recommended when the control may display items of different sizes.

Click here to watch a movie on how Scroll Line by Line works.

Use the EnsureVisibleItem property to ensure that an item fits the control's client area. Use the ScrollBars property to hide the control's scroll bars. Use the Scroll method to programmatically scroll the control's content. Use the ItemsAllowSizing property to specify whether all items are resizable or not. Use the ItemAllowSizing property to specify whether the user can resize the item at runtime.

property G2antt.ScrollFont (ScrollBar as ScrollBarEnum) as IFontDisp

Retrieves or sets the scrollbar's font.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBarEnum expression that indicates the vertical or the horizontal scroll bar.
IFontDisp	A Font object

Use the ScrollFont property to specify the font in the control's scroll bar. Use the ScrolPartCaption property to specify the caption of the scroll's part. Use the ScrollPartVisible property to add or remove buttons/parts in the control's scrollbar. By default, when a part becomes visible, the ScrollPartEnable property is automatically called, so the parts becomes enabled. Use the ScrollPartEnable property to specify enable or disable parts in the control's scrollbar. Use the ScrollBars property to specify the visible scrollbars in the control. Use the OffsetChanged event to notify your application that the scroll position is changed. Use the OversizeChanged event to notify your application whether the range for a specified scroll bar is changed. Use the ScrollPos property to specify the position for the control's scroll bar. The control fires the ScrollButtonClick event when the user clicks a part of the scroll bar.

property G2antt.ScrollHeight as Long

Specifies the height of the horizontal scrollbar.

Type	Description
Long	A long expression that defines the height of the horizontal scroll bar.

By default, the ScrollHeight property is -1. If the ScrollHeight property is -1, the control uses the default height of the horizontal scroll bar from the system. Use the ScrollHeight property to specify the height of the horizontal scroll bar. Use the ScrollBars property to specify which scroll bar is visible or hidden in the control. Use the ScrollButtonWidth property to specify the width of the buttons in the horizontal scroll bar. Use the ScrollWidth property to specify the width of the vertical scroll bar. Use the ScrollButtonHeight property to specify the height of the buttons in the vertical scroll bar. Use the ScrollPartVisible property to specify the visible parts in the control's scroll bar. Use the ScrollThumbSize property to define a fixed size for the scrollbar's thumb.

property G2antt.ScrollOrderParts(ScrollBar as ScrollBarEnum) as String

Specifies the order of the buttons in the scroll bar.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBar expression that indicates the scrollbar where the order of buttons is displayed.
String	A String expression that indicates the order of the buttons in the scroll bar. The list includes expressions like l, l1, ..., l5, t, r, r1, ..., r6 separated by comma, each expression indicating a part of the scroll bar, and its position indicating the displaying order.

Use the ScrollOrderParts to customize the order of the buttons in the scroll bar. By default, the ScrollOrderParts property is empty. If the ScrollOrderParts property is empty the default order of the buttons in the scroll bar are displayed like follows:

so, the order of the parts is: l1, l2, l3, l4, l5, l, t, r, r1, r2, r3, r4, r5 and r6. Use the ScrollPartVisible to specify whether a button in the scrollbar is visible or hidden. Use the ScrollPartEnable property to enable or disable a button in the scroll bar. Use the ScrollPartCaption property to assign a caption to a button in the scroll bar.

Use the ScrollOrderParts property to change the order of the buttons in the scroll bar. For instance, "l,r,t,l1,r1" puts the left and right buttons to the left of the thumb area, and the l1 and r1 buttons right after the thumb area. If the parts are not specified in the ScrollOrderParts property, automatically they are added to the end.

The list of supported literals in the ScrollOrderParts property is:

- **l** for exLeftBPart, (<) The left or top button.
- **l1** for exLeftB1Part, (L1) The first additional button, in the left or top area.
- **l2** for exLeftB2Part, (L2) The second additional button, in the left or top area.
- **l3** for exLeftB3Part, (L3) The third additional button, in the left or top area.
- **l4** for exLeftB4Part, (L4) The forth additional button, in the left or top area.
- **l5** for exLeftB5Part, (L5) The fifth additional button, in the left or top area.
- **t** for exLowerBackPart, exThumbPart and exUpperBackPart, The union between the exLowerBackPart and the exUpperBackPart parts.
- **r** for exRightBPart, (>) The right or down button.
- **r1** for exRightB1Part, (R1) The first additional button in the right or down side.

- **r2** for exRightB2Part, (R2) The second additional button in the right or down side.
- **r3** for exRightB3Part, (R3) The third additional button in the right or down side.
- **r4** for exRightB4Part, (R4) The forth additional button in the right or down side.
- **r5** for exRightB5Part, (R5) The fifth additional button in the right or down side.
- **r6** for exRightB6Part, (R6) The sixth additional button in the right or down side.

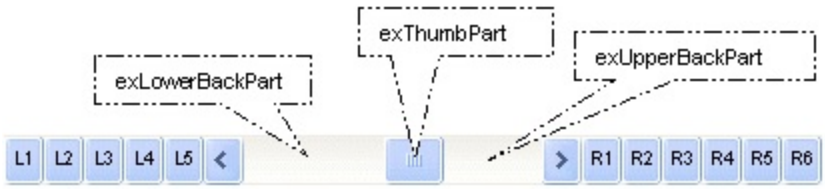
Any other literal between commas is ignored. If duplicate literals are found, the second is ignored, and so on. For instance, "t,l,r" indicates that the left/top and right/bottom buttons are displayed right/bottom after the thumb area.

property G2antt.ScrollPartCaption(ScrollBar as ScrollBarEnum, Part as ScrollPartEnum) as String

Specifies the caption being displayed on the specified scroll part.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBar expression that indicates the scrollbar where the caption is displayed.
Part as ScrollPartEnum	A ScrollPartEnum expression that specifies the parts of the scroll where the text is displayed
String	A String expression that specifies the caption being displayed on the part of the scroll bar.

Use the ScrolPartCaption property to specify the caption of the scroll's part. Use the ScrollPartVisible property to add or remove buttons/parts in the control's scrollbar. By default, when a part becomes visible, the ScrollPartEnable property is automatically called, so the parts becomes enabled. Use the ScrollPartEnable property to specify enable or disable parts in the control's scrollbar. Use the ScrollBars property to specify the visible scrollbars in the control. Use the OffsetChanged event to notify your application that the scroll position is changed. Use the OversizeChanged event to notify your application whether the range for a specified scroll bar is changed. Use the ScrollPos property to specify the position for the control's scroll bar. The control fires the ScrollButtonClick event when the user clicks a part of the scroll bar. Use the ScrollFont property to specify the font in the control's scroll bar. Use the ScrollOrderParts property to customize the order of the buttons in the scroll bar. The ScrollPartCaptionAlignment property specifies the alignment of the caption in the part of the scroll bar.



By default, the following parts are shown:

- exLeftBPart (the left or up button of the control)
- exLowerBackPart (the part between the left/up button and the thumb part of the control)
- exThumbPart (the thumb/scrollbox part)
- exUpperBackPart (the part between the the thumb and the right/down button of the control)
- exRightBPart (the right or down button of the control)

The following VB sample adds up and down additional buttons to the control's vertical scroll

bar :

```
With G2antt1
   .BeginUpdate
      .ScrollBars = exDisableBoth
      .ScrollPartVisible(exVScroll, exLeftB1Part Or exRightB1Part) = True
      .ScrollPartCaption(exVScroll, exLeftB1Part) = "<img></img>1"
      .ScrollPartCaption(exVScroll, exRightB1Part) = "<img></img>2"
   .EndUpdate
End With
```

The following VB.NET sample adds up and down additional buttons to the control's vertical scroll bar :

```
With AxG2antt1
   .BeginUpdate()
   .ScrollBars = EXG2ANTTLib.ScrollBarsEnum.exDisableBoth
   .set_ScrollPartVisible(EXG2ANTTLib.ScrollBarEnum.exVScroll,
EXG2ANTTLib.ScrollPartEnum.exLeftB1Part Or
EXG2ANTTLib.ScrollPartEnum.exRightB1Part, True)
   .set_ScrollPartCaption(EXG2ANTTLib.ScrollBarEnum.exVScroll,
EXG2ANTTLib.ScrollPartEnum.exLeftB1Part, "<img></img>1")
   .set_ScrollPartCaption(EXG2ANTTLib.ScrollBarEnum.exVScroll,
EXG2ANTTLib.ScrollPartEnum.exRightB1Part, "<img></img>2")
   .EndUpdate()
End With
```

The following C# sample adds up and down additional buttons to the control's vertical scroll bar :

```
axG2antt1.BeginUpdate();
axG2antt1.ScrollBars = EXG2ANTTLib.ScrollBarsEnum.exDisableBoth;
axG2antt1.set_ScrollPartVisible(EXG2ANTTLib.ScrollBarEnum.exVScroll,
EXG2ANTTLib.ScrollPartEnum.exLeftB1Part | EXG2ANTTLib.ScrollPartEnum.exRightB1Part,
true);
axG2antt1.set_ScrollPartCaption(EXG2ANTTLib.ScrollBarEnum.exVScroll,
EXG2ANTTLib.ScrollPartEnum.exLeftB1Part , "<img></img>1");
axG2antt1.set_ScrollPartCaption(EXG2ANTTLib.ScrollBarEnum.exVScroll,
EXG2ANTTLib.ScrollPartEnum.exRightB1Part, "<img></img>2");
```

```
axG2antt1.EndUpdate();
```

The following C++ sample adds up and down additional buttons to the control's vertical scroll bar :

```
m_g2antt.BeginUpdate();
m_g2antt.SetScrollBars( 15 /*exDisableBoth*/ );
m_g2antt.SetScrollPartVisible( 0 /*exVScroll*/, 32768  /*exLeftB1Part*/ | 32
/*exRightB1Part*/, TRUE );
m_g2antt.SetScrollPartCaption( 0 /*exVScroll*/, 32768  /*exLeftB1Part*/ , _T("<img>
</img>1") );
m_g2antt.SetScrollPartCaption( 0 /*exVScroll*/, 32 /*exRightB1Part*/ , _T("<img>
</img>2") );
m_g2antt.EndUpdate();
```

The following VFP sample adds up and down additional buttons to the control's vertical scroll bar :

```
With thisform.G2antt1
   .BeginUpdate
      .ScrollBars = 15
      .ScrollPartVisible(0, bitor(32768,32)) = .t.
      .ScrollPartCaption(0,32768) = "<img></img>1"
      .ScrollPartCaption(0, 32) = "<img></img>2"
   .EndUpdate
EndWith
```

```
*** ActiveX Control Event ***
LPARAMETERS scrollpart

wait window nowait ltrim(str(scrollpart))
```

property G2antt.ScrollPartCaptionAlignment(ScrollBar as ScrollBarEnum, Part as ScrollPartEnum) as AlignmentEnum

Specifies the alignment of the caption in the part of the scroll bar.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBar expression that indicates the scrollbar where the caption is displayed.
Part as ScrollPartEnum	A ScrollPartEnum expression that specifies the parts of the scroll where the text is displayed
AlignmentEnum	An AlignmentEnum expression that specifies the alignment of the caption in the part of the scrollbar.

The ScrollPartCaptionAlignment property specifies the alignment of the caption in the part of the scroll bar. By default, the caption is centered. Use the ScrolPartCaption property to specify the caption being displayed on specified part of the scroll bar. Use the ScrollPartVisible property to add or remove buttons/parts in the control's scrollbar.

The following VB sample displays "left" aligned to the left on the lower part of the control's horizontal scroll bar, and "right" aligned to the right on the upper part of the control's horizontal scroll bar:

```
With G2antt1
   .ScrollPartCaption(exHScroll,exLowerBackPart) = "left"
   .ScrollPartCaptionAlignment(exHScroll,exLowerBackPart) = LeftAlignment
   .ScrollPartCaption(exHScroll,exUpperBackPart) = "right"
   .ScrollPartCaptionAlignment(exHScroll,exUpperBackPart) = RightAlignment
   .ColumnAutoResize = False
   .Columns.Add 1
   .Columns.Add 2
   .Columns.Add 3
   .Columns.Add 4
End With
```

The following VB.NET sample displays "left" aligned to the left on the lower part of the control's horizontal scroll bar, and "right" aligned to the right on the upper part of the control's horizontal scroll bar:

```
With AxG2antt1
```

```
.set_ScrollPartCaption(EXG2ANTTLib.ScrollBarEnum.exHScroll,EXG2ANTTLib.ScrollPartEnum



.set_ScrollPartCaptionAlignment(EXG2ANTTLib.ScrollBarEnum.exHScroll,EXG2ANTTLib.Scro



.set_ScrollPartCaption(EXG2ANTTLib.ScrollBarEnum.exHScroll,EXG2ANTTLib.ScrollPartEnum



.set_ScrollPartCaptionAlignment(EXG2ANTTLib.ScrollBarEnum.exHScroll,EXG2ANTTLib.Scro

    .ColumnAutoResize = False
    .Columns.Add 1
    .Columns.Add 2
    .Columns.Add 3
    .Columns.Add 4
End With
```

The following C# sample displays "left" aligned to the left on the lower part of the control's horizontal scroll bar, and "right" aligned to the right on the upper part of the control's horizontal scroll bar:

```
axG2antt1.set_ScrollPartCaption(EXG2ANTTLib.ScrollBarEnum.exHScroll,EXG2ANTTLib.Scro

axG2antt1.set_ScrollPartCaptionAlignment(EXG2ANTTLib.ScrollBarEnum.exHScroll,EXG2AN

axG2antt1.set_ScrollPartCaption(EXG2ANTTLib.ScrollBarEnum.exHScroll,EXG2ANTTLib.Scro

axG2antt1.set_ScrollPartCaptionAlignment(EXG2ANTTLib.ScrollBarEnum.exHScroll,EXG2AN

axG2antt1.ColumnAutoResize = false;
axG2antt1.Columns.Add(1.ToString());
axG2antt1.Columns.Add(2.ToString());
axG2antt1.Columns.Add(3.ToString());
axG2antt1.Columns.Add(4.ToString());
```

The following C++ sample displays "left" aligned to the left on the lower part of the control's horizontal scroll bar, and "right" aligned to the right on the upper part of the control's

horizontal scroll bar:

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import "ExG2antt.dll"
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1-
>PutScrollPartCaption(EXG2ANTTLib::exHScroll,EXG2ANTTLib::exLowerBackPart,L"left");
spG2antt1-
>PutScrollPartCaptionAlignment(EXG2ANTTLib::exHScroll,EXG2ANTTLib::exLowerBackPart,

spG2antt1-
>PutScrollPartCaption(EXG2ANTTLib::exHScroll,EXG2ANTTLib::exUpperBackPart,L"right");
spG2antt1-
>PutScrollPartCaptionAlignment(EXG2ANTTLib::exHScroll,EXG2ANTTLib::exUpperBackPart,

spG2antt1->PutColumnAutoResize(VARIANT_FALSE);
spG2antt1->GetColumns()->Add(L"1");
spG2antt1->GetColumns()->Add(L"2");
spG2antt1->GetColumns()->Add(L"3");
spG2antt1->GetColumns()->Add(L"4");
```

The following VFP sample displays "left" aligned to the left on the lower part of the control's horizontal scroll bar, and "right" aligned to the right on the upper part of the control's horizontal scroll bar:

```
with thisform.G2antt1
   .ScrollPartCaption(1,512) = "left"
   .ScrollPartCaptionAlignment(1,512) = 0
   .ScrollPartCaption(1,128) = "right"
   .ScrollPartCaptionAlignment(1,128) = 2
   .ColumnAutoResize = .F.
   .Columns.Add(1)
```

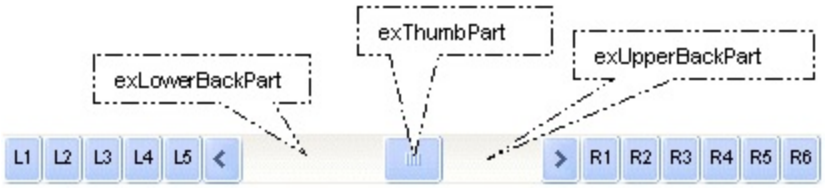
```
    .Columns.Add(2)
    .Columns.Add(3)
    .Columns.Add(4)
endwith
```

property G2antt.ScrollPartEnable(ScrollBar as ScrollBarEnum, Part as ScrollPartEnum) as Boolean

Indicates whether the specified scroll part is enabled or disabled.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBar expression that indicates the scrollbar where the part is enabled or disabled.
Part as ScrollPartEnum	A ScrollPartEnum expression that specifies the parts of the scroll bar being enabled or disabled.
Boolean	A Boolean expression that specifies whether the scrollbar's part is enabled or disabled.

By default, when a part becomes visible, the ScrollPartEnable property is automatically called, so the parts becomes enabled. Use the ScrollPartVisible property to add or remove buttons/parts in the control's scrollbar. Use the ScrollPartEnable property to specify enable or disable parts in the control's scrollbar. Use the ScrollBars property to specify the visible scrollbars in the control. Use the ScrolPartCaption property to specify the caption of the scroll's part. Use the OffsetChanged event to notify your application that the scroll position is changed. Use the OversizeChanged event to notify your application whether the range for a specified scroll bar is changed. Use the ScrollPos property to specify the position for the control's scroll bar. The control fires the ScrollButtonClick event when the user clicks a part of the scroll bar. Use the ScrollOrderParts property to customize the order of the buttons in the scroll bar.

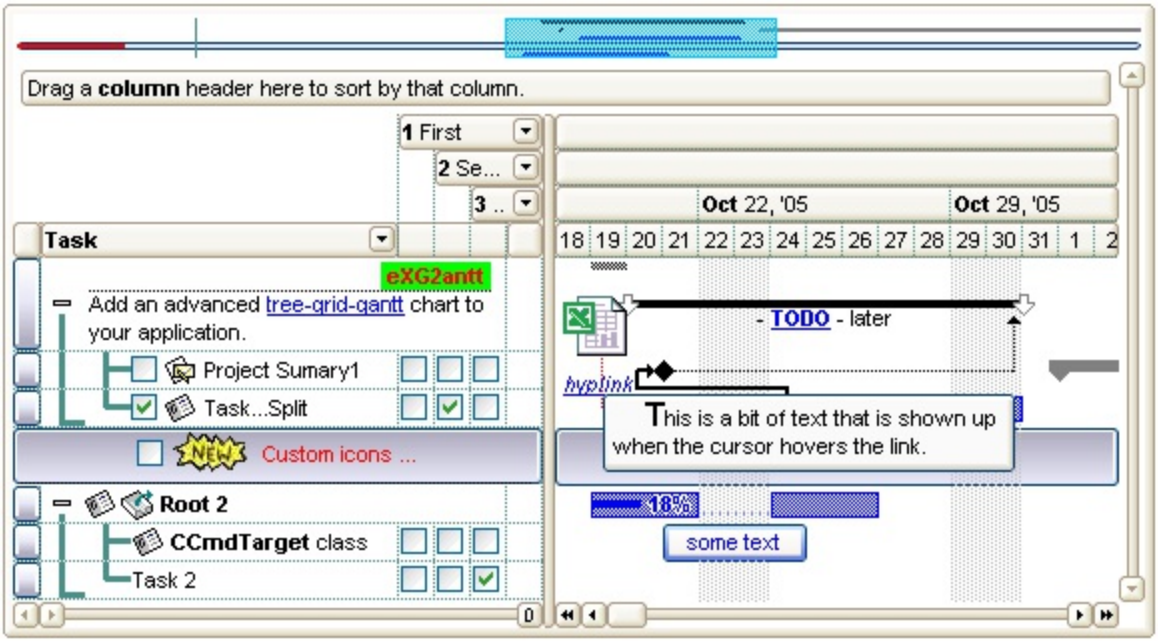


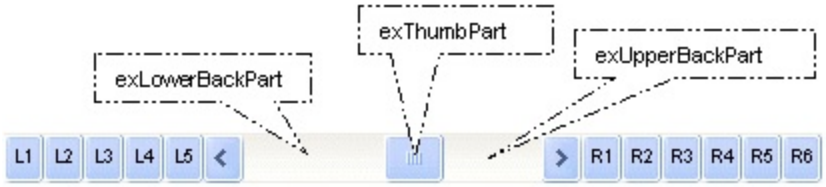
property G2antt.ScrollPartVisible(ScrollBar as ScrollBarEnum, Part as ScrollPartEnum) as Boolean

Indicates whether the specified scroll part is visible or hidden.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBar expression that indicates the scrollbar where the part is visible or hidden.
Part as ScrollPartEnum	A ScrollPartEnum expression that specifies the parts of the scroll bar being visible
Boolean	A Boolean expression that specifies whether the scrollbar's part is visible or hidden.

Use the ScrollPartVisible property to add or remove buttons/parts in the control's scrollbar. By default, when a part becomes visible, the ScrollPartEnable property is automatically called, so the parts becomes enabled. Use the ScrollPartEnable property to specify enable or disable parts in the control's scrollbar. Use the ScrollBars property to specify the visible scrollbars in the control. Use the ScrolPartCaption property to specify the caption of the scroll's part. Use the OffsetChanged event to notify your application that the scroll position is changed. Use the OversizeChanged event to notify your application whether the range for a specified scroll bar is changed. Use the ScrollPos property to specify the position for the control's scroll bar. The control fires the ScrollButtonClick event when the user clicks a part of the scroll bar. Use the Background property to change the visual appearance for any part in the control's scroll bar. Use the ScrollOrderParts property to customize the order of the buttons in the scroll bar.





By default, the following parts are shown:

- exLeftBPart (the left or up button of the control)
- exLowerBackPart (the part between the left/up button and the thumb part of the control)
- exThumbPart (the thumb/scrollbox part)
- exUpperBackPart (the part between the the thumb and the right/down button of the control)
- exRightBPart (the right or down button of the control)

The following VB sample adds up and down additional buttons to the control's vertical scroll bar :

```
With G2antt1
   .BeginUpdate
      .ScrollBars = exDisableBoth
      .ScrollPartVisible(exVScroll, exLeftB1Part Or exRightB1Part) = True
      .ScrollPartCaption(exVScroll, exLeftB1Part) = "<img></img>1"
      .ScrollPartCaption(exVScroll, exRightB1Part) = "<img></img>2"
   .EndUpdate
End With
```

The following VB.NET sample adds up and down additional buttons to the control's vertical scroll bar :

```
With AxG2antt1
   .BeginUpdate()
   .ScrollBars = EXG2ANTTLib.ScrollBarsEnum.exDisableBoth
   .set_ScrollPartVisible(EXG2ANTTLib.ScrollBarEnum.exVScroll,
EXG2ANTTLib.ScrollPartEnum.exLeftB1Part Or
EXG2ANTTLib.ScrollPartEnum.exRightB1Part, True)
   .set_ScrollPartCaption(EXG2ANTTLib.ScrollBarEnum.exVScroll,
EXG2ANTTLib.ScrollPartEnum.exLeftB1Part, "<img></img>1")
   .set_ScrollPartCaption(EXG2ANTTLib.ScrollBarEnum.exVScroll,
EXG2ANTTLib.ScrollPartEnum.exRightB1Part, "<img></img>2")
```

```
    .EndUpdate()
End With
```

The following C# sample adds up and down additional buttons to the control's vertical scroll bar :

```
axG2antt1.BeginUpdate();
axG2antt1.ScrollBars = EXG2ANTTLib.ScrollBarsEnum.exDisableBoth;
axG2antt1.set_ScrollPartVisible(EXG2ANTTLib.ScrollBarEnum.exVScroll,
EXG2ANTTLib.ScrollPartEnum.exLeftB1Part | EXG2ANTTLib.ScrollPartEnum.exRightB1Part,
true);
axG2antt1.set_ScrollPartCaption(EXG2ANTTLib.ScrollBarEnum.exVScroll,
EXG2ANTTLib.ScrollPartEnum.exLeftB1Part , "<img></img>1");
axG2antt1.set_ScrollPartCaption(EXG2ANTTLib.ScrollBarEnum.exVScroll,
EXG2ANTTLib.ScrollPartEnum.exRightB1Part, "<img></img>2");
axG2antt1.EndUpdate();
```

The following C++ sample adds up and down additional buttons to the control's vertical scroll bar :

```
m_g2antt.BeginUpdate();
m_g2antt.SetScrollBars( 15 /*exDisableBoth*/ );
m_g2antt.SetScrollPartVisible( 0 /*exVScroll*/, 32768  /*exLeftB1Part*/ | 32
/*exRightB1Part*/, TRUE );
m_g2antt.SetScrollPartCaption( 0 /*exVScroll*/, 32768  /*exLeftB1Part*/ , _T("<img>
</img>1") );
m_g2antt.SetScrollPartCaption( 0 /*exVScroll*/, 32 /*exRightB1Part*/ , _T("<img>
</img>2") );
m_g2antt.EndUpdate();
```

The following VFP sample adds up and down additional buttons to the control's vertical scroll bar :

```
With thisform.G2antt1
    .BeginUpdate
        .ScrollBars = 15
        .ScrollPartVisible(0, bitor(32768,32)) = .t.
        .ScrollPartCaption(0,32768) = "<img></img>1"
        .ScrollPartCaption(0, 32) = "<img></img>2"
```

```
    .EndUpdate
EndWith
```

```
*** ActiveX Control Event ***
LPARAMETERS scrollpart

wait window nowait ltrim(str(scrollpart))
```

property G2antt.ScrollPos(Vertical as Boolean) as Long

Specifies the vertical/horizontal scroll position.

Type	Description
Vertical as Boolean	A boolean expression that specifies the scrollbar being requested. True indicates the Vertical scroll bar, False indicates the Horizontal scroll bar.
Long	A long expression that defines the scroll bar position.

Use the ScrollPos property to change programmatically the position of the control's scroll bar. Use the ScrollPos property to get the horizontal or vertical scroll position.Use the ScrollBars property to define the control's scroll bars. Use the Scroll method to scroll programmatically the control's content. The control fires the OffsetChanged event when the control's scroll position is changed. Use the ScrollTo method to ensure that a specified date fits the chart's client area. The FirstVisibleDate property specifies the first visible date.

The following VB sample scrolls to the row 10,000:

```
With G2antt1
   .ScrollPos(True) = 10000
End With
```

property G2antt.ScrollThumbSize(ScrollBar as ScrollBarEnum) as Long

Specifies the size of the thumb in the scrollbar.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBarEnum expression that indicates the vertical or the horizontal scroll bar.
Long	A long expression that defines the size of the scrollbar's thumb.

Use the ScrollThumbSize property to define a fixed size for the scrollbar's thumb. By default, the ScrollThumbSize property is -1, that makes the control computes automatically the size of the thumb based on the scrollbar's range. If case, use the fixed size for your thumb when you change its visual appearance using the Background(exVSThumb) or Background(exHSThumb) property. Use the ScrollWidth property to specify the width of the vertical scroll bar. Use the ScrollButtonWidth property to specify the width of the buttons in the horizontal scroll bar. Use the ScrollHeight property to specify the height of the horizontal scroll bar. Use the ScrollButtonHeight property to specify the height of the buttons in the vertical scroll bar. Use the ScrollPartVisible property to specify the visible parts in the control's scroll bar.

property G2antt.ScrollToolTip(ScrollBar as ScrollBarEnum) as String

Specifies the tooltip being shown when the user moves the scroll box.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBarEnum expression that indicates the vertical scroll bar or the horizontal scroll bar.
String	A string expression being shown when the user clicks and moves the scrollbar's thumb.

Use the ScrollToolTip property to specify whether the control displays a tooltip when the user clicks and moves the scrollbar's thumb. By default, the ScrollToolTip property is empty. If the ScrollToolTip property is empty, the tooltip is not shown when the user clicks and moves the thumb of the scroll bar. The OffsetChanged event notifies your application that the user changes the scroll position. Use the SortPartVisible property to specify the parts being visible in the control's scroll bar. Use the ScrollBars property to specify the visible scrollbars in the control.

The following VB sample displays a tooltip when the user clicks and moves the thumb in the control's scroll bar:

```
Private Sub G2antt1_OffsetChanged(ByVal Horizontal As Boolean, ByVal NewVal As Long)
  If (Not Horizontal) Then
    G2antt1.ScrollToolTip(exVScroll) = "Record " & NewVal
  End If
End Sub
```

The following VB.NET sample displays a tooltip when the user clicks and moves the thumb in the control's scroll bar:

```
Private Sub AxG2antt1_OffsetChanged(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_OffsetChangedEvent) Handles AxG2antt1.OffsetChanged
  If (Not e.horizontal) Then
    AxG2antt1.set_ScrollToolTip(EXG2ANTTLib.ScrollBarEnum.exVScroll, "Record " &
e.newVal.ToString())
  End If
End Sub
```

The following C++ sample displays a tooltip when the user clicks and moves the thumb in the control's scroll bar:

```
void OnOffsetChangedG2antt1(BOOL Horizontal, long NewVal)
{
    if ( !Horizontal )
    {
        CString strFormat;
        strFormat.Format( _T("%i"), NewVal );
        m_g2antt.SetScrollToolTip( 0, strFormat );
    }
}
```

The following C# sample displays a tooltip when the user clicks and moves the thumb in the control's scroll bar:

```
private void axG2antt1_OffsetChanged(object sender,
AxEXG2ANTTLib._IG2anttEvents_OffsetChangedEvent e)
{
    if ( !e.horizontal )
        axG2antt1.set_ScrollToolTip(EXG2ANTTLib.ScrollBarEnum.exVScroll, "Record " +
e.newVal.ToString());
}
```

The following VFP sample displays a tooltip when the user clicks and moves the thumb in the control's scroll bar:

```
*** ActiveX Control Event ***
LPARAMETERS horizontal, newval

If (1 # horizontal) Then
    thisform.G2antt1.ScrollToolTip(0) = "Record " + ltrim(str(newval))
EndIf
```

property G2antt.ScrollWidth as Long

Specifies the width of the vertical scrollbar.

Type	Description
Long	A long expression that defines the width of the vertical scroll bar.

By default, the ScrollWidth property is -1. If the ScrollWidth property is -1, the control uses the default width of the vertical scroll bar from the system. Use the ScrollWidth property to specify the width of the vertical scroll bar. Use the ScrollBars property to specify which scroll bar is visible or hidden in the control. Use the ScrollButtonWidth property to specify the width of the buttons in the horizontal scroll bar. Use the ScrollHeight property to specify the height of the horizontal scroll bar. Use the ScrollButtonHeight property to specify the height of the buttons in the vertical scroll bar. Use the ScrollPartVisible property to specify the visible parts in the control's scroll bar. Use the ScrollThumbSize property to define a fixed size for the scrollbar's thumb.

property G2antt.SearchColumnIndex as Long

Retrieves or sets a value indicating the column's index that is used for auto search feature.

Type	Description
Long	A long expression indicating the column's index that is used for auto search feature.

The SearchColumnIndex property indicates the index of the column being used by the control's incremental search feature. The user changes the searching column if he presses TAB or Shift + TAB. Use the UseTabKey property to specify whether the control uses the TAB key. Use the AutoSearch property to specify whether the control enables the incremental searching feature. Use the AutoSearch property to specify the type of incremental searching the control supports within the column. Use the MarkSearchColumn property to hide the rectangle around the searching column.

property G2antt.SelBackColor as Color

Retrieves or sets a value that indicates the selection background color.

Type	Description
Color	A color expression that indicates the selection background color. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

By default, the SelBackColor property applies the background color only to list area. Use the Chart.SelBackColor property to specify the background color for selected items in the chart area. Use the SelBackColor and SelForeColor properties to define the colors used for selected items. The control highlights the selected items only if the SelBackColor and BackColor properties have different values, and the SelForeColor and ForeColor properties have different values. Use the SelectCount property to get the number of selected items. Use the SelectedItem property to get the selected item. Use the SelectItem to select or unselect a specified item. Use the FocusItem property to get the focused item. The control fires the SelectionChanged event when user changes the selection. Use the SelectableItem property to specify the user can select an item. The SelBarColor property specifies the color to highlight the selected bars. The SelBackMode property specifies the way the selected items are shown in the control.

For instance, the following VB sample changes the visual appearance for the selected item. The SelBackColor property indicates the selection background color. Shortly, we need to add a skin to the Appearance object using the Add method, and we need to set the last 7 bits in the SelBackColor property to indicates the index of the skin that we want to use. The sample applies the "" to the selected item(s):

```
With G2antt1
   With .VisualAppearance
      .Add &H23, App.Path + "\selected.ebn"
   End With
   .SelForeColor = RGB(0, 0, 0)
   .SelBackColor = &H23000000
End With
```

The sample adds the skin with the index 35 (Hexa 23), and applies to the selected item using the SelBackColor property.

The following C++ sample applies a new appearance to the selected item(s):

```
#include "Appearance.h"
m_g2antt.GetVisualAppearance().Add( 0x23,
COleVariant(_T("D:\\Temp\\ExG2antt_Help\\selected.ebn")) );
m_g2antt.SetSelBackColor( 0x23000000 );
m_g2antt.SetSelForeColor( 0 );
```

The following VB.NET sample applies a new appearance to the selected item(s):

```
With AxG2antt1
   With .VisualAppearance
      .Add(&H23, "D:\Temp\ExG2antt_Help\selected.ebn")
   End With
   .SelForeColor = Color.Black
   .Template = "SelBackColor = 587202560"
End With
```

The VB.NET sample uses the Template property to assign a new value to the SelBackColor property. The 587202560 value represents &23000000 in hexadecimal.

The following C# sample applies a new appearance to the selected item(s):

```
axG2antt1.VisualAppearance.Add(0x23, "D:\\Temp\\ExG2antt_Help\\selected.ebn");
axG2antt1.Template = "SelBackColor = 587202560";
```

The following VFP sample applies a new appearance to the selected item(s):

```
With thisform.G2antt1
   With .VisualAppearance
      .Add(35, "D:\Temp\ExG2antt_Help\selected.ebn")
   EndWith
   .SelForeColor = RGB(0, 0, 0)
   .SelBackColor = .587202560
EndWith
```

The 587202560 value represents &23000000 in hexadecimal. The 32 value represents &23 in hexadecimal

property G2antt.SelBackMode as BackModeEnum

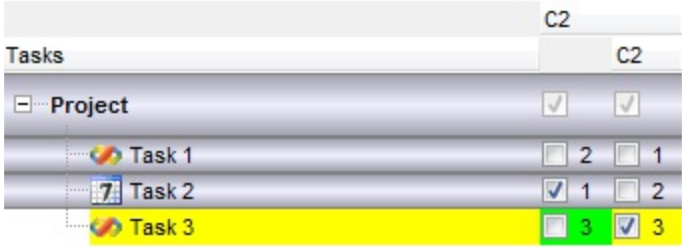
Retrieves or sets a value that indicates whether the selection is transparent or opaque.

Type	Description
BackModeEnum	A BackModeEnum expression that indicates whether the selection is transparent or opaque.

By default, the SelBackMode property is exOpaque. Use the SelBackMode property to specify how the selection is shown in the control. Use the SelBackMode property to specify a specify a semi-transparent color so the selected rows do not lose the colors, pictures, when they are selected. Use the SelBackColor property to specify the visual appearance or the background color for selected items. Use the SelForeColor property to specify the selection foreground color. The SingleSel property specifies whether the control supports single or multiple selection. The control fires the SelectionChanged event when user selects an item. Use the SelectedItem property to get the selected item. Use the SelectItem to select or unselect a specified item. The FullRowSelect property specifies whether the full item or a single cell is being selected.

The following screen shot shows the control when no items are selected:

The following screen shot shows the first three items selected, while the SelBackMode property is **exOpaque**:



The following screen shot shows the first three items selected, while the SelBackMode property is **exOpaque**, and FullRowSelect property is 0:

The following screen shot shows the first three items selected, while the SelBackMode property is **exTransparent**:

The following screen shot shows the first three items selected, while the SelBackMode property is **exGrid**:

property G2antt.SelectByDrag as Boolean

Specifies whether the user selects multiple items by dragging.

Type	Description
Boolean	A boolean expression that specifies whether the user may select multiple items by drag and drop.

By default, SelectByDrag property is True. Use the SelectByDrag property to disable selecting multiple items by dragging. The SelectByDrag property has effect only if the control supports multiple selection. The SingleSel property controls the number of items that the user may select. For instance, if the SingleSel property is True, the user can't select multiple items, and so a single item may be selected at the time. If the SingleSel property is False, the user can select multiple items using the mouse, keyboard or both. When the SelectByDrag property is True, the user may click the non text area to start select items by dragging. Use the SelectByDrag property on False when your control requires OLE drag and drop operations, like when you select multiple items and drag them to a new position. Use the OLEDropMode property to specify whether the OLE drag and drop operations inside the control is allowed. For instance, if the SelectByDrag and OLEDropMode properties are on, sometimes it is confused what control should do when user clicks and start to select items. The AllowSelectNothing property specifies whether the current selection is erased, once the user clicks outside of the items section. The SelectOnRelease property indicates whether the selection occurs when the user releases the mouse button.

property G2antt.SelectColumn as Boolean

Specifies whether the user selects cells only in SelectColumnIndex column, while FullRowSelect property is False.

Type	Description
Boolean	A boolean expression that specifies whether the user selects cells only in SelectColumnIndex column, while the FullRowSelect property is False

By default, the SelectColumn property is False. The SelectColumn property has effect only if the FullRowSelect is False. The control displays the selected cell in the SelectColumnIndex column. The SelectColumnIndex property specifies the index of selected column. Use the SelectableItem property to specify the user can select an item.

property G2antt.SelectColumnIndex as Long

Retrieves or sets a value that indicates the column's index where the user can select an item by clicking.

Type	Description
Long	A long expression that indicates the column's index where the user can select the item.

The property has effect only if the FullRowSelect property is False. Use the SelectedItem property to determine the selected items. Use the SelectColumnInner property to get the index of the inner cell that's selected or focused. Use the SplitCell property to split a cell. Use the SelectableItem property to specify the user can select an item.

property G2antt.SelectColumnInner as Long

Retrieves or sets a value that indicates the index of the inner cell that's selected.

Type	Description
Long	A long expression that indicates the index of the inner cell that's focused or selected.

Use the SelectColumnInner property to get the index of the inner cell that's selected or focused. The SelectColumnInner property may be greater than zero, if the control contains inner cells. The SplitCell method splits a cell in two cells. The newly created cell is called inner cell. The FocusItem property indicates the focused item. The SelectColumnIndex property determines the index of the column that's selected when FullRowSelect property is False. Use the SelectableItem property to specify the user can select an item.

property G2antt.SelectOnRelease as Boolean

Indicates whether the selection occurs when the user releases the mouse button.

Type	Description
Boolean	A Boolean expression that indicates whether the selection occurs when the user releases the mouse button.

By default, the SelectOnRelease property is False. By default, the selection occurs, as soon as the user clicks an object. The SelectOnRelease property indicates whether the selection occurs when the user releases the mouse button. The SelectOnRelease property has no effect if the SingleSel property is False, and SelectByDrag property is True.

property G2antt.SelForeColor as Color

Retrieves or sets a value that indicates the selection foreground color.

Type	Description
Color	A color expression that indicates the selection foreground color.

By default, the SelForeColor property is applied ONLY to selected items being displayed in the list area. Use the SelForeColor property to change the foreground color of selected items being displayed in the chart area. Use the SelForeColor and SelBackColor properties to change the colors used for selected items. The control highlights the selected items only if the SelBackColor and BackColor properties have different values, and the SelForeColor and ForeColor properties have different values. Use the SelectCount property to get the number of selected items. Use the SelectedItem property to get the selected item. Use the SelectItem to select or unselect a specified item. Use the FocusItem property to get the focused item. The control fires the SelectionChanged event when user changes the selection. Use the SelectableItem property to specify the user can select an item. The SelForeColor property is applied only if it is different that the control's foreground color. Use the SelectOnClick property to disable selecting new items when the user clicks the chart area. The SelBackMode property specifies the way the selected items are shown in the control.

property G2antt.ShowFocusRect as Boolean

Retrieves or sets a value indicating whether the control draws a thin rectangle around the focused item.

Type	Description
Boolean	A boolean expression that indicates whether the control draws a thin rectangle around the focused item.

Use the ShowFocusRect property to hide the rectangle drawn around the focused item. The FocusItem property specifies the handle of the focused item. If there is no focused item the FocusItem property retrieves 0. At one moment, only one item can be focused. When the selection is changed the focused item is changed too. Use the SelectCount property to get the number of selected items. Use the SelectedItem property to get the selected item. Use the SelectItem to select or unselect a specified item. If the control supports only single selection, you can use the FocusItem property to get the selected/focused item because they are always the same.

property G2antt.ShowImageList as Boolean

Specifies whether the control's image list window is visible or hidden.

Type	Description
Boolean	A boolean expression that specifies whether the control's image list window is visible or hidden.

By default, the ShowImageList property is True. Use the ShowImageList property to hide the control's images list window. The control's images list window is visible only at design time. Use the Images method to associate an images list control to the control. Use the RepaceIcon method to add, remove or clear icons in the control's images collection. Use the CellImage, CellImages properties to assign icons to a cell. Use the CellPicture property to assign a picture to a cell. Use the CheckImage or RadioImage property to specify a different look for checkboxes or radio buttons in the cells.

property G2antt.ShowLockedItems as Boolean

Retrieves or sets a value that indicates whether the locked items are visible or hidden.

Type	Description
Boolean	A boolean expression that specifies whether the locked items are shown or hidden.

A locked or fixed item is always displayed on the top or bottom side of the control no matter if the control's list is scrolled up or down. Use the ShowLockedItems property to show or hide the locked items. Use the LockedItemCount property to add or remove items fixed/locked to the top or bottom side of the control. Use the LockedItem property to access a locked item by its position. Use the CellValue property to specify the caption for a cell.

method G2antt.ShowToolTip (ToolTip as String, [Title as Variant], [Alignment as Variant], [X as Variant], [Y as Variant])

Shows the specified tooltip at given position.

Type	Description
ToolTip as String	The ToolTip parameter can be any of the following: • NULL(BSTR) or "<null>"(string) to indicate that the tooltip for the object being hovered is not changed • A String expression that indicates the description of the tooltip, that supports built-in HTML format (adds, replaces or changes the object's tooltip)
Title as Variant	The Title parameter can be any of the following: • missing (VT_EMPTY, VT_ERROR type) or "<null>" (string) the title for the object being hovered is not changed. • A String expression that indicates the title of the tooltip (no built-in HTML format) (adds, replaces or changes the object's title)
Alignment as Variant	A long expression that indicates the alignment of the tooltip relative to the position of the cursor. If missing (VT_EMPTY, VT_ERROR) the alignment of the tooltip for the object being hovered is not changed. The Alignment parameter can be one of the following: • 0 - exTopLeft • 1 - exTopRight • 2 - exBottomLeft • 3 - exBottomRight • 0x10 - exCenter • 0x11 - exCenterLeft • 0x12 - exCenterRight • 0x13 - exCenterTop • 0x14 - exCenterBottom By default, the tooltip is aligned relative to the top-left corner (0 - exTopLeft).

X as Variant	Specifies the horizontal position to display the tooltip as one of the following: • missing (VT_EMPTY, VT_ERROR type), indicates that the tooltip is shown on its default position / current cursor position (ignored) • -1, indicates the current horizontal position of the cursor (current x-position) • a numeric expression that indicates the horizontal screen position to show the tooltip (fixed screen x-position) • a string expression that indicates the horizontal displacement relative to default position to show the tooltip (moved)
Y as Variant	Specifies the vertical position to display the tooltip as one of the following: • missing (VT_EMPTY, VT_ERROR type), indicates that the tooltip is shown on its default position / current cursor position (ignored) • -1, indicates the current vertical position of the cursor (current y-position) • a numeric expression that indicates the vertical screen position to show the tooltip (fixed screen y-position) • a string expression that indicates the vertical displacement relative to default position to show the tooltip (displacement)

Use the ShowToolTip method to display a custom tooltip at specified position or to update the object's tooltip, title or position. You can call the ShowToolTip method during the MouseMove/ToolTip event. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the ToolTipWidth property to specify the width of the tooltip window. Use the ToolTipFont property to change the tooltip's font. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color. Use the CellToolTip property to specify the cell's tooltip. Use the ItemBar(,,exBarToolTip) property to specify a tooltip for a bar. Use the Link(,exLinkToolTip) property to specify the link's tooltip.

For instance:

- ShowToolTip(`<null>`,`<null>`,,`+8`,`+8`), shows the tooltip of the object moved relative to its default position
- ShowToolTip(`<null>`,`new title`), adds, changes or replaces the title of the object's tooltip
- ShowToolTip(`new content`), adds, changes or replaces the object's tooltip
- ShowToolTip(`new content`,`new title`), shows the tooltip and title at current position
- ShowToolTip(`new content`,`new title`,,`+8`,`+8`), shows the tooltip and title moved relative to the current position
- ShowToolTip(`new content`,``,,128,128), displays the tooltip at a fixed position
- ShowToolTip(``,``), hides the tooltip

The ToolTip parameter supports the built-in HTML format like follows:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.

  The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using <a ;exp=> or <a ;e64=> anchor tags. The exp/e64 field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption.

  - exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
  - e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

  Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show

lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a

known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold

- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with $^{\text{subscript}}$
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

shadow

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

outline anti-aliasing

The following VB sample displays (as tooltip) the identifier of the anchor element from the cursor:

```
Private Sub G2antt1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    With G2antt1
        .ShowToolTip .AnchorFromPoint(-1, -1)
    End With
End Sub
```

The following VB.NET sample displays (as tooltip) the identifier of the anchor element from the cursor:

```
Private Sub AxG2antt1_MouseMoveEvent(ByVal sender As System.Object, ByVal e As AxEXG2ANTTLib._IG2anttEvents_MouseMoveEvent) Handles AxG2antt1.MouseMoveEvent
    With AxG2antt1
        .ShowToolTip(.get_AnchorFromPoint(-1, -1))
    End With
End Sub
```

The following C# sample displays (as tooltip) the identifier of the anchor element from the cursor:

```
private void axG2antt1_MouseMoveEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_MouseMoveEvent e)
{
    axG2antt1.ShowToolTip(axG2antt1.get_AnchorFromPoint(-1, -1));
}
```

The following C++ sample displays (as tooltip) the identifier of the anchor element from the cursor:

```
void OnMouseMoveG2antt1(short Button, short Shift, long X, long Y)
{
    COleVariant vtEmpty; V_VT( &vtEmpty ) = VT_ERROR;
    m_g2antt.ShowToolTip( m_g2antt.GetAnchorFromPoint( -1, -1 ), vtEmpty, vtEmpty,
vtEmpty );
}
```

The following VFP sample displays (as tooltip) the identifier of the anchor element from

the cursor:

```
*** ActiveX Control Event ***
LPARAMETERS button, shift, x, y

with thisform
   With .G2antt1
      .ShowToolTip(.AnchorFromPoint(-1, -1))
   EndWith
endwith
```

property G2antt.SingleSel as Boolean

Retrieves or sets a value that indicates whether the control supports single or multiple selection.

Type	Description
Boolean	A boolean expression that indicates whether the control supports single or multiple selection.

Use the SingleSel property to enable multiple selection. Use the SelectCount property to get the number of selected items. Use the SelectedItem property to get the selected item. Use the SelectItem to select or unselect a specified item. Use the FocusItem property to get the focused item. If the control supports only single selection, you can use the FocusItem property to get the selected/focused item because they are always the same. The control fires the SelectionChanged event when user selects an item. Use the SelForeColor and SelBackColor properties to specify colors for selected items. Use the SelectableItem property to specify the user can select an item. The FullRowSelect property specifies whether the selection spans the entire width of the control. Use the SelectAll method to select all visible items. Use the SelectOnClick property to disable selecting new items when the user clicks the chart area. The AllowSelectNothing property specifies whether the current selection is erased, once the user clicks outside of the items section. The SelectOnRelease property indicates whether the selection occurs when the user releases the mouse button.

property G2antt.SingleSort as Boolean

Returns or sets a value that indicates whether the control supports sorting by single or multiple columns.

Type	Description
Boolean	A boolean expression that indicates whether the control supports sorting by single or multiple columns.

Use the SingleSort property to allow sorting by multiple columns. Sorting by a single column in the control is a simple matter of clicking on the column head. Sorting by multiple columns, however, is not so obvious. But it's actually quite easy. The user has two options to sort by multiple columns:

- First, sort by the first criterion, by clicking on the column head. Then hold the SHIFT key down as you click on a second heading.
- Click the column head and drag to the control's sort bar in the desired position.

By default, the SingleSort property is True, and so the user can have sorting by a single column only. Use the SortBarVisible property to show the control's sort bar. The SingleSort property is automatically set on False, if the SortBarVisible property is set to True. Use the SortOnClick property to specify the action that control should execute when the user clicks the control's header. Use the SortOrder property to sort a column programmatically. Use the SortPosition property to specify the position of the column in the sorted columns list. The control fires the Sort event when the user sorts a column. Use the ItemBySortPosition property to get the columns being sorted in their order.

For instance, if the control contains multiple sorted columns, changing the SingleSort property on True, erases all the columns in the sorting columns collection, and so no column is sorted.

property G2antt.SortBarCaption as String

Specifies the caption being displayed on the control's sort bar when the sort bar contains no columns.

Type	Description
String	A String expression that indicates the caption of the control's sort bar.

The SortBarCaption property specifies the caption of the control's sort bar, when it contains no sorted columns. Use the SortBarVisible property to show the control's sort bar. Use the BackColorSortBar, BackColorSortBarCaption and ForeColorSortBar properties to specify colors for the control's sort bar. Use the SortBarHeight property to specify the height of the control's sort bar. Use the SortBarColumnWidth property to specify the width of the column in the control's sort bar. By default, the SortBarCaption property is "Drag a column header here to sort by that column.". Use the Font property to specify the control's font. Use the ItemBySortPosition property to access the columns in the control's sort bar.

The SortBarCaption property may include built-in HTML tags like follows:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.

  The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using <a ;exp=> or <a ;e64=> anchor tags. The exp/e64 field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption.

  - exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
  - e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The

Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously

loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.

- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{subscript}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

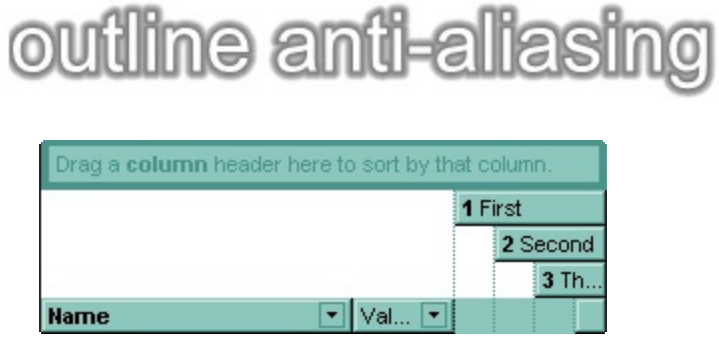
- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

shadow

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor></**sha**>*" gets:



property G2antt.SortBarColumnWidth as Long

Specifies the maximum width a column can be in the control's sort bar.

Type	Description
Long	A long expression that indicates the width of the columns in the control's sort bar. If the value is negative, all columns in the sort bar are displayed with the same width (the absolute value represents the width of the columns, in pixels). If the value is positive, it indicates the maximum width, so the width of the columns in the sort bar may differ.

Use the SortBarColumnWidth property to specify the width of the column in the control's sort bar. Use the SortBarVisible property to show the control's sort bar. Use the Width property to specify the width of the column in the control's header bar. Use the SortBarHeight property to specify the height of the control's sort bar. Use the SortBarCaption property to specify the caption being displayed in the control's sort bar when it contains no columns.

property G2antt.SortBarHeight as Long

Retrieves or sets a value that indicates the height of the control's sort bar.

Type	Description
Long	A long expression that indicates the height of the control's sort bar, in pixels.

Use the SortBarHeight property to specify the height of the control's sort bar. Use the SortBarVisible property to show the control's sort bar. By default, the SortBarHeight property is 18 pixels. Use the HeaderHeight property to specify the height of the control's header bar. Use the SortBarColumnWidth property to specify the width of the columns being displayed in the control's sort bar. Use the BackColorSortBar, BackColorSortBarCaption and ForeColorSortBar properties to specify colors for the control's sort bar. Use the SortBarCaption property to specify the caption being displayed in the control's sort bar when it contains no columns.

property G2antt.SortBarVisible as Boolean

Retrieves or sets a value that indicates whether control's sort bar is visible or hidden.

Type	Description
Boolean	A boolean expression that indicates whether the sort bar is visible or hidden.

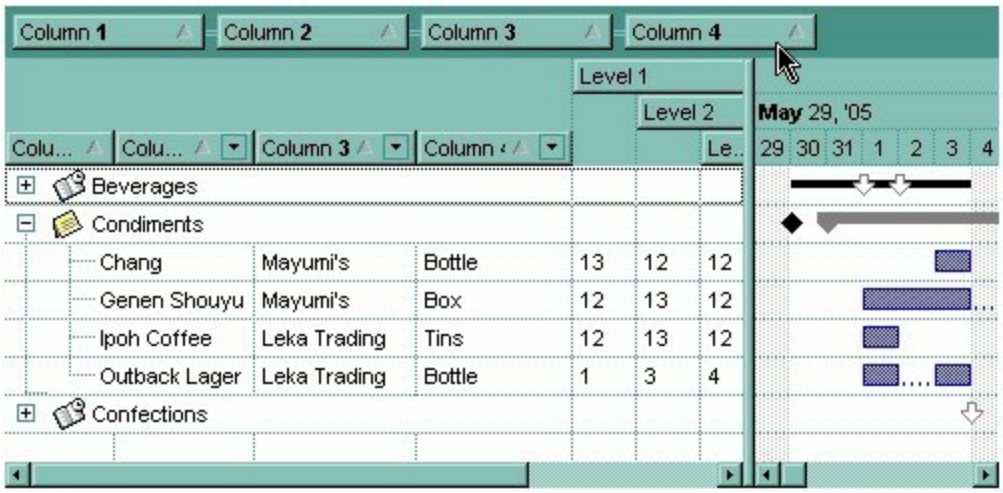
Use the SortBarVisible property to show the control's sort bar. By default, the SortBarVisible property is False. Use the SingleSort property to specify whether the control supports sorting by single or multiple columns. Sorting by a single column in the control is a simple matter of clicking on the column head. Sorting by multiple columns, however, is not so obvious. But it's actually quite easy. The user has two options to sort by multiple columns:

- First, sort by the first criterion, by clicking on the column head. Then hold the SHIFT key down as you click on a second heading.
- Click the column head and drag to the control's sort bar in the desired position.

The HeaderEnabled property enables or disables the control's header (including the control's sort/groupby-bar)

The control's sort bar displays the SortBarCaption expression, when it contains no columns, like follows (the "Drag a **column** header ..." area is the control's sort bar) :

The sort bar displays the list of columns being sorted in their order as follows:



The SortOrder property adds or removes programmatically columns in the control's sort bar. Use the SortPosition property to specify the position of the column in the sorting columns collection. Use the ItemBySortPosition property to access the columns being sorted. Use the SortOnClick property to specify the action that control should execute when user clicks the column's header. Use the AllowSort property to specify whether the user sorts a column by clicking the column's header. The control fires the Sort event when the user sorts a column. Use the Chart object to access all properties and methods related to the G2antt chart. Use the OverviewVisible property to show or hide the chart's overview area.

property G2antt.SortOnClick as SortOnClickEnum

Retrieves or sets a value that indicates whether the control sorts automatically the data when the user click on column's caption.

Type	Description
SortOnClickEnum	A SortOnClick expression that indicates whether the control sorts automatically the data when the user click on the column's header.

Use the SortOnClick property to disable sorting items when the user clicks on the column's header. Use the SortBarVisible property to show the control's sort bar. Use the SingleSort property to allow sorting by single or multiple columns. Use the AllowSort property to avoid sorting a column when user clicks the column. Use the DefaultSortOrder property to specify the column's default sort order, when the user first clicks the column's header.

There are two methods to get the items sorted like follows:

- Using the SortOrder property of the Column object::

  ```
  G2antt1.Columns(ColIndex).SortOrder = SortAscending
  ```

  The SortOrder property adds the sorting icon to the column's header, if the DisplaySortIcon property is True.

- Using the SortChildren method of the Items collection. The SortChildren sorts the items. The SortChildren method sorts the child items of the given parent item in the control. SortChildren will not recourse through the tree, only the immediate children of the item will be sorted. The following sample sorts descending the list of root items on the "Column 1"(if your control displays a list, all items are considered being root items).

  ```
  G2antt1.Items.SortChildren 0, "Column 1", False
  ```

The control fires the Sort event when the control sorts a column (the user clicks the column's head) or when the sorting position is changed in the control's sort bar. Use the Sort event to sort the data when the SortOnClk property is exUserSort.

property G2antt.Statistics as String

Gives statistics data of objects being hold by the control.

Type	Description
String	A String expression that gives information about objects being loaded into the control.

The Statistics property gives statistics data of objects being hold by the control. The Statistics property gives a rough idea on how many columns, items, cell, bars, links, notes and so on are loaded into the control. Also, the Statistics property gives percentage usage of base-memory of different objects within the memory.

The following output shows how the Statistics looks like, on a 32-bits machine:

```
Cells: 1,817,609 x 69 = 125,415,021 (90.16%)
Link: 16,986 x 440 = 7,473,840 (5.37%)
Item-Bars: 16,987 x 250 = 4,246,750 (3.05%)
Item: 16,987 x 106 = 1,800,622 (1.29%)
Column: 107 x 1,104 = 118,128 (0.08%)
Control: 1 x 30,352 = 30,352 (0.02%)
Charts: 1 x 9,552 = 9,552 (0.01%)
Bar: 7 x 944 = 6,608 (0.00%)
Levels: 1 x 1,488 = 1,488 (0.00%)
Level: 2 x 712 = 1,424 (0.00%)
Items: 1 x 852 = 852 (0.00%)
InsideZooms: 1 x 424 = 424 (0.00%)
Links: 1 x 256 = 256 (0.00%)
Columns: 1 x 172 = 172 (0.00%)
Notes: 1 x 36 = 36 (0.00%)
Appearances: 1 x 28 = 28 (0.00%)
Bars: 1 x 28 = 28 (0.00%)
Appearance: 0 x 712 = 0 (0.00%)
CComVariant: 0 x 16 = 0 (0.00%)
Cells(Inner): 0 x 69 = 0 (0.00%)
Chart: 0 x 9,560 = 0 (0.00%)
CSmartVariant: 0 x 9 = 0 (0.00%)
InsideZoom: 0 x 96 = 0 (0.00%)
Note: 0 x 672 = 0 (0.00%)
```

The following output shows how the Statistics looks like, on a 64-bits machine:

```
Cells: 1,817,609 x 121 = 219,930,689 (91.62%)
Link: 16,986 x 632 = 10,735,152 (4.47%)
Item-Bars: 16,987 x 342 = 5,809,554 (2.42%)
Item: 16,987 x 194 = 3,295,478 (1.37%)
Column: 107 x 1,808 = 193,456 (0.08%)
Control: 1 x 49,336 = 49,336 (0.02%)
Charts: 1 x 15,240 = 15,240 (0.01%)
Bar: 7 x 1,560 = 10,920 (0.00%)
Handles: 236 x 12 = 2,832 (0.00%)
Level: 2 x 1,200 = 2,400 (0.00%)
Levels: 1 x 2,224 = 2,224 (0.00%)
Items: 1 x 1,640 = 1,640 (0.00%)
InsideZooms: 1 x 632 = 632 (0.00%)
Links: 1 x 408 = 408 (0.00%)
Columns: 1 x 320 = 320 (0.00%)
Notes: 1 x 64 = 64 (0.00%)
Appearances: 1 x 48 = 48 (0.00%)
Bars: 1 x 48 = 48 (0.00%)
Appearance: 0 x 1,168 = 0 (0.00%)
CComVariant: 0 x 24 = 0 (0.00%)
Cells(Inner): 0 x 121 = 0 (0.00%)
Chart: 0 x 15,248 = 0 (0.00%)
CSmartVariant: 0 x 9 = 0 (0.00%)
InsideZoom: 0 x 136 = 0 (0.00%)
Note: 0 x 984 = 0 (0.00%)
```

property G2antt.Template as String

Specifies the control's template.

Type	Description
String	A string expression that indicates the control's template.

The control's template uses the X-Script language to initialize the control's content. Use the Template property page of the control to update the control's Template property. Use the Template property to execute code by passing instructions as a string (template string). Use the ExecuteTemplate property to execute a template script and gets the result.

Most of our UI components provide a Template page that's accessible in design mode. No matter what programming language you are using, you can have a quick view of the component's features using the WYSWYG Template editor.

- Place the control to your form or dialog.
- Locate the Properties item, in the control's context menu, in design mode. If your environment doesn't provide a Properties item in the control's context menu, please try to locate in the Properties browser.
- Click it, and locate the Template page.
- Click the Help button. In the left side, you will see the component, in the right side, you will see a x-script code that calls methods and properties of the control.

The control's Template page helps user to initialize the control's look and feel in design mode, using the x-script language that's easy and powerful. The Template page displays the control on the left side of the page. On the right side of the Template page, a simple editor is displayed where user writes the initialization code. The control's look and feel is automatically updated as soon as the user types new instructions. The Template script is saved to the container persistence (when Apply button is pressed), and it is executed when the control is initialized at runtime. Any component that provides a WYSWYG Template page, provides a Template property. The Template property executes code from a string (template string).

The Template or x-script is composed by lines of instructions. Instructions are separated by "\n\r" (newline characters) or ";" character. The ; character may be available only for newer versions of the components.

An x-script instruction/line can be one of the following:

- **Dim** list of variables *Declares the variables. Multiple variables are separated by commas. (Sample: Dim h, h1, h2)*
- variable **=** property(list of arguments) *Assigns the result of the property to a variable. The "variable" is the name of a declared variable. The "property" is the property name*

of the object in the context. The "list or arguments" may include variables or values separated by commas. (Sample: h = InsertItem(0,"New Child"))
- property**(** list of arguments **)** = value *Changes the property. The value can be a variable, a string, a number, a boolean value or a RGB value.*
- method**(** list of arguments **)** *Invokes the method. The "list or arguments" may include variables or values separated by commas.*
- **{** *Beginning the object's context. The properties or methods called between { and } are related to the last object returned by the property prior to { declaration.*
- **}** *Ending the object's context*
- object**.** property(list of arguments)**.**property(list of arguments)**....** *The .(dot) character splits the object from its property. For instance, the Columns.Add("Column1").HeaderBackColor = RGB(255,0,0), adds a new column and changes the column's header back color.*

The x-script may uses constant expressions as follow:

- *boolean* expression with possible values as *True* or *False*
- *numeric* expression may starts with 0x which indicates a hexa decimal representation, else it should starts with digit, or +/- followed by a digit, and . is the decimal separator. *Sample: 13 indicates the integer 13, or 12.45 indicates the double expression 12,45*
- *date* expression is delimited by **#** character in the format #mm/dd/yyyy hh:mm:ss#. *Sample: #31/12/1971# indicates the December 31, 1971*
- *string* expression is delimited by **"** or ` characters. If using the ` character, please make sure that it is different than ' which allows adding comments inline. *Sample: "text" indicates the string text.*

Also , the template or x-script code may support general functions as follows:

- **Me** *property indicates the original object.*
- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the R G B values for the color being specified. For instance, the following code changes the control's background color to red: BackColor = RGB(255,0,0)*
- **LoadPicture(**file**)** *property loads a picture from a file or from BASE64 encoded strings, and returns a Picture object required by the picture properties.*
- **CreateObject(**progID**)** *property creates and retrieves a single uninitialized object of the class associated with a specified program identifier.*

property G2antt.TemplateDef as Variant

Defines inside variables for the next Template/ExecuteTemplate call.

Type	Description
Variant	A string expression that indicates the Dim declaration, or any Object expression to be assigned to previously declared variables.

The TemplateDef property / TemplatePut method has been added to allow programming languages such as dBASE Plus to set control's properties with multiple parameters. It is known that programming languages such as **dBASE Plus or XBasic from AlphaFive**, does not support setting a property with multiple parameters. In other words, these programming languages does not support something like *Property(Parameters)* = *Value*, so our controls provide an alternative using the TemplateDef / TemplatePut method. The first call of the TemplateDef should be a declaration such as "Dim a,b" which means the next 2 calls of the TemplateDef defines the variables a and b. The next call should be Template or ExecuteTemplate property which can use the variable a and b being defined previously.

So, calling the TemplateDef property should be as follows:

```
with (Control)
   TemplateDef = [Dim var_Column]
   TemplateDef = var_Column
   Template = [var_Column.Def(4) = 255]
endwith
```

This sample allocates a variable var_Column, assigns the value to the variable (the second call of the TemplateDef), and the Template call uses the var_Column variable (as an object), to call its Def property with the parameter 4.

Let's say we need to define the background color for a specified column, so we need to call the Def(exCellBackColor) property of the column, to define the color for all cells in the column.

The following **VB6** sample shows setting the Def property such as:

```
With Control
   .Columns.Add("Column 1").Def(exCellBackColor) = 255
   .Columns.Add "Column 2"
   .Items.AddItem 0
   .Items.AddItem 1
```

```
   .Items.AddItem 2
End With
```

In **dBASE Plus**, calling the Def(4) has no effect, instead using the TemplateDef helps you to use properly the Def property as follows:

```
local Control,var_Column

Control = form.Activex1.nativeObject
// Control.Columns.Add("Column 1").Def(4) = 255
var_Column = Control.Columns.Add("Column 1")
with (Control)
   TemplateDef = [Dim var_Column]
   TemplateDef = var_Column
   Template = [var_Column.Def(4) = 255]
endwith
Control.Columns.Add("Column 2")
Control.Items.AddItem(0)
Control.Items.AddItem(1)
Control.Items.AddItem(2)
```

The equivalent sample for **XBasic in A5**, is as follows:

```
Dim Control as P
Dim var_Column as P

Control = topparent:CONTROL_ACTIVEX1.activex
' Control.Columns.Add("Column 1").Def(4) = 255
var_Column = Control.Columns.Add("Column 1")
Control.TemplateDef = "Dim var_Column"
Control.TemplateDef = var_Column
Control.Template = "var_Column.Def(4) = 255"

Control.Columns.Add("Column 2")
Control.Items.AddItem(0)
Control.Items.AddItem(1)
Control.Items.AddItem(2)
```

The samples just call the Column.Def(4) = Value, using the TemplateDef. The first call of TemplateDef property is "Dim var_Column", which indicates that the next call of the TemplateDef will defines the value of the variable var_Column, in other words, it defines the object var_Column. The last call of the Template property uses the var_Column member to use the x-script and so to set the Def property so a new color is being assigned to the column.

The TemplateDef, TemplatePut, Template and ExecuteTemplate support x-script language (Template script of the Exontrols), like explained bellow:

The Template or x-script is composed by lines of instructions. Instructions are separated by "\n\r" (newline characters) or ";" character. The ; character may be available only for newer versions of the components.

An x-script instruction/line can be one of the following:

- **Dim** list of variables *Declares the variables. Multiple variables are separated by commas. (Sample: Dim h, h1, h2)*
- variable **=** property(list of arguments) *Assigns the result of the property to a variable. The "variable" is the name of a declared variable. The "property" is the property name of the object in the context. The "list or arguments" may include variables or values separated by commas. (Sample: h = InsertItem(0,"New Child"))*
- property**(** list of arguments **)** = value *Changes the property. The value can be a variable, a string, a number, a boolean value or a RGB value.*
- method**(** list of arguments **)** *Invokes the method. The "list or arguments" may include variables or values separated by commas.*
- **{** *Beginning the object's context. The properties or methods called between { and } are related to the last object returned by the property prior to { declaration.*
- **}** *Ending the object's context*
- object**.** property(list of arguments)**.**property(list of arguments)**....** *The .(dot) character splits the object from its property. For instance, the Columns.Add("Column1").HeaderBackColor = RGB(255,0,0), adds a new column and changes the column's header back color.*

The x-script may uses constant expressions as follow:

- *boolean* expression with possible values as *True* or *False*
- *numeric* expression may starts with 0x which indicates a hexa decimal representation, else it should starts with digit, or +/- followed by a digit, and . is the decimal separator. *Sample: 13 indicates the integer 13, or 12.45 indicates the double expression 12,45*
- *date* expression is delimited by **#** character in the format #mm/dd/yyyy hh:mm:ss#. *Sample: #31/12/1971# indicates the December 31, 1971*
- *string* expression is delimited by **"** or ` characters. If using the ` character, please

make sure that it is different than ' which allows adding comments inline. *Sample: "text" indicates the string text.*

Also , the template or x-script code may support general functions as follows:

- **Me** *property indicates the original object.*
- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the R G B values for the color being specified. For instance, the following code changes the control's background color to red: BackColor = RGB(255,0,0)*
- **LoadPicture(**file**)** *property loads a picture from a file or from BASE64 encoded strings, and returns a Picture object required by the picture properties.*
- **CreateObject(**progID**)** *property creates and retrieves a single uninitialized object of the class associated with a specified program identifier.*

method G2antt.TemplatePut (NewVal as Variant)

Defines inside variables for the next Template/ExecuteTemplate call.

Type	Description
NewVal as Variant	A string expression that indicates the Dim declaration, or any Object expression to be assigned to previously declared variables.

The TemplatePut method / TemplateDef property has been added to allow programming languages such as dBASE Plus to set control's properties with multiple parameters. It is known that programming languages such as **dBASE Plus or XBasic from AlphaFive**, does not support setting a property with multiple parameters. In other words, these programming languages does not support something like *Property(Parameters) = Value*, so our controls provide an alternative using the TemplateDef / TemplatePut method. The first call of the TemplateDef should be a declaration such as "Dim a,b" which means the next 2 calls of the TemplateDef defines the variables a and b. The next call should be Template or ExecuteTemplate property which can use the variable a and b being defined previously.

The TemplateDef, TemplatePut, Template and ExecuteTemplate support x-script language (Template script of the Exontrols), like explained bellow:

The Template or x-script is composed by lines of instructions. Instructions are separated by "\n\r" (newline characters) or ";" character. The ; character may be available only for newer versions of the components.

An x-script instruction/line can be one of the following:

- **Dim** list of variables *Declares the variables. Multiple variables are separated by commas. (Sample: Dim h, h1, h2)*
- variable **=** property(list of arguments) *Assigns the result of the property to a variable. The "variable" is the name of a declared variable. The "property" is the property name of the object in the context. The "list or arguments" may include variables or values separated by commas. (Sample: h = InsertItem(0,"New Child"))*
- property**(** list of arguments **)** = value *Changes the property. The value can be a variable, a string, a number, a boolean value or a RGB value.*
- method**(** list of arguments **)** *Invokes the method. The "list or arguments" may include variables or values separated by commas.*
- **{** *Beginning the object's context. The properties or methods called between { and } are related to the last object returned by the property prior to { declaration.*
- **}** *Ending the object's context*
- object**.** property(list of arguments)**.**property(list of arguments)…. *The .(dot) character splits the object from its property. For instance, the*

Columns.Add("Column1").HeaderBackColor = RGB(255,0,0), adds a new column and changes the column's header back color.

The x-script may uses constant expressions as follow:

- *boolean* expression with possible values as *True* or *False*
- *numeric* expression may starts with 0x which indicates a hexa decimal representation, else it should starts with digit, or +/- followed by a digit, and . is the decimal separator. *Sample: 13 indicates the integer 13, or 12.45 indicates the double expression 12,45*
- *date* expression is delimited by **#** character in the format #mm/dd/yyyy hh:mm:ss#. *Sample: #31/12/1971# indicates the December 31, 1971*
- *string* expression is delimited by **"** or ` characters. If using the ` character, please make sure that it is different than ' which allows adding comments inline. *Sample: "text" indicates the string text.*

Also , the template or x-script code may support general functions as follows:

- **Me** *property indicates the original object.*
- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the R G B values for the color being specified. For instance, the following code changes the control's background color to red: BackColor = RGB(255,0,0)*
- **LoadPicture(**file**)** *property loads a picture from a file or from BASE64 encoded strings, and returns a Picture object required by the picture properties.*
- **CreateObject(**progID**)** *property creates and retrieves a single uninitialized object of the class associated with a specified program identifier.*

property G2antt.TooltipCellsColor as Color

Retrieves or sets a value that indicates the color used to mark the cells that have tool tips.

Type	Description
Color	A color expression that specifies the color used to mark the cells that have a tool tip associated.

The property has effect only if the MarkTooltipCells property is True. Use the CellToolTip property to assign a tooltip to a cell. Use the ToolTipWidth property to specify the width of the tooltip window. The control fires the ToolTip event when the column's tooltip is about to be displayed.

property G2antt.ToolTipDelay as Long

Specifies the time in ms that passes before the ToolTip appears.

Type	Description
Long	A long expression that specifies the time in ms that passes before the ToolTip appears.

If the ToolTipDelay or ToolTipPopDelay property is 0, the control displays no tooltips. The ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. The ToolTipMargin property defines the size of the control's tooltip margins. Use the ToolTipWidth property to specify the width of the tooltip window. Use the ToolTipFont property to assign a font for the control's tooltip. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color. Use the CellToolTip property to specify the cell's tooltip. Use the ItemBar(,,exBarToolTip) property to specify a tooltip for a bar. Use the Link(,exLinkToolTip) property to specify the tooltip to be shown when the cursor hovers the link.

The following sample shows how you can temporarily/programmatically hide the control's tooltip:

```
Public Sub hideToolTip(ByRef g As EXG2ANTTLib.G2antt)
   Dim nToolTipDelay As Long
   With g
      nToolTipDelay = .ToolTipDelay
      .ToolTipDelay = 0
      .ToolTipDelay = nToolTipDelay
   End With
End Sub
```

property G2antt.ToolTipFont as IFontDisp

Retrieves or sets the tooltip's font.

Type	Description
IFontDisp	A Font object being used to display the tooltip.

Use the ToolTipFont property to assign a font for the control's tooltip. The ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipWidth property to specify the width of the tooltip window. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the <font> HTML element to assign a different font for portions of text inside the tooltip. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color. Use the ShowToolTip method to display a custom tooltip. Use the CellToolTip property to specify the cell's tooltip. Use the ItemBar(,,exBarToolTip) property to specify a tooltip for a bar. Use the Link(,exLinkToolTip) property to specify the tooltip to be shown when the cursor hovers the link.

property G2antt.ToolTipMargin as String

Defines the size of the control's tooltip margins.

Type	Description
String	A string expression that defines the horizontal and vertical margins (separated by comma) of the control's tooltip as one of the following formats: • "value", where value is a positive number, that specifies the horizontal and vertical margins, such as "4" equivalent of "4,4" • "value,", where value is a positive number, that specifies the horizontal margin, such as "4," equivalent of "4,0" • ",value", where value is a positive number, that specifies the vertical margin, such as ",4" equivalent of "0,4" • "horizontal,vertical", where horizontal and vertical are positive numbers, that specifies the horizontal and vertical margins, such as "4,4"

By default, the size of the tooltip margin is "4" (horizontal and vertical). For instance, ToolTipMargin = "8" changes the horizontal and vertical margins are set to 8 pixels. ToolTipMargin = "8,4" changes the horizontal margin to 8 pixels and the vertical margin to 4 pixels. The ToolTipWidth property specifies a value that indicates the width of the tooltip window, in pixels. Use the ShowToolTip method to display a custom tooltip. Use the ToolTipFont property to assign a font for the control's tooltip. The ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears.

property G2antt.ToolTipPopDelay as Long

Specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control.

Type	Description
Long	A long expression that specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control.

If the ToolTipDelay or ToolTipPopDelay property is 0, the control displays no tooltips. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the ToolTipWidth property to specify the width of the tooltip window. The ToolTipMargin property defines the size of the control's tooltip margins. Use the ToolTipFont property to assign a font for the control's tooltip. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color. Use the ShowToolTip method to display a custom tooltip. Use the CellToolTip property to specify the cell's tooltip. Use the ItemBar(,,exBarToolTip) property to specify a tooltip for a bar. Use the Link(,exLinkToolTip) property to specify the tooltip to be shown when the cursor hovers the link.

The following sample shows how you can temporarily/programmatically hide the control's tooltip:

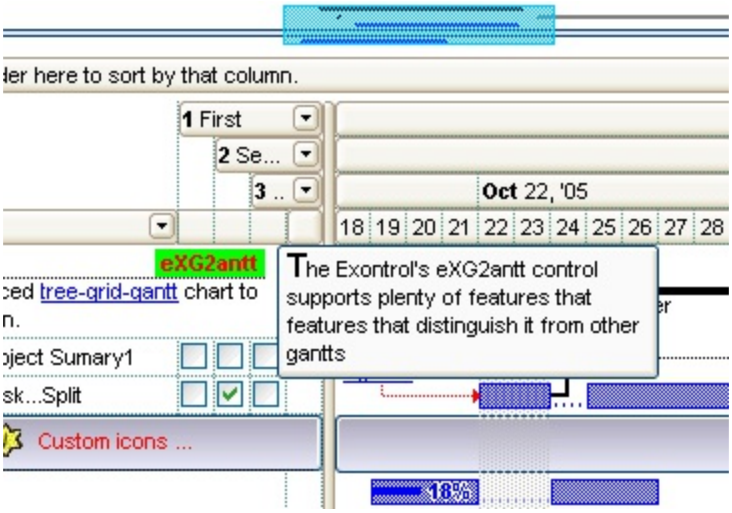
```
Public Sub hideToolTip(ByRef g As EXG2ANTTLib.G2antt)
   Dim nToolTipDelay As Long
   With g
      nToolTipDelay = .ToolTipDelay
      .ToolTipDelay = 0
      .ToolTipDelay = nToolTipDelay
   End With
End Sub
```

property G2antt.ToolTipWidth as Long

Specifies a value that indicates the width of the tooltip window, in pixels.

Type	Description
Long	A long expression that indicates the width of the tooltip window.

Use the ToolTipWidth property to change the tooltip window width. The height of the tooltip window is automatically computed based on tooltip's description. The ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. The ToolTipMargin property defines the size of the control's tooltip margins. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color. Use the ShowToolTip method to display a custom tooltip. Use the ToolTipFont property to assign a font for the control's tooltip. Use the CellToolTip property to specify the cell's tooltip. Use the ItemBar(,,exBarToolTip) property to specify a tooltip for a bar. Use the Link(,exLinkToolTip) property to specify the tooltip to be shown when the cursor hovers the links.



property G2antt.TreeColumnIndex as Long

Retrieves or sets a value indicating the column's index where the hierarchy will be displayed.

Type	Description
Long	A long expression that indicates the index of the column where the control's hierarchy is displayed.

Use the TreeColumnIndex property to change the column's index where the hierarchy lines are painted. Use HasLines and LinesAtRoot properties to show the hierarchy lines. Use the HasButtons property to define the +/- signs appearance. If the TreeColumnIndex property is -1, the control doesn't paint the hierarchy. Use the Indent property to define the amount, in pixels, that child items are indented relative to their parent items.

method G2antt.Ungroup ()

Ungroups the columns, if they have been previously grouped.

Type	Description

property G2antt.UseTabKey as Boolean

Specifies whether the TAB key is used to change the searching column.

Type	Description
Boolean	A boolean expression that specifies whether the TAB key is used to change the incremental searching column.

By default, the UseTabKey property is True. The UseTabKey property specifies whether the control uses the TAB key to change the searching column. If the UseTabKey property is False, the TAB key is used to navigate through the form's controls.

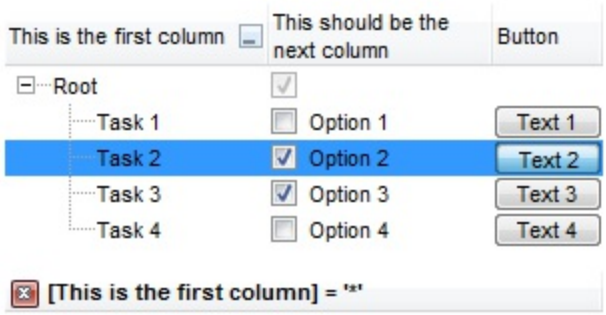
property G2antt.UseVisualTheme as UIVisualThemeEnum

Specifies whether the control uses the current visual theme to display certain UI parts.

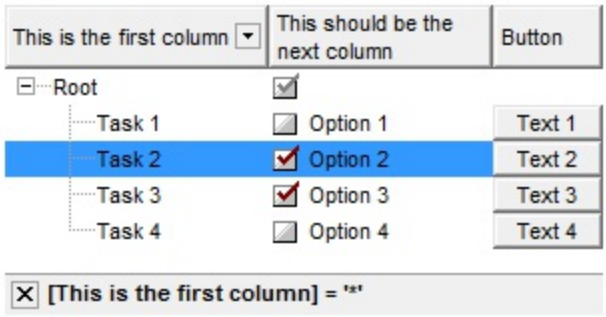
Type	Description
UIVisualThemeEnum	An UIVisualThemeEnum expression that specifies which UI parts of the control are shown using the current visual theme.

By default, the UseVisualTheme property is exDefaultVisualTheme, which means that all known UI parts are shown as in the current theme. The UseVisualTheme property may specify the UI parts that you need to enable or disable the current visual theme. The UI Parts are like header, filterbar, check-boxes, buttons and so on. The UseVisualTheme property has effect only a current theme is selected for your desktop. The UseVisualTheme property. Use the Appearance property of the control to provide your own visual appearance using the EBN files.

The following screen shot shows the control while the UseVisualTheme property is exDefaultVisualTheme:



since the second screen shot shows the same data as the UseVisualTheme property is exNoVisualTheme:



property G2antt.Version as String

Retrieves the control's version.

Type	Description
String	A string expression that indicates the control's version.

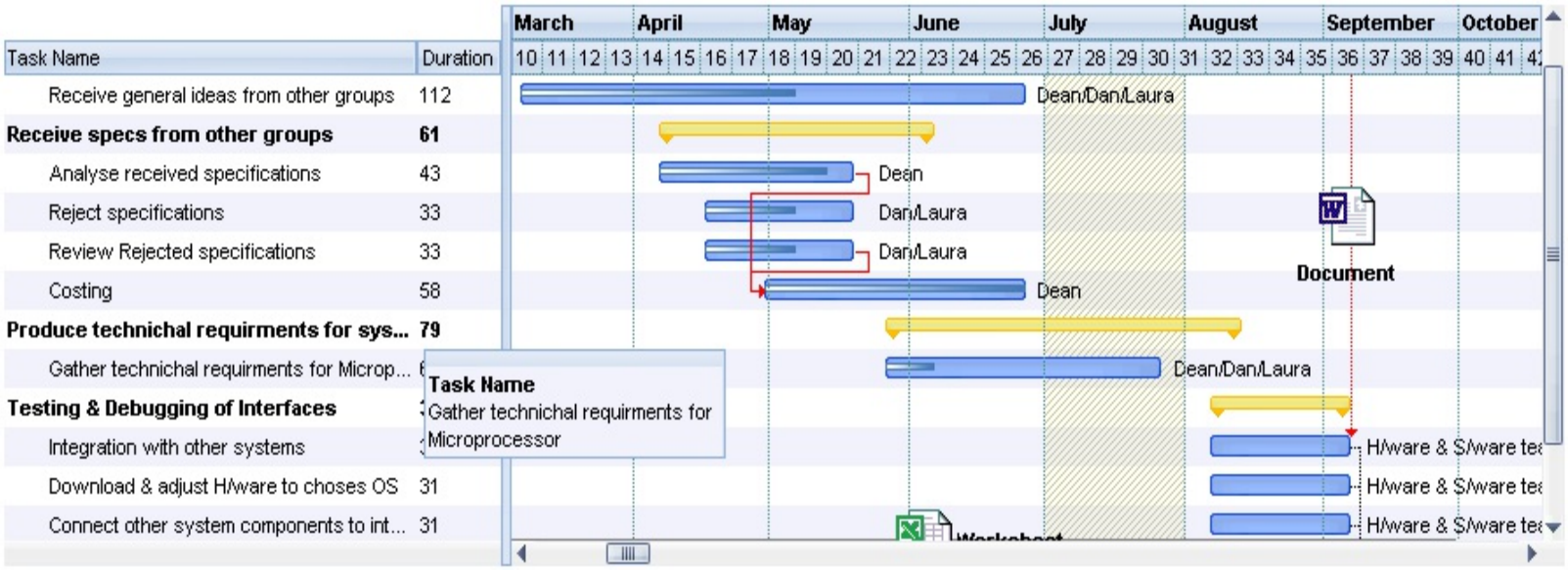
The version property specifies the control's version.

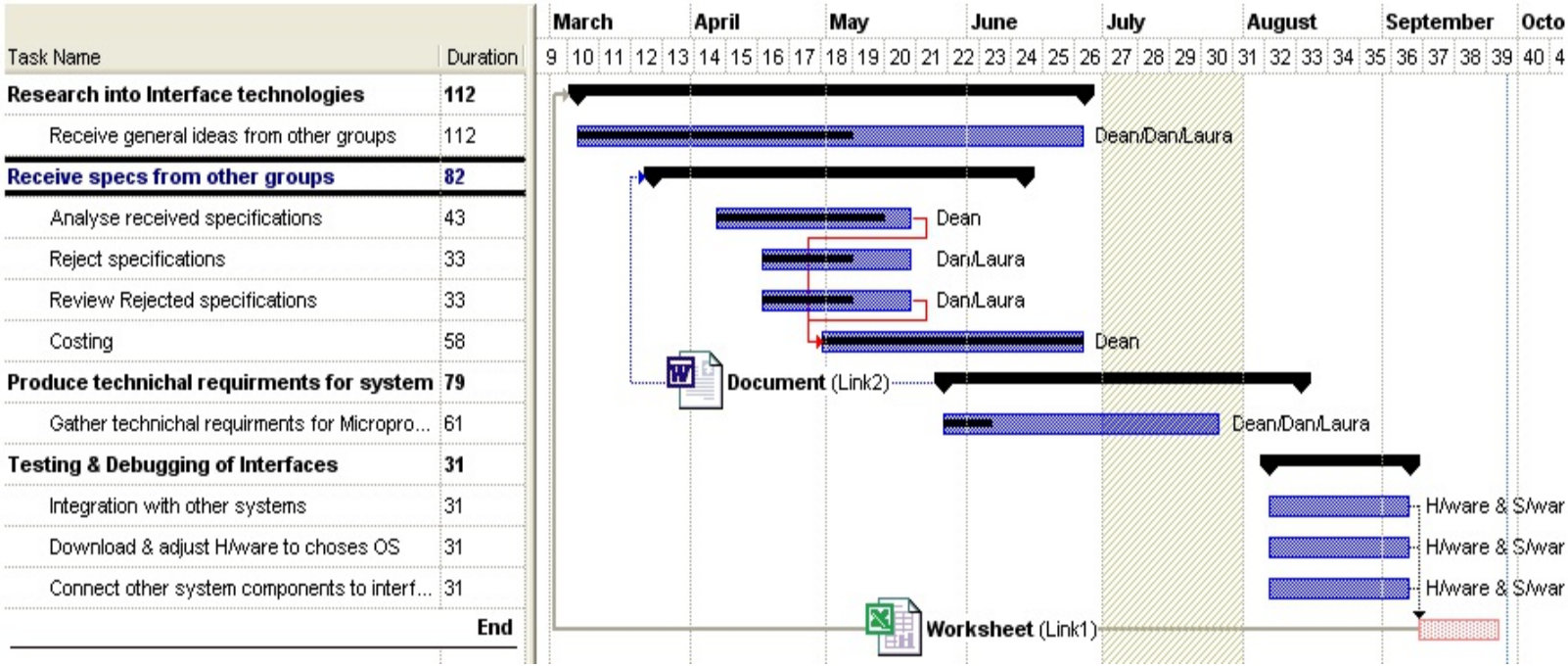
property G2antt.VisualAppearance as Appearance

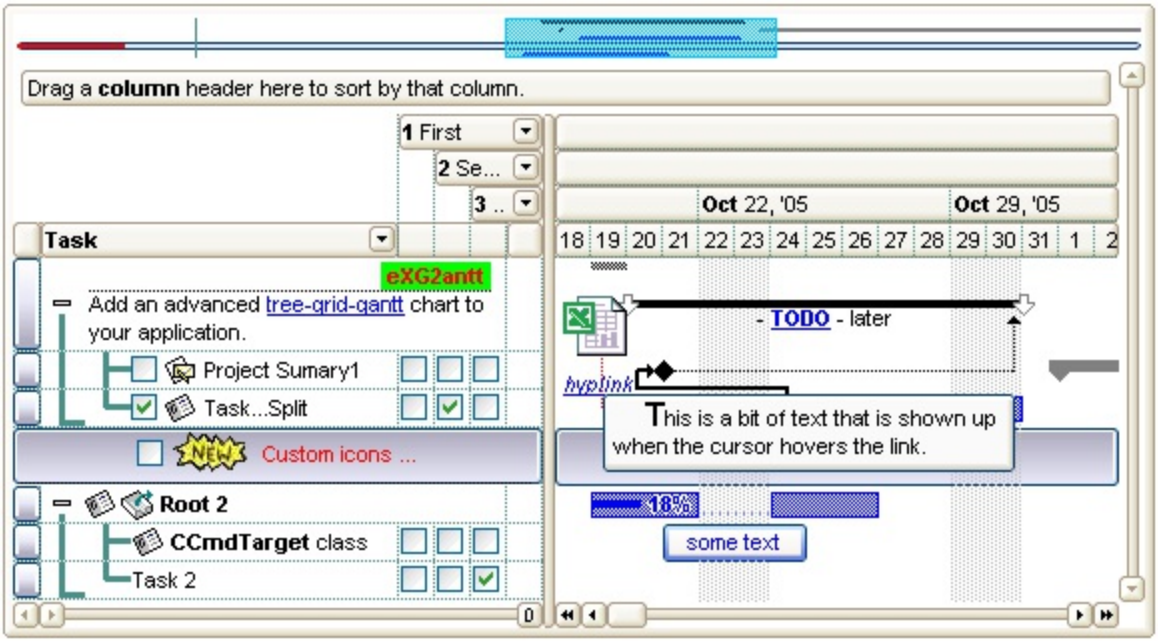
Retrieves the control's appearance.

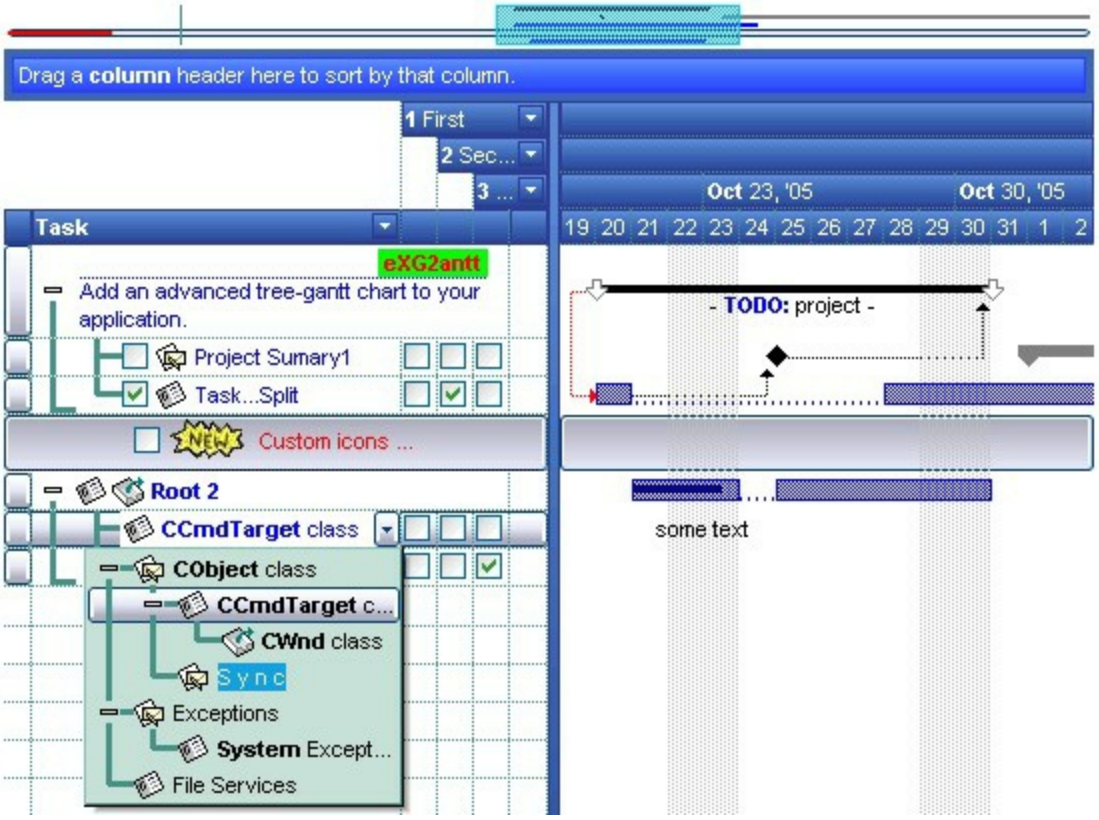
Type	Description
Appearance	An Appearance object that holds a collection of skins.

Use the Add method to add or replace skins to the control. The skin method, in it's simplest form, uses a single graphic file (*.ebn) assigned to a part of the control. By using a collection of objects laid over the graphic, it is possible to define which sections of the graphic will be used as borders, corners and other possible elements, fixing them to their proper position regardless of the size of the part.









The skin method may change the visual appearance for the following parts in the control:

- **levels** on the chart area, BackColor property, BackColorLevelHeader property
- bar's background, ItemBar(exBarBackColor) property
- control's **header bar**, BackColorHeader property
- control's **filter bar**, FilterBarBackColor property
- control's **sort bar**, BackColorSort property
- the caption of the control's sort bar, BackColorSortCaption property
- **selected item** or cell, SelBackColor property
- **item**, ItemBackColor property
- **cell**, CellBackColor property

- cell's **button**, "**drop down**" filter bar button, "close" filter bar button, and so on, Background property
- 'focus' box in the chart's overview area, OverviewSelBackColor property.

property G2antt.VisualDesign as String

Invokes the control's VisualAppearance designer.

Type	Description
String	A String expression that encodes the control's Visual Appearance.

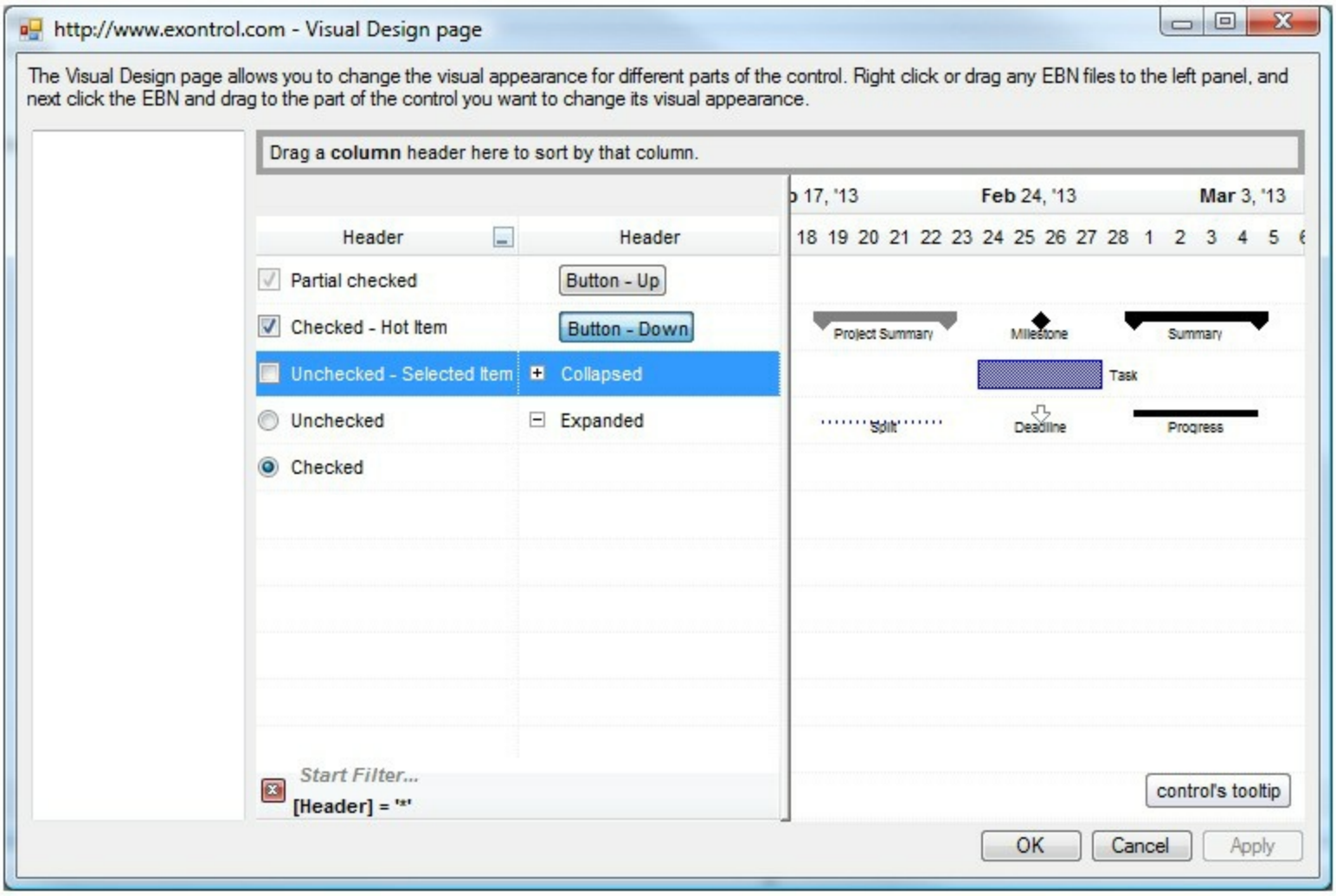
By default, the VisualDesign property is "". The VisualDesign property helps you to define fast and easy the control's visual appearance using the XP-Theme elements or EBN objects. The VisualDesign property can be accessed on design mode, and it can be used to design the visual appearance of different parts of the control by drag and drop XP or EBN elements. The VisualAppearance designer returns an encoded string that can be used to define different looks, just by calling the VisualDesign = encoded_string. If you require removing the current visual appearance, you can call the VisualDesign on "" (empty string). The VisualDesign property encodes EBN or XP-Theme nodes, using the Add method of the Appearance collection being accessed through the VisualAppearance property.

- For the /COM version, click the control in Design mode, select the Properties, and choose the "Visual Design" page.
- For the /NET version, select the VisualDesign property in the Properties browser, and then click ... so the "Visual Design" page is displayed.
- The /WPF version does not provide a VisualAppearance designer, instead you can use the values being generated by the /COM or /NET to apply the same visual appearance.

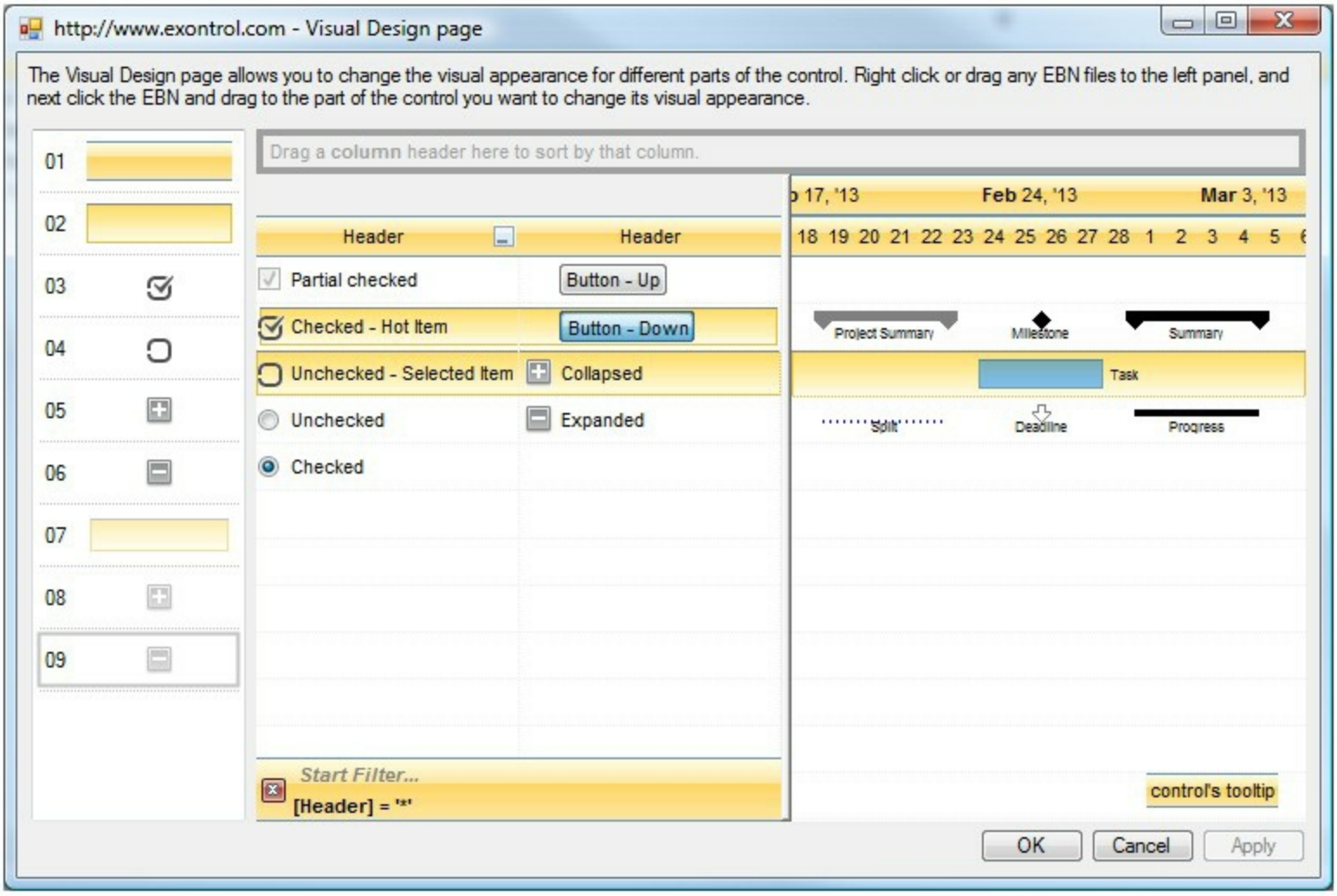
- Click here to watch a movie on how you define the control's visual appearance using the XP-Theme
- Click here to watch a movie on how you define the control's visual appearance using the EBN files.

The left panel, should be user to add your EBN or XP-Theme elements. Once you add them drag and drop the EBN or XP-Theme element from the left side to the part which visual appearance you want to change.

The following picture shows the control's VisualDesign form (empty):



The following picture shows the control's VisualDesign form after applying some EBN objects:

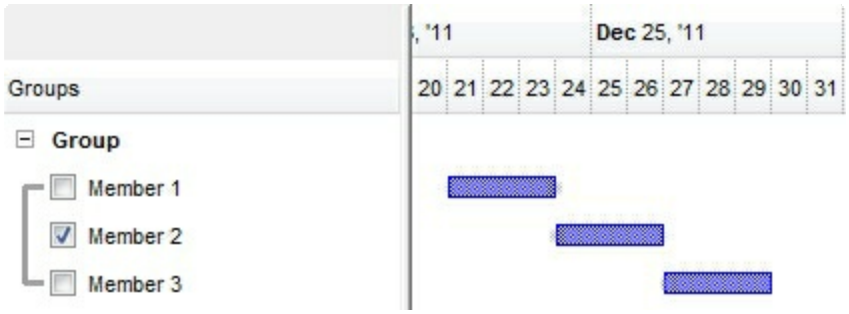


This layout generates the following code:

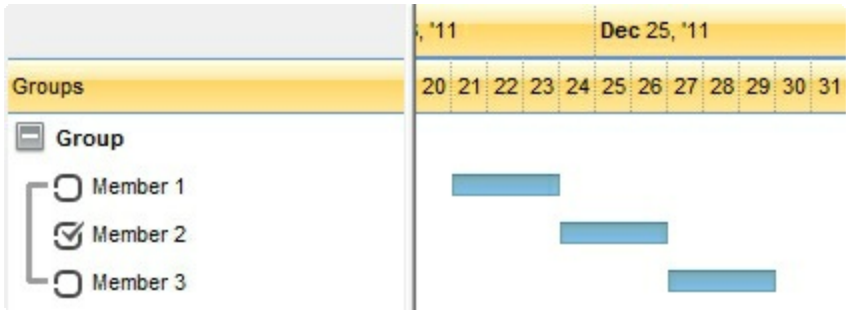
```
With Exg2antt1
   .VisualDesign =
"gBFLBWIgBAEHhEJAEGg7oB0HBSQAwABsIfj/jEJAcKhYEjgCAscA8ThQBA8cAgIjgDh8KBAPj
& _

"RuF6FxmAkchiheZg5gYZIW0yMhZhqD55jIboamcCY2HGG5nCmVh0h2ZYUAyCQ4Xqbh9h8
& _

"o5B8MwE4HsD4/g/ijHQHoLwrxUjrH0H4Z4rR2h7A8N8UggRNBnGCP8eA/A/gXGSPMfg3w
& _

"DCDgJQFICxhDQGYBofYQYFCwD4J+XYQwIBECiCwJIExhhnCIDoNAnhzj8CyBcIosQ+BlAwM
& _

"J8YQIwaBMCaCMd6hRnBpE+HoIwIQ9hdEKM8VYawoCcC8BUSYtxqBuDuFsOwTgLgLhZhAh
& _
```

```
"zhGhtoEB+AsArhnhLhehUB5BfA4BfARBPgWB9h3hhBZB/AvA+BzhkhLhCh7hPg8g1BfhzAKE
 & _

"hQH1hSgAgcAmghgIg2AugLBigiBqAnAzBiVdglA1ANAjBEgbAmAJMwA+gLgjgyBWA4A0E
 & _

"IAUgCA0AMhjA0ggWUgjh+GhBihI1yAKhiByBqAkV1gCAKAiV3141516g+Jmhj19V+V/AI2A

End With
```

If running the empty control we get the following picture:



If running the control using the code being generated by the VisualAppearance designer we get:



property G2antt.WordFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS, Highlight as Boolean, [Reserved as Variant]) as String

Retrieves the word from the cursor.

Type	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Highlight as Boolean	A Boolean expression that indicates whether the word from the position is highlighted.
Reserved as Variant	A long expression that specifies the part/parts of the control to search for the word from the cursor.
String	A String expression that indicates the word from the cursor.

Use the WordFromPoint property to retrieve the word from the cursor. The X and Y coordinates are expressed in client coordinates, so a conversion must be done in case your coordinates are relative to the screen or to other window. **If the X parameter is -1 and Y parameter is -1 the WordFromPoint property determines the index word from the cursor**. By default, the WordFromPoint property looks for the words in the Items area but it can search for words in any part of the control (Reserved parameter). Shortly, in the area where the items are displayed. A word is being defined as the sequence of the characters between two space/tab characters (empty characters). The word being returned does not include any HTML tags, in case the cursor hovers an HTML text. Use the ItemFromPoint property to get the item or cell from the cursor. Use the BarFromPoint property to get the bar from the point. Use the LinkFromPoint property to get the link from the point. Use the AnchorFromPoint property to determine the identifier of the anchor from the point. Use the CellCaption property to retrieve the entire cell's caption. Use the CellValue property to retrieve the cell's value. The ShowToolTip property shows programmatically your text as a tooltip.

The following VB sample displays the word from the cursor:

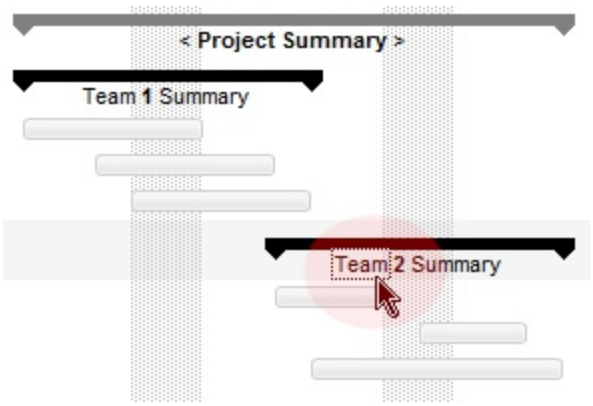
```
Private Sub G2antt1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
  Debug.Print G2antt1.WordFromPoint(-1, -1)
```

```
End Sub
```

The following VB sample highlights the word from the cursor, as soon as the cursor hovers a word:

```
Private Sub G2antt1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Debug.Print G2antt1.WordFromPoint(-1, -1, True)
End Sub
```

The following screen shot shows the "Team" word being highlighted when the cursor hovers it:



As soon as the cursor hovers another word it gets highlighted. The highlighting is temporary so as soon as the control is repainted the highlight is lost. For instance, you resize a column, scroll, or select a new item.

The following VB sample highlight the word from the cursor, and displays a context menu (eXPopupMenu) when the user right clicks the control:

```
Private Sub G2antt1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Debug.Print G2antt1.WordFromPoint(-1, -1, True)
End Sub

Private Sub G2antt1_RClick()
    Dim i As Long
    With PopupMenu1.Items
        .Clear
        .Add G2antt1.WordFromPoint(-1, -1, True)
    End With
    i = PopupMenu1.ShowAtCursor()
```

```
End Sub
```

Currently, the Reserved parameter could be a combination (bitwise OR) of any of the following:

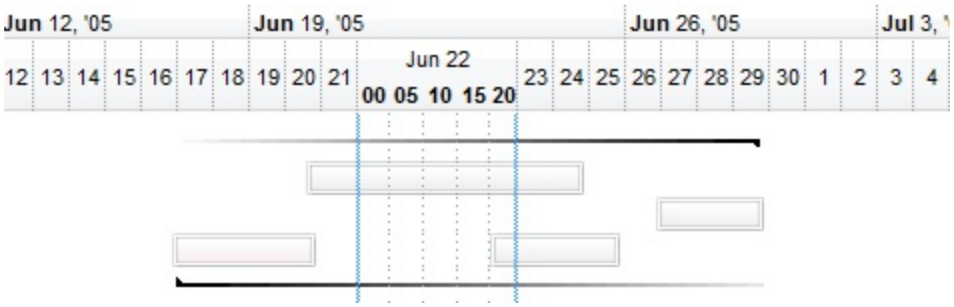
- (1) - Items area (the area where the items are shown)
- (2) - Header area (the area where the column's caption is displayed)
- (4) - SortBar area (the part of the control that shows the sorting columns, if multiple columns sorting is enabled)
- (8) - FilterBar area (the part of the control that displays the filter bar)
- (256) - Histogram area (the part of the control that displays the chart's histogram)

For instance, if the Reserved parameter is 1 + 2 (3), the WordFromPoint property retrieves the word from Items or Header area as well. If missing the control looks for the word in the Items section.

InsideZoom object

The InsideZoom object holds a unit being zoomed. The InsideZooms property retrieves the collection of InsideZoom objects. The inside zoom feature allows displaying portions of the chart with different time scale units. For instance, you can display the bars on hours, while the chart still displays days. The chart may display any unit in a different format. The InsideZoom objects get displayed ONLY of the AllowInsideZoom property is true.

The following screen shot shows the hours for Jun 22, while the rest of the chart displays days:



The InsideZoom object supports the following properties and methods.

Name	Description
AllowCustomFormat	Gets or sets a value that indicates whether the time unit displays a custom format or default format.
AllowInsideFormat	Specifies whether the unit allows displaying the inside format.
AllowResize	Specifies whether the user can resize the date/time unit.
CustomFormat	Gets the custom format of the time unit.
EndDate	Returns the date where the zoom unit ends.
StartDate	Returns the date where the zoom unit begins.
Width	Specifies the width of the date/time in the chart.

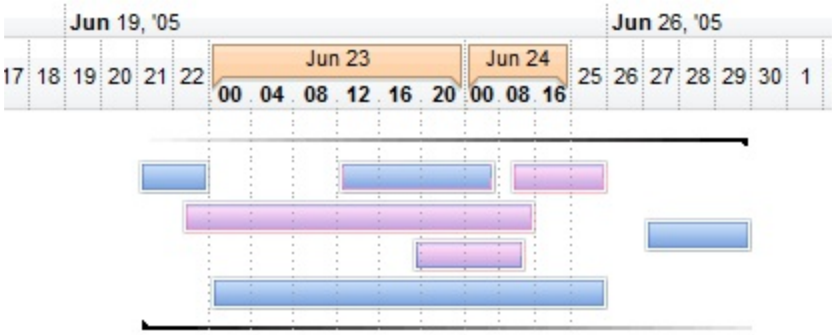
property InsideZoom.AllowCustomFormat as Boolean

Gets or sets a value that indicates whether the time unit displays a custom format or default format.

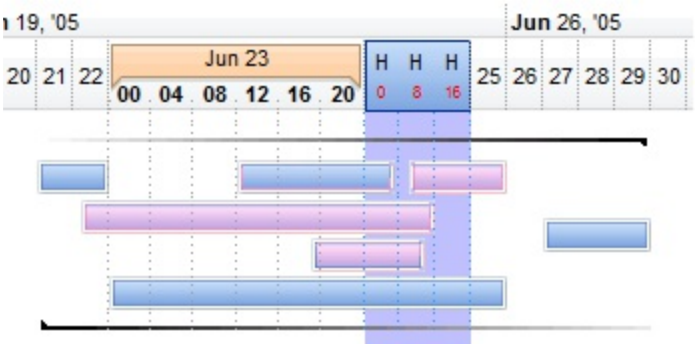
Type	Description
Boolean	A Boolean expression that specifies whether the unit uses the CustomFormat property to define a new visual appearance for the unit.

By default, the AllowCustomFormat property is False. Use the AllowCustomFormat and CustomFormat properties to define units with different appearances, including labels, background or foreground colors, and so on. If the AllowCustomFormat property is false, the inside zoom units use the DefaultInsideZoomFormat property to define the look and feel of the inside units. If the AllowCustomFormat property, the CustomFormat property defines a custom look and feel for the inside unit. The CustomFormat property retrieves nothing, if the AllowCustomFormat property is False. Once the AllowCustomFormat property is changed to True, it creates a new custom format object, that can be accessed later using the CustomFormat property. The new custom object copies the attributes of the DefaultInsideZoomFormat object.

The following screen shot shows the the units using the same format (June 23 and June 24), AllowCustomFormat property is False:



The following screen shot shows the the units using the different formats (June 23 and June 24), AllowCustomFormat property is True:



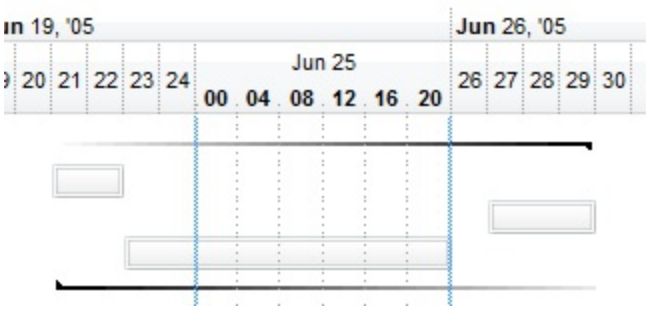
property InsideZoom.AllowInsideFormat as Boolean

Specifies whether the unit allows displaying the inside format.

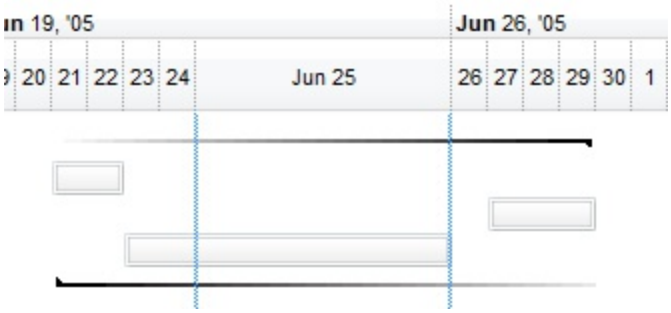
Type	Description
Boolean	A boolean expression that specifies whether the inside unit displays the new time scale unit, and so the InsideLabel property.

By default, the AllowInsideFormat property is True. If the AllowInsideFormat property is True, the level displays the new units base on the properties as InsideLabel, InsideUnit and InsideCount. If the AllowInsideFormat property is False, the unit does not display the new time scale unit, instead it displays the label as indicated by OwnerLabel property. In other words, you can use the AllowInsideFormat property on True, when you need to display a new time scale unit, or on False, when actually you need to change the visual appearance for specified unit. Use the SplitBaseLevel property to specify whether the base level is divided in two lines, so the first displays the owner label, and the second line displays the inside format. If the AllowInsideFormat property is False, the grid lines, the inside label, and the tick lines are not shown.

The following screen shot shows the inside zoom unit with AllowInsideFormat property on True:



The following screen shot shows the inside zoom unit with AllowInsideFormat property on False:



The following VB sample shows how can I change the foreground color for a time unit:

```
With G2antt1
```

```
  .BeginUpdate
  With .Chart
    .LevelCount = 2
    .FirstVisibleDate = #1/1/2008#
    .AllowInsideZoom = True
    .AllowResizeInsideZoom = False
    .InsideZoomOnDblClick = False
    .DefaultInsideZoomFormat.ForeColor = 255
    With .InsideZooms
      .SplitBaseLevel = False
      .DefaultWidth = 18
      .Add(#1/4/2008#).AllowInsideFormat = False
    End With
  End With
  .EndUpdate
End With
```

The following VB.NET sample shows how can I change the foreground color for a time unit:

```
With AxG2antt1
  .BeginUpdate
  With .Chart
    .LevelCount = 2
    .FirstVisibleDate = #1/1/2008#
    .AllowInsideZoom = True
    .AllowResizeInsideZoom = False
    .InsideZoomOnDblClick = False
    .DefaultInsideZoomFormat.ForeColor = 255
    With .InsideZooms
      .SplitBaseLevel = False
      .DefaultWidth = 18
      .Add(#1/4/2008#).AllowInsideFormat = False
    End With
  End With
  .EndUpdate
End With
```

The following C++ sample shows how can I change the foreground color for a time unit:

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutLevelCount(2);
   var_Chart->PutFirstVisibleDate("1/1/2008");
   var_Chart->PutAllowInsideZoom(VARIANT_TRUE);
   var_Chart->PutAllowResizeInsideZoom(VARIANT_FALSE);
   var_Chart->PutInsideZoomOnDblClick(VARIANT_FALSE);
   var_Chart->GetDefaultInsideZoomFormat()->PutForeColor(255);
   EXG2ANTTLib::IInsideZoomsPtr var_InsideZooms = var_Chart->GetInsideZooms();
      var_InsideZooms->PutSplitBaseLevel(VARIANT_FALSE);
      var_InsideZooms->PutDefaultWidth(18);
      var_InsideZooms->Add("1/4/2008")->PutAllowInsideFormat(VARIANT_FALSE);
spG2antt1->EndUpdate();
```

The following C# sample shows how can I change the foreground color for a time unit:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.LevelCount = 2;
   var_Chart.FirstVisibleDate = "1/1/2008";
   var_Chart.AllowInsideZoom = true;
   var_Chart.AllowResizeInsideZoom = false;
   var_Chart.InsideZoomOnDblClick = false;
   var_Chart.DefaultInsideZoomFormat.ForeColor = 255;
   EXG2ANTTLib.InsideZooms var_InsideZooms = var_Chart.InsideZooms;
      var_InsideZooms.SplitBaseLevel = false;
      var_InsideZooms.DefaultWidth = 18;
      var_InsideZooms.Add("1/4/2008").AllowInsideFormat = false;
```

```
axG2antt1.EndUpdate();
```

The following VFP sample shows how can I change the foreground color for a time unit:

```
with thisform.G2antt1
   .BeginUpdate
   with .Chart
      .LevelCount = 2
      .FirstVisibleDate = {^2008-1-1}
      .AllowInsideZoom = .T.
      .AllowResizeInsideZoom = .F.
      .InsideZoomOnDblClick = .F.
      .DefaultInsideZoomFormat.ForeColor = 255
      with .InsideZooms
         .SplitBaseLevel = .F.
         .DefaultWidth = 18
         .Add({^2008-1-4}).AllowInsideFormat = .F.
      endwith
   endwith
   .EndUpdate
endwith
```

property InsideZoom.AllowResize as Boolean

Specifies whether the user can resize the date/time unit.

Type	Description
Boolean	A Boolean expression that specifies whether the unit may be resized at runtime.

By default, the AllowResize property is True. If the AllowResize property is True, the resizing cursor shows up when the cursor pointer hovers the unit in the base level. The Width property specifies the width in pixels of the unit. You can use the CondInsideZoom property to specify a formula to define the time units that may be resized if the AllowResizeInsideZoom property is True.

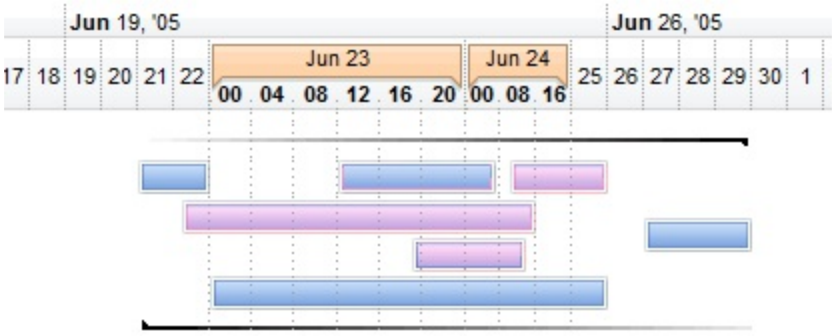
property InsideZoom.CustomFormat as InsideZoomFormat

Gets the custom format of the time unit.

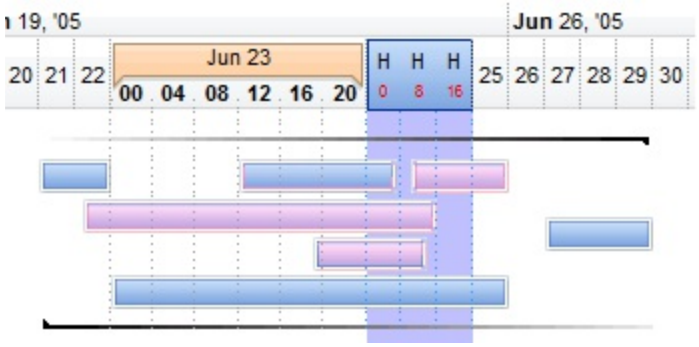
Type	Description
InsideZoomFormat	An InsideZoomFormat object that defines the custom appearance for the unit.

By default, the CustomFormat property returns nothing, as the AllowCustomFormat property is False by default. Once the AllowCustomFormat property is changed to True, it creates a new custom format object, that can be accessed later using the CustomFormat property. The new custom object copies the attributes of the DefaultInsideZoomFormat object. Use the AllowCustomFormat and CustomFormat properties to define units with different appearances, including labels, background or foreground colors, and so on. If the AllowCustomFormat property is false, the inside zoom units use the DefaultInsideZoomFormat property to define the look and feel of the inside units. If the AllowCustomFormat property, the CustomFormat property defines a custom look and feel for the inside unit. The CustomFormat property retrieves nothing, if the AllowCustomFormat property is False.

The following screen shot shows the the units using the same format (June 23 and June 24), AllowCustomFormat property is False:



The following screen shot shows the the units using the different formats (June 23 and June 24), AllowCustomFormat property is True:



property InsideZoom.EndDate as Date

Returns the date where the zoom unit ends.

Type	Description
Date	A Date-Time expression that specifies the ending date of the inside zoom unit.

By default, the EndDate is defined once the the Add method is called as being the end date of the unit as indicated by the base level in the chart. For instance, if the chart displays days, the StartDate indicate the date where the inside zoom unit start, and the EndDate where the date ends, in other words the next day of starting date. If the chart displays weeks, the start date indicates where the week beings, since the EndDate indicates the end date on the week, in other words, where the next week begins. So, the EndDate property indicates the ending date of inside zoom unit, as specified by the base level ate the adding time. If the chart's time scale is changed, the StartDate and the EndDate of the InsideZoom objects are not changed accordingly to the new base level.

property InsideZoom.StartDate as Date

Returns the date where the zoom unit begins.

Type	Description
Date	A Date-Time expression that specifies the starting date of the inside zoom unit.

By default, the StartDate property indicates the date being used when the Add method is called, or the same as DateTime parameter of the InsideZoom event. The EndDate property indicates the ending date of the unit. If the chart's time scale is changed, the StartDate and the EndDate of the InsideZoom objects are not changed accordingly to the new base level.

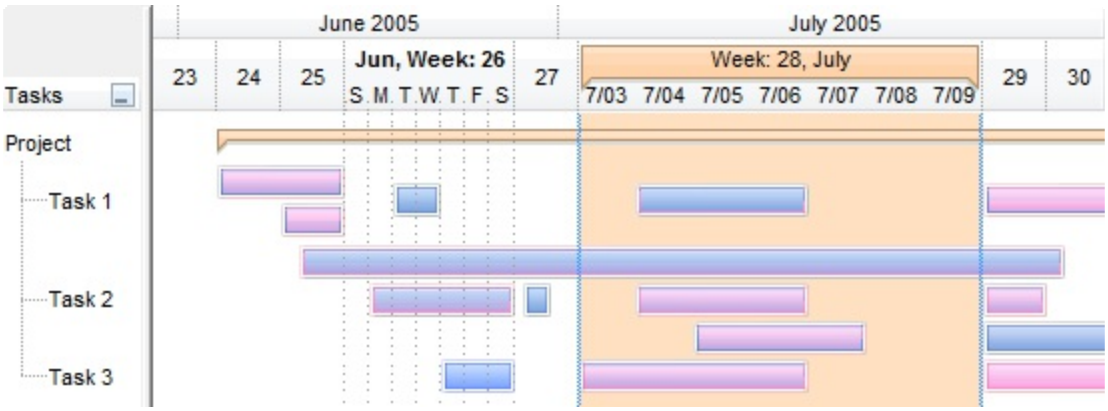
property InsideZoom.Width as Long

Specifies the width of the date/time in the chart.

Type	Description
Long	A long expression that specifies the width in pixels, of the date-time unit.

The Width property defines the width in pixels of the unit being shown in the levels area. By default, the Width property is indicated by the DefaultWidth property (by default 128). Use the Width property to specify the width of the unit in the level area. The AllowResize property specifies whether the unit is resizable, ie the resizing cursor shows up when the cursor hovers the unit. You can use Width property on 0 to hide a specified unit. Use the CondInsideZoom property to specify a formula to define the time units that may be resized if the AllowResizeInsideZoom property is True.

The following screen show displays two inside zoom units with different size (Week 26, and Week 28):



The following VB sample shows how can I change the width for a specified date time unit:

```
With G2antt1
   .BeginUpdate
   With .Chart
      .LevelCount = 2
      .FirstVisibleDate = #1/1/2008#
      .AllowInsideZoom = True
      .AllowResizeInsideZoom = False
      .InsideZoomOnDblClick = False
      With .InsideZooms
         With .Add(#1/4/2008#)
            .Width = 32
```

```
            .AllowInsideFormat = False
        End With
    End With
  End With
  .EndUpdate
End With
```

The following VB.NET sample shows how can I change the width for a specified date time unit:

```
With AxG2antt1
  .BeginUpdate
  With .Chart
    .LevelCount = 2
    .FirstVisibleDate = #1/1/2008#
    .AllowInsideZoom = True
    .AllowResizeInsideZoom = False
    .InsideZoomOnDblClick = False
    With .InsideZooms
      With .Add(#1/4/2008#)
        .Width = 32
        .AllowInsideFormat = False
      End With
    End With
  End With
  .EndUpdate
End With
```

The following C++ sample shows how can I change the width for a specified date time unit:

```
/*
  Copy and paste the following directives to your header file as
  it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

  #import <ExG2antt.dll>
  using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
```

```
>GetControlUnknown();
spG2antt1->BeginUpdate();
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutLevelCount(2);
   var_Chart->PutFirstVisibleDate("1/1/2008");
   var_Chart->PutAllowInsideZoom(VARIANT_TRUE);
   var_Chart->PutAllowResizeInsideZoom(VARIANT_FALSE);
   var_Chart->PutInsideZoomOnDblClick(VARIANT_FALSE);
   EXG2ANTTLib::IInsideZoomsPtr var_InsideZooms = var_Chart->GetInsideZooms();
      EXG2ANTTLib::IInsideZoomPtr var_InsideZoom = var_InsideZooms-
>Add("1/4/2008");
         var_InsideZoom->PutWidth(32);
         var_InsideZoom->PutAllowInsideFormat(VARIANT_FALSE);
spG2antt1->EndUpdate();
```

The following C# sample shows how can I change the width for a specified date time unit:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.LevelCount = 2;
   var_Chart.FirstVisibleDate = "1/1/2008";
   var_Chart.AllowInsideZoom = true;
   var_Chart.AllowResizeInsideZoom = false;
   var_Chart.InsideZoomOnDblClick = false;
   EXG2ANTTLib.InsideZooms var_InsideZooms = var_Chart.InsideZooms;
      EXG2ANTTLib.InsideZoom var_InsideZoom = var_InsideZooms.Add("1/4/2008");
         var_InsideZoom.Width = 32;
         var_InsideZoom.AllowInsideFormat = false;
axG2antt1.EndUpdate();
```

The following VFP sample shows how can I change the width for a specified date time unit:

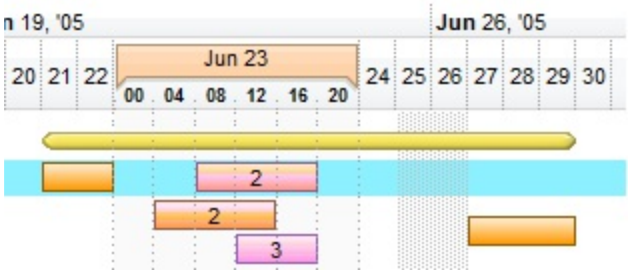
```
with thisform.G2antt1
   .BeginUpdate
   with .Chart
      .LevelCount = 2
      .FirstVisibleDate = {^2008-1-1}
      .AllowInsideZoom = .T.
```

```
    .AllowResizeInsideZoom = .F.
    .InsideZoomOnDblClick = .F.
    with .InsideZooms
      with .Add({^2008-1-4})
        .Width = 32
        .AllowInsideFormat = .F.
      endwith
    endwith
  endwith
  .EndUpdate
endwith
```

InsideZoomFormat object

The InsideZoomFormat object defines the look and feel for the inside zoom time scale units. Use the AllowInsideZoom property to specify whether the chart supports inside zoom units. The inside zoom feature allows displaying portions of the chart with different time scale units. For instance, you can display the bars on hours, while the chart still displays days.

The following screen shot shows the inside zoom units for June 23, that displays the hours, while the rest of the chart displays only days.



The InsideZoomFormat object supports the following properties and methods:

Name	Description
BackColor	Retrieves or sets a value that indicates the inside level's background color.
BackColorChart	Retrieves or sets a value that indicates the chart level's background color.
DisplayOwnerLabel	Specifies whether the owner's label is shown for a time scale unit.
DrawGridLines	Specifies whether the grid lines are shown or hidden.
DrawTickLines	Specifies whether the tick lines are shown or hidden.
ForeColor	Retrieves or sets a value that indicates the inside level's foreground color.
GridLineColor	Specifies the grid line color for the inside level.
GridLineStyle	specifies the style for the vertical gridlines when a time scale unit is being zoomed.
InsideCount	Counts the units in the inside level.
InsideLabel	Retrieves or sets a value that indicates the format of the inside level's label.
InsideUnit	Retrieves or sets a value that indicates the unit of the inside level.
OwnerLabel	Retrieves or sets a value that indicates the format of the original level's label.

PatternChart	Specifies the pattern to show on the chart.
PatternColorChart	Specifies the color of the pattern to show on the chart.

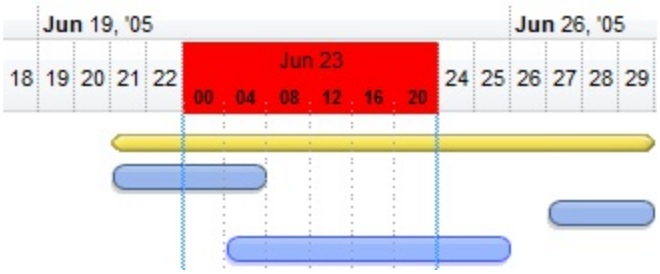
property InsideZoomFormat.BackColor as Color

Retrieves or sets a value that indicates the inside level's background color.

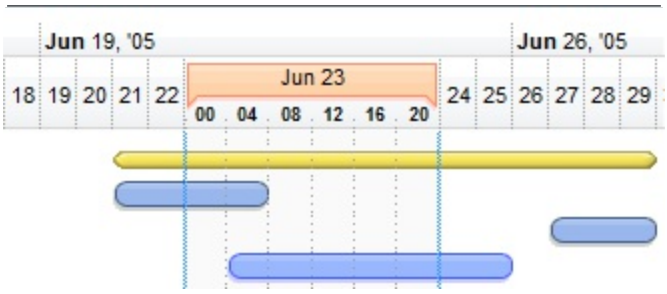
Type	Description
Color	A Color expression that specifies the background color for the time unit, being displayed in the chart's header (only if it is not 0). The last 7 bits in the high significant byte of the color indicates the identifier of the skin being used to paint the header. Use the Add method to add new skins to the control.

By default, the BackColor property is 0. The property has no effect if it is 0. The BackColor property controls the background of the time scale unit being shown in the chart's header. Use the DrawTickLines property to specify whether the tick lines splits the time units in the chart's header. The BackColorChart property specifies the background color for the time scale unit being shown in the chart area (not in the header). Use the SplitBaseLevel property specifies whether the chart's base level gets divided when inside zoom are shown. The ForeColor property specifies the unit's foreground color. The PatternChartColor property determines the color to be used for pattern.

Use the BackColor property to change the visual appearance for a time unit, in the chart's header as in the following screen shot (in red):



or when using a skin object the time-scale unit may show as follows:



The following VB sample shows how can I change the background color for a time unit:

```
With G2antt1
  .BeginUpdate
```

```
  With .Chart
    .LevelCount = 2
    .FirstVisibleDate = #1/1/2008#
    .AllowInsideZoom = True
    .AllowResizeInsideZoom = False
    .InsideZoomOnDblClick = False
    .DefaultInsideZoomFormat.BackColor = 255
    With .InsideZooms
      .SplitBaseLevel = False
      .DefaultWidth = 18
      .Add(#1/4/2008#).AllowInsideFormat = False
    End With
  End With
  .EndUpdate
End With
```

The following VB.NET sample shows how can I change the background color for a time unit:

```
With AxG2antt1
  .BeginUpdate
  With .Chart
    .LevelCount = 2
    .FirstVisibleDate = #1/1/2008#
    .AllowInsideZoom = True
    .AllowResizeInsideZoom = False
    .InsideZoomOnDblClick = False
    .DefaultInsideZoomFormat.BackColor = 255
    With .InsideZooms
      .SplitBaseLevel = False
      .DefaultWidth = 18
      .Add(#1/4/2008#).AllowInsideFormat = False
    End With
  End With
  .EndUpdate
End With
```

The following C++ sample shows how can I change the background color for a time unit:

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutLevelCount(2);
   var_Chart->PutFirstVisibleDate("1/1/2008");
   var_Chart->PutAllowInsideZoom(VARIANT_TRUE);
   var_Chart->PutAllowResizeInsideZoom(VARIANT_FALSE);
   var_Chart->PutInsideZoomOnDblClick(VARIANT_FALSE);
   var_Chart->GetDefaultInsideZoomFormat()->PutBackColor(255);
   EXG2ANTTLib::IInsideZoomsPtr var_InsideZooms = var_Chart->GetInsideZooms();
      var_InsideZooms->PutSplitBaseLevel(VARIANT_FALSE);
      var_InsideZooms->PutDefaultWidth(18);
      var_InsideZooms->Add("1/4/2008")->PutAllowInsideFormat(VARIANT_FALSE);
spG2antt1->EndUpdate();
```

The following C# sample shows how can I change the background color for a time unit:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.LevelCount = 2;
   var_Chart.FirstVisibleDate = "1/1/2008";
   var_Chart.AllowInsideZoom = true;
   var_Chart.AllowResizeInsideZoom = false;
   var_Chart.InsideZoomOnDblClick = false;
   var_Chart.DefaultInsideZoomFormat.BackColor = 255;
   EXG2ANTTLib.InsideZooms var_InsideZooms = var_Chart.InsideZooms;
      var_InsideZooms.SplitBaseLevel = false;
      var_InsideZooms.DefaultWidth = 18;
      var_InsideZooms.Add("1/4/2008").AllowInsideFormat = false;
```

```
axG2antt1.EndUpdate();
```

The following VFP sample shows how can I change the background color for a time unit:

```
with thisform.G2antt1
   .BeginUpdate
   with .Chart
      .LevelCount = 2
      .FirstVisibleDate = {^2008-1-1}
      .AllowInsideZoom = .T.
      .AllowResizeInsideZoom = .F.
      .InsideZoomOnDblClick = .F.
      .DefaultInsideZoomFormat.BackColor = 255
      with .InsideZooms
         .SplitBaseLevel = .F.
         .DefaultWidth = 18
         .Add({^2008-1-4}).AllowInsideFormat = .F.
      endwith
   endwith
   .EndUpdate
endwith
```

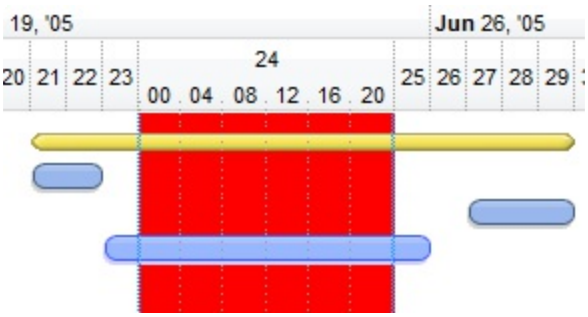
property InsideZoomFormat.BackColorChart as Color

Retrieves or sets a value that indicates the chart level's background color.

Type	Description
Color	A Color expression that specifies the background color for the time unit, being displayed in the chart area (only if it is not 0).

By default, the BackColorChart property is 0. The property has no effect if it is 0. The BackColorChart property controls the background of the time scale unit being shown in the chart. Use the DrawGridLines property to specify whether the grid lines are shown in the time unit in the chart area. The BackColor property controls the background of the time scale unit being shown in the chart's header. The PatternChart property specifies the pattern to show on the chart.

Use the BackColorChart property to change the visual appearance for a time unit, in the chart as in the following screen shot (in red):



The following VB sample shows how can I change the background color for a time unit, in the chart area:

```
With G2antt1
   .BeginUpdate
   With .Chart
      .LevelCount = 2
      .FirstVisibleDate = #1/1/2008#
      .AllowInsideZoom = True
      .AllowResizeInsideZoom = False
      .InsideZoomOnDblClick = False
      .DefaultInsideZoomFormat.BackColorChart = 255
      With .InsideZooms
         .SplitBaseLevel = False
```

```
        .DefaultWidth = 18
        .Add(#1/4/2008#).AllowInsideFormat = False
      End With
    End With
    .EndUpdate
End With
```

The following VB.NET sample shows how can I change the background color for a time unit, in the chart area:

```
With AxG2antt1
    .BeginUpdate
    With .Chart
      .LevelCount = 2
      .FirstVisibleDate = #1/1/2008#
      .AllowInsideZoom = True
      .AllowResizeInsideZoom = False
      .InsideZoomOnDblClick = False
      .DefaultInsideZoomFormat.BackColorChart = 255
      With .InsideZooms
        .SplitBaseLevel = False
        .DefaultWidth = 18
        .Add(#1/4/2008#).AllowInsideFormat = False
      End With
    End With
    .EndUpdate
End With
```

The following C++ sample shows how can I change the background color for a time unit, in the chart area:

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
```

```
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutLevelCount(2);
   var_Chart->PutFirstVisibleDate("1/1/2008");
   var_Chart->PutAllowInsideZoom(VARIANT_TRUE);
   var_Chart->PutAllowResizeInsideZoom(VARIANT_FALSE);
   var_Chart->PutInsideZoomOnDblClick(VARIANT_FALSE);
   var_Chart->GetDefaultInsideZoomFormat()->PutBackColorChart(255);
   EXG2ANTTLib::IInsideZoomsPtr var_InsideZooms = var_Chart->GetInsideZooms();
      var_InsideZooms->PutSplitBaseLevel(VARIANT_FALSE);
      var_InsideZooms->PutDefaultWidth(18);
      var_InsideZooms->Add("1/4/2008")->PutAllowInsideFormat(VARIANT_FALSE);
spG2antt1->EndUpdate();
```

The following C# sample shows how can I change the background color for a time unit, in the chart area:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.LevelCount = 2;
   var_Chart.FirstVisibleDate = "1/1/2008";
   var_Chart.AllowInsideZoom = true;
   var_Chart.AllowResizeInsideZoom = false;
   var_Chart.InsideZoomOnDblClick = false;
   var_Chart.DefaultInsideZoomFormat.BackColorChart = 255;
   EXG2ANTTLib.InsideZooms var_InsideZooms = var_Chart.InsideZooms;
      var_InsideZooms.SplitBaseLevel = false;
      var_InsideZooms.DefaultWidth = 18;
      var_InsideZooms.Add("1/4/2008").AllowInsideFormat = false;
axG2antt1.EndUpdate();
```

The following VFP sample shows how can I change the background color for a time unit, in the chart area:

```
with thisform.G2antt1
   .BeginUpdate
```

```
   with .Chart
      .LevelCount = 2
      .FirstVisibleDate = {^2008-1-1}
      .AllowInsideZoom = .T.
      .AllowResizeInsideZoom = .F.
      .InsideZoomOnDblClick = .F.
      .DefaultInsideZoomFormat.BackColorChart = 255
      with .InsideZooms
         .SplitBaseLevel = .F.
         .DefaultWidth = 18
         .Add({^2008-1-4}).AllowInsideFormat = .F.
      endwith
   endwith
   .EndUpdate
endwith
```

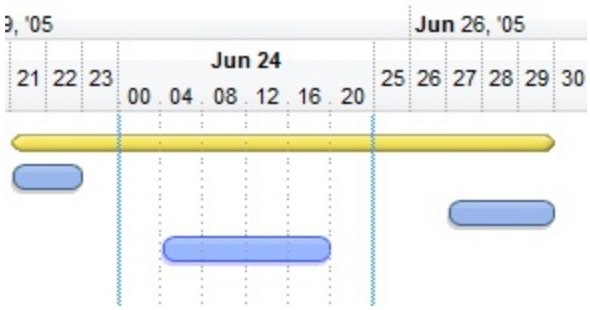
property InsideZoomFormat.DisplayOwnerLabel as Boolean

Specifies whether the owner's label is shown for a time scale unit.

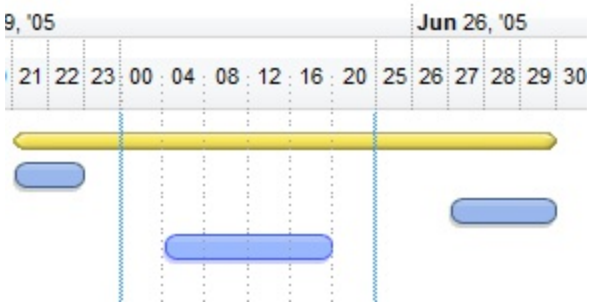
Type	Description
Boolean	A boolean expression that specifies whether the owner level is being displayed.

By default, the DisplayOwneLevel property is True. The OwnerLabel property indicates the label being displayed in the time unit, once it gets magnified. Use the AllowInsideFormat property to avoid displaying the inside units, and so the inside labels. The InsideLabel property defines the label for inside units being displayed. Use the SplitBaseLevel property specifies whether the chart's base level gets divided when inside zoom are shown.

The following screen shot shows the unit with DisplayOwnerLabel property is True (The Jun 24 is getting displayed on the first line, while the second line displays hours):



while the next screen shot shows the unit with DisplayOwnerLabel property is False (The Jun 24 is not displayed, and actually only hours are displayed):



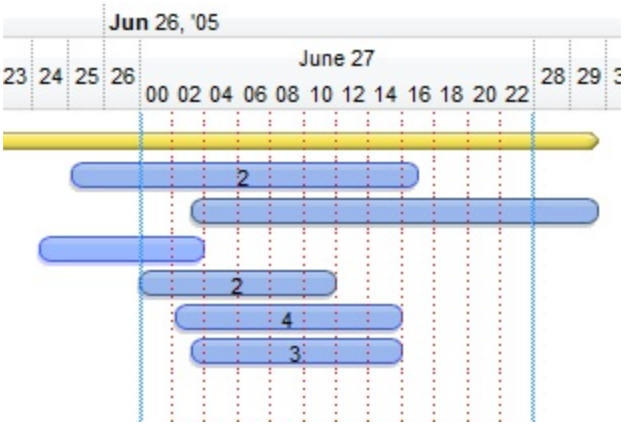
property InsideZoomFormat.DrawGridLines as Boolean

Specifies whether the grid lines are shown or hidden.

Type	Description
Boolean	A boolean expression that specifies whether the inside zoom unit displays grid lines. The grid lines are shown only in the chart area.

By default, the DrawGridLines property is True. The grid lines are shown in the chart area only, if the DrawGridLines property is True and the DrawGridLines property of the Chart object is not exNoLines or exHLines. Use the DrawTickLines property to specify whether the tick lines are shown in the unit's header. The GridLineStyle property specifies the style to shown the vertical gridlines inside the time-scale unit. The GridLineColor property specifies the color for grid lines being shown in the chart area for specified unit. The BackColorChart property specifies the unit's background color being shown in the chart (not in the header).

The following screen shot shows the grid lines in red:



property InsideZoomFormat.DrawTickLines as Boolean

Specifies whether the tick lines are shown or hidden.

Type	Description
Boolean	A boolean expression that specifies whether the time unit displays tick lines that splits the inside zoom units. The tick lines are shown only in the chart's header.

By default, the DrawTickLines property is True. The tick lines are shown only in the header area, not in the chat area. Use the DrawGridLines property to specify whether the grid lines are shown in the time unit in the chart area. The BackColor property controls the background of the time scale unit being shown in the chart's header.

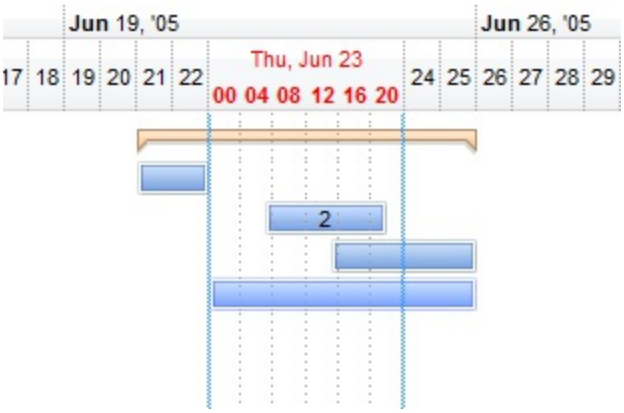
property InsideZoomFormat.ForeColor as Color

Retrieves or sets a value that indicates the inside level's foreground color.

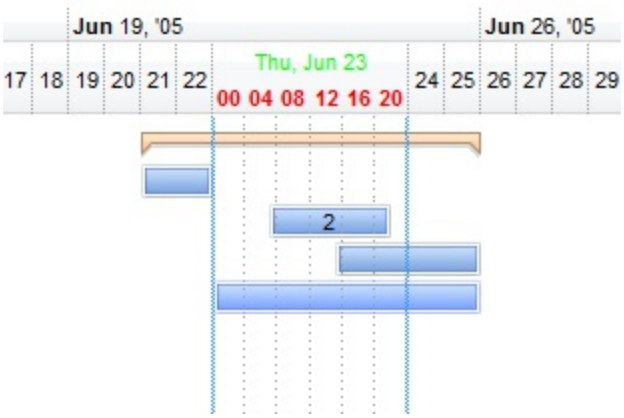
Type	Description
Color	A Color expression that specifies the unit's foreground color, if it is not 0.

By default, the ForeColor property is 0. If not zero, the ForeColor property indicates the unit's foreground color. You are still able to define colors for inside or owner levels using the <fgcolor> HTML tag in OwnerLabel or InsideLabel properties. The BackColor property controls the background of the time scale unit being shown in the chart's header. Use the SplitBaseLevel property specifies whether the chart's base level gets divided when inside zoom are shown.

The following screen shot shows a unit using a different foreground color:



The following sample shows the inside and owner levels using different colors (<fgcolor> HTML tag):



The following VB sample shows how can I change the foreground color for a time unit:

```
With G2antt1
  .BeginUpdate
```

```
  With .Chart
    .LevelCount = 2
    .FirstVisibleDate = #1/1/2008#
    .AllowInsideZoom = True
    .AllowResizeInsideZoom = False
    .InsideZoomOnDblClick = False
    .DefaultInsideZoomFormat.ForeColor = 255
    With .InsideZooms
      .SplitBaseLevel = False
      .DefaultWidth = 18
      .Add(#1/4/2008#).AllowInsideFormat = False
    End With
  End With
  .EndUpdate
End With
```

The following VB.NET sample shows how can I change the foreground color for a time unit:

```
With AxG2antt1
  .BeginUpdate
  With .Chart
    .LevelCount = 2
    .FirstVisibleDate = #1/1/2008#
    .AllowInsideZoom = True
    .AllowResizeInsideZoom = False
    .InsideZoomOnDblClick = False
    .DefaultInsideZoomFormat.ForeColor = 255
    With .InsideZooms
      .SplitBaseLevel = False
      .DefaultWidth = 18
      .Add(#1/4/2008#).AllowInsideFormat = False
    End With
  End With
  .EndUpdate
End With
```

The following C++ sample shows how can I change the foreground color for a time unit:

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutLevelCount(2);
   var_Chart->PutFirstVisibleDate("1/1/2008");
   var_Chart->PutAllowInsideZoom(VARIANT_TRUE);
   var_Chart->PutAllowResizeInsideZoom(VARIANT_FALSE);
   var_Chart->PutInsideZoomOnDblClick(VARIANT_FALSE);
   var_Chart->GetDefaultInsideZoomFormat()->PutForeColor(255);
   EXG2ANTTLib::IInsideZoomsPtr var_InsideZooms = var_Chart->GetInsideZooms();
      var_InsideZooms->PutSplitBaseLevel(VARIANT_FALSE);
      var_InsideZooms->PutDefaultWidth(18);
      var_InsideZooms->Add("1/4/2008")->PutAllowInsideFormat(VARIANT_FALSE);
spG2antt1->EndUpdate();
```

The following C# sample shows how can I change the foreground color for a time unit:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.LevelCount = 2;
   var_Chart.FirstVisibleDate = "1/1/2008";
   var_Chart.AllowInsideZoom = true;
   var_Chart.AllowResizeInsideZoom = false;
   var_Chart.InsideZoomOnDblClick = false;
   var_Chart.DefaultInsideZoomFormat.ForeColor = 255;
   EXG2ANTTLib.InsideZooms var_InsideZooms = var_Chart.InsideZooms;
      var_InsideZooms.SplitBaseLevel = false;
      var_InsideZooms.DefaultWidth = 18;
      var_InsideZooms.Add("1/4/2008").AllowInsideFormat = false;
```

```
axG2antt1.EndUpdate();
```

The following VFP sample shows how can I change the foreground color for a time unit:

```
with thisform.G2antt1
   .BeginUpdate
   with .Chart
      .LevelCount = 2
      .FirstVisibleDate = {^2008-1-1}
      .AllowInsideZoom = .T.
      .AllowResizeInsideZoom = .F.
      .InsideZoomOnDblClick = .F.
      .DefaultInsideZoomFormat.ForeColor = 255
      with .InsideZooms
         .SplitBaseLevel = .F.
         .DefaultWidth = 18
         .Add({^2008-1-4}).AllowInsideFormat = .F.
      endwith
   endwith
   .EndUpdate
endwith
```

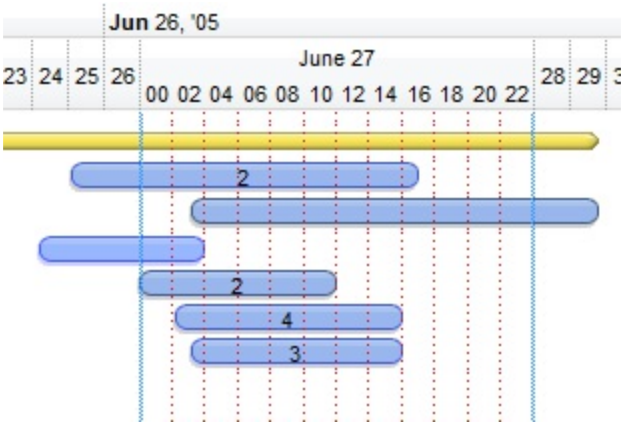
property InsideZoomFormat.GridLineColor as Color

Specifies the grid line color for the inside level.

Type	Description
Color	A Color expression that defines the unit's grid lines color.

Use the DrawGridLines property to show or hide the unit's grid lines. The grid lines are shown in the chart area only, if the DrawGridLines property is True and the DrawGridLines property of the Chart object is not exNoLines or exHLines. The GridLineStyle property specifies the style to shown the vertical gridlines inside the time-scale unit. The BackColorChart property specifies the unit's background color being shown in the chart (not in the header).

The following screen shot shows the grid lines in red:



property InsideZoomFormat.GridLineStyle as GridLinesStyleEnum

specifies the style for the vertical gridlines when a time scale unit is being zoomed.

Type	Description
GridLinesStyleEnum	A GridLinesStyleEnum expression that specifies the style to show the chart's vertical gridlines.

By default, the GridLineStyle property is exGridLinesDot4. The GridLineStyle property indicates the style of vertical gridlines being shown in the inside time-unit. The GridLineStyle property has effect only DrawGridLines property is True. The GridLineColor property specifies the color for grid lines being shown in the chart area for specified unit. The BackColorChart property specifies the unit's background color being shown in the chart (not in the header). The BackColor property controls the background of the time scale unit being shown in the chart's header.

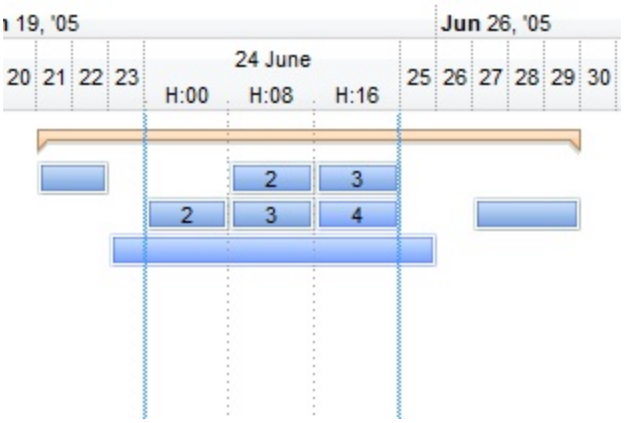
property InsideZoomFormat.InsideCount as Long

Counts the units in the inside level.

Type	Description
Long	A long expression that specifies the number on inside units being displayed at one time.

By default, the InsideCount property is 1. The InsideUnit property specifies the time scale unit being used to paint the inside zoom units. The InsideLabel property defines the label being displayed in the unit's header. For instance, if the InsideUnit is exHour, and InsideCount property is 1, the inside level displays if possible, the level from hour to hour. If the InsideCount property 4, the level displays the level from 4 hours in 4 hours. The InsideLabel and InsideUnit properties define also the resizing unit while the user creates, moves or resizes a bar. Use the ResizeUnitScale and ResizeUnitCount properties to specify the resizing unit in the chart.

The following screen shot shows the hours being displayed from 8 to 8 hours:



The following VB sample shows how can I zoom or magnify the selected date to display the hours, from 8 to 8:

```
With G2antt1
   .BeginUpdate
   With .Chart
      .PaneWidth(0) = 0
      .LevelCount = 2
      .FirstVisibleDate = #1/1/2008#
      .AllowInsideZoom = True
      With .DefaultInsideZoomFormat
         .InsideLabel = "H: <b><%hh%></b>"
         .InsideUnit = exHour
```

```
        .InsideCount = 8
      End With
      With .InsideZooms
        .Add #1/4/2008#
      End With
    End With
    .EndUpdate
End With
```

The following VB.NET sample shows how can I zoom or magnify the selected date to display the hours, from 8 to 8:

```
With AxG2antt1
    .BeginUpdate
    With .Chart
      .PaneWidth(0) = 0
      .LevelCount = 2
      .FirstVisibleDate = #1/1/2008#
      .AllowInsideZoom = True
      With .DefaultInsideZoomFormat
        .InsideLabel = "H: <b><%hh%></b>"
        .InsideUnit = EXG2ANTTLib.UnitEnum.exHour
        .InsideCount = 8
      End With
      With .InsideZooms
        .Add #1/4/2008#
      End With
    End With
    .EndUpdate
End With
```

The following C++ sample shows how can I zoom or magnify the selected date to display the hours, from 8 to 8:

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'
```

```
   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutPaneWidth(0,0);
   var_Chart->PutLevelCount(2);
   var_Chart->PutFirstVisibleDate("1/1/2008");
   var_Chart->PutAllowInsideZoom(VARIANT_TRUE);
   EXG2ANTTLib::IInsideZoomFormatPtr var_InsideZoomFormat = var_Chart-
>GetDefaultInsideZoomFormat();
      var_InsideZoomFormat->PutInsideLabel(L"H: <b><%hh%></b>");
      var_InsideZoomFormat->PutInsideUnit(EXG2ANTTLib::exHour);
      var_InsideZoomFormat->PutInsideCount(8);
   EXG2ANTTLib::IInsideZoomsPtr var_InsideZooms = var_Chart->GetInsideZooms();
      var_InsideZooms->Add("1/4/2008");
spG2antt1->EndUpdate();
```

The following C# sample shows how can I zoom or magnify the selected date to display the hours, from 8 to 8:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.set_PaneWidth(0 != 0,0);
   var_Chart.LevelCount = 2;
   var_Chart.FirstVisibleDate = "1/1/2008";
   var_Chart.AllowInsideZoom = true;
   EXG2ANTTLib.InsideZoomFormat var_InsideZoomFormat =
var_Chart.DefaultInsideZoomFormat;
      var_InsideZoomFormat.InsideLabel = "H: <b><%hh%></b>";
      var_InsideZoomFormat.InsideUnit = EXG2ANTTLib.UnitEnum.exHour;
      var_InsideZoomFormat.InsideCount = 8;
   EXG2ANTTLib.InsideZooms var_InsideZooms = var_Chart.InsideZooms;
      var_InsideZooms.Add("1/4/2008");
axG2antt1.EndUpdate();
```

The following VFP sample shows how can I zoom or magnify the selected date to display the hours, from 8 to 8:

```
with thisform.G2antt1
   .BeginUpdate
   with .Chart
      .PaneWidth(0) = 0
      .LevelCount = 2
      .FirstVisibleDate = {^2008-1-1}
      .AllowInsideZoom = .T.
      with .DefaultInsideZoomFormat
         .InsideLabel = "H: <b><%hh%></b>"
         .InsideUnit = 65536
         .InsideCount = 8
      endwith
      with .InsideZooms
         .Add({^2008-1-4})
      endwith
   endwith
   .EndUpdate
endwith
```

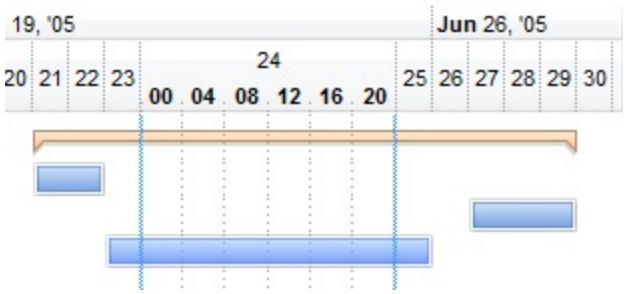
property InsideZoomFormat.InsideLabel as String

Retrieves or sets a value that indicates the format of the inside level's label.

Type	Description
String	A String expression that specifies the label being displayed for inside zoom units. It supports built-in HTML format as well.

By default, the InsideLabel property is "<%hh%>", that shows the hours in 2 digits. The AllowInsideFormat property indicates whether the inside label is being displayed or not. The InsideUnit property specifies the time scale unit being used to paint the inside zoom units. The InsideCount property specifies the number on inside units being displayed at one time. Even if the InsideLabel property is empty, the inside level may display the tick lines, or the grid lines.

The following screen shows the inside units (hours) in bold (*InsideLabel property is " <%hh%>"*):



The InsideLabel property supports the following:

- <%**d**%> - Day of the month in one or two numeric digits, as needed (1 to 31).
- <%**dd**%> - Day of the month in two numeric digits (01 to 31).
- <%**d1**%> - First letter of the weekday (S to S). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**d2**%> - First two letters of the weekday (Su to Sa). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**d3**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**ddd**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays property to specify the name of the days in the week). You can use the

<%**loc_ddd**%> that indicates the day of week as a three-letter abbreviation using the current user regional and language settings.

- <%**loc_ddd**%> - Indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%**dddd**%> - Full name of the weekday (Sunday to Saturday). (Use the WeekDays property to specify the name of the days in the week). You can use the <%**loc_dddd**%> that indicates day of week as its full name using the current user regional and language settings.
- <%**loc_dddd**%> - Indicates day of week as its full name using the current user regional and language settings.
- <%**i**%> - Displays the number instead the date. For instance, you can display numbers as 1000, 1001, 1002, 1003, instead dates. (the valid range is from -647,434 to 2,958,465)
- <%**w**%> - Day of the week (1 to 7).
- <%**ww**%> - Week of the year (1 to 53).
- <%**m**%> - Month of the year in one or two numeric digits, as needed (1 to 12).
- <%**mr**%> - Month of the year in Roman numerals, as needed (I to XII).
- <%**mm**%> - Month of the year in two numeric digits (01 to 12).
- <%**m1**%> - First letter of the month (J to D). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**m2**%> - First two letters of the month (Ja to De). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**m3**%> - First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**mmm**%> - First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmm**%> that indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%**loc_mmm**%> - Indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%**mmmm**%> - Full name of the month (January to December). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmmm**%> that indicates month as its full name using the current user regional and language settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user regional and language settings.
- <%**q**%> - Date displayed as the quarter of the year (1 to 4).

- <%**y**%> - Number of the day of the year (1 to 366).
- <%**yy**%> - Last two digits of the year (01 to 99).
- <%**yyyy**%> - Full year (0100 to 9999).
- <%**hy**%> - Date displayed as the half of the year (1 to 2).
- <%**loc_gg**%> - Indicates period/era using the current user regional and language settings.
- <%**loc_sdate**%> - Indicates the date in the short format using the current user regional and language settings.
- <%**loc_ldate**%> - Indicates the date in the long format using the current user regional and language settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user regional and language settings (/).
- <%**h**%> - Hour in one or two digits, as needed (0 to 23).
- <%**hh**%> - Hour in two digits (00 to 23).
- <%**h12**%> - Hour in 12-hour time format, in one or two digits - [0(12),11]
- <%**hh12**%> - hour in 12-hour time format, in two digits - [00(12),11]
- <%**n**%> - Minute in one or two digits, as needed (0 to 59).
- <%**nn**%> - Minute in two digits (00 to 59).
- <%**s**%> - Second in one or two digits, as needed (0 to 59).
- <%**ss**%> - Second in two digits (00 to 59).
- <%**AM/PM**%> - Twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate. (Use the AMPM property to specify the name of the AM and PM indicators). You can use the <%**loc_AM/PM**%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%**loc_A/P**%> that indicates the one character time marker such as A or P using the current user regional and language settings
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user regional and language settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user regional and language settings.
- <%**loc_time**%> - Indicates the time using the current user regional and language settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%**loc_tsep**%> - indicates the time separator using the current user regional and language settings (:)
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy"

pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

The following tags are displayed based on the user's Regional and Language Options:

- <%**loc_sdate**%> - Indicates the date in the short format using the current user settings.
- <%**loc_ldate**%> - Indicates the date in the long format using the current user settings.
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**loc_ddd**%> - Indicates day of week as a three-letters abbreviation using the current user settings.
- <%**loc_dddd**%> - Indicates day of week as its full name using the current user settings.
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**loc_mmm**%> - Indicates month as a three-letters abbreviation using the current user settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user settings.
- <%**loc_gg**%> - Indicates period/era using the current user settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user settings.
- <%**loc_time**%> - Indicates the time using the current user settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user settings.
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user settings.
- <%**loc_tsep**%> - Indicates the time separator using the current user settings
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.

- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

The InsideLabel property supports:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.

  The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using <a ;exp=> or <a ;e64=> anchor tags. The exp/e64 field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption.

  - exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
  - e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

  Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** displays portions of text with a different font and/or

different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.

- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset

parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{subscript}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}

- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

shadow

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

outline anti-aliasing

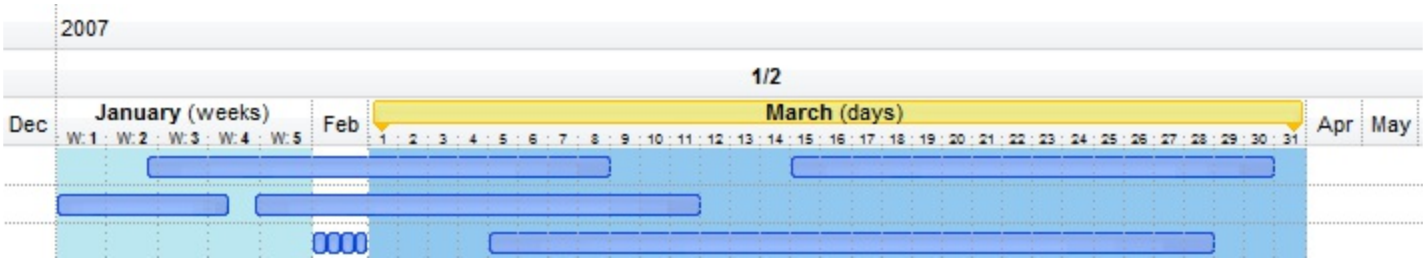
property InsideZoomFormat.InsideUnit as UnitEnum

Retrieves or sets a value that indicates the unit of the inside level.

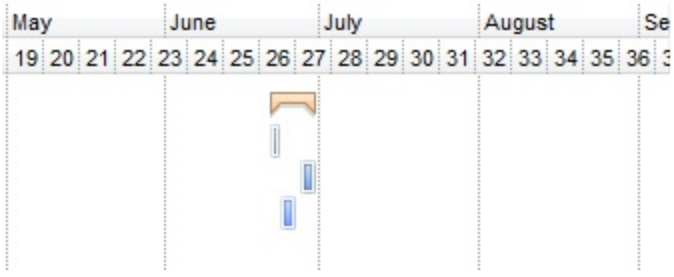
Type	Description
UnitEnum	A UnitEnum expression that specifies the time scale unit being used to display the inside units.

By default, the InsideUnit property is exHour. Use the InsideUnit property to change the inside zoom scale once the user magnify a time in the chart's base level. The InsideCount property specifies the number on inside units being displayed at one time. The InsideLabel property defines the label being displayed in the unit's header. For instance, if the InsideUnit is exHour, and InsideCount property is 1, the inside level displays if possible, the level from hour to hour. If the InsideCount property 4, the level displays the level from 4 hours in 4 hours. The InsideLabel and InsideUnit properties define also the resizing unit while the user creates, moves or resizes a bar. Use the ResizeUnitScale and ResizeUnitCount properties to specify the resizing unit in the chart.

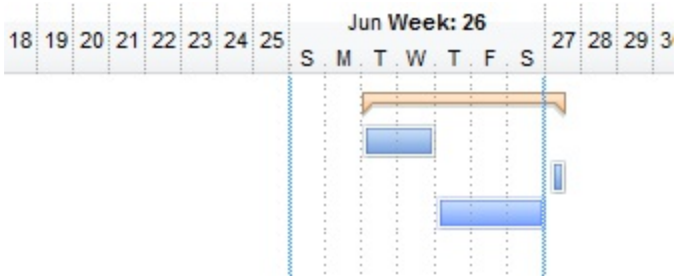
The following screen shot splits the time scale chart, and displays different sections using different time scale units (the chart displays months, the January displays weeks, while the March displays days):



The following screen shot shows the chart displays week numbers:



The following screen shot shows the week days once a week gets expanded/magnified (the chart displays weeks):



The following VB sample shows how can I change the scale unit when doing inside zoom (the chart displays weeks, and we want week days):

```
With G2antt1
   .BeginUpdate
   With .Chart
      .ShowNonworkingDates = False
      .PaneWidth(0) = 0
      .LevelCount = 2
      With .Level(0)
         .Label = "<%mmmm%>"
         .Unit = exMonth
      End With
      With .Level(1)
         .Label = "<%ww%>"
         .Unit = exWeek
      End With
      .FirstVisibleDate = #1/1/2008#
      .AllowInsideZoom = True
      With .DefaultInsideZoomFormat
         .OwnerLabel = "<font ;7><%mmm%> Week: <%ww%>"
         .InsideLabel = "<font ;7><b><%d1%></b>"
         .InsideUnit = exDay
      End With
      With .InsideZooms
         .SplitBaseLevel = False
         .Add #2/3/2008#
      End With
   End With
   .EndUpdate
End With
```

The following VB.NET sample shows how can I change the scale unit when doing inside zoom (the chart displays weeks, and we want week days):

```
With AxG2antt1
   .BeginUpdate
   With .Chart
      .ShowNonworkingDates = False
      .PaneWidth(0) = 0
      .LevelCount = 2
      With .Level(0)
         .Label = "<%mmmm%>"
         .Unit = EXG2ANTTLib.UnitEnum.exMonth
      End With
      With .Level(1)
         .Label = "<%ww%>"
         .Unit = EXG2ANTTLib.UnitEnum.exWeek
      End With
      .FirstVisibleDate = #1/1/2008#
      .AllowInsideZoom = True
      With .DefaultInsideZoomFormat
         .OwnerLabel = "<font ;7><%mmm%> Week: <%ww%>"
         .InsideLabel = "<font ;7><b><%d1%></b>"
         .InsideUnit = EXG2ANTTLib.UnitEnum.exDay
      End With
      With .InsideZooms
         .SplitBaseLevel = False
         .Add #2/3/2008#
      End With
   End With
   .EndUpdate
End With
```

The following C++ sample shows how can I change the scale unit when doing inside zoom (the chart displays weeks, and we want week days):

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'
```

```
   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutShowNonworkingDates(VARIANT_FALSE);
   var_Chart->PutPaneWidth(0,0);
   var_Chart->PutLevelCount(2);
   EXG2ANTTLib::ILevelPtr var_Level = var_Chart->GetLevel(0);
      var_Level->PutLabel("<%mmmm%>");
      var_Level->PutUnit(EXG2ANTTLib::exMonth);
   EXG2ANTTLib::ILevelPtr var_Level1 = var_Chart->GetLevel(1);
      var_Level1->PutLabel("<%ww%>");
      var_Level1->PutUnit(EXG2ANTTLib::exWeek);
   var_Chart->PutFirstVisibleDate("1/1/2008");
   var_Chart->PutAllowInsideZoom(VARIANT_TRUE);
   EXG2ANTTLib::IInsideZoomFormatPtr var_InsideZoomFormat = var_Chart-
>GetDefaultInsideZoomFormat();
      var_InsideZoomFormat->PutOwnerLabel(L"<font ;7><%mmm%> Week: <%ww%>");
      var_InsideZoomFormat->PutInsideLabel(L"<font ;7><b><%d1%></b>");
      var_InsideZoomFormat->PutInsideUnit(EXG2ANTTLib::exDay);
   EXG2ANTTLib::IInsideZoomsPtr var_InsideZooms = var_Chart->GetInsideZooms();
      var_InsideZooms->PutSplitBaseLevel(VARIANT_FALSE);
      var_InsideZooms->Add("2/3/2008");
spG2antt1->EndUpdate();
```

The following C# sample shows how can I change the scale unit when doing inside zoom (the chart displays weeks, and we want week days):

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.ShowNonworkingDates = false;
   var_Chart.set_PaneWidth(0 != 0,0);
   var_Chart.LevelCount = 2;
   EXG2ANTTLib.Level var_Level = var_Chart.get_Level(0);
```

```
    var_Level.Label = "<%mmmm%>";
    var_Level.Unit = EXG2ANTTLib.UnitEnum.exMonth;
  EXG2ANTTLib.Level var_Level1 = var_Chart.get_Level(1);
    var_Level1.Label = "<%ww%>";
    var_Level1.Unit = EXG2ANTTLib.UnitEnum.exWeek;
  var_Chart.FirstVisibleDate = "1/1/2008";
  var_Chart.AllowInsideZoom = true;
  EXG2ANTTLib.InsideZoomFormat var_InsideZoomFormat =
var_Chart.DefaultInsideZoomFormat;
    var_InsideZoomFormat.OwnerLabel = "<font ;7><%mmm%> Week: <%ww%>";
    var_InsideZoomFormat.InsideLabel = "<font ;7><b><%d1%></b>";
    var_InsideZoomFormat.InsideUnit = EXG2ANTTLib.UnitEnum.exDay;
  EXG2ANTTLib.InsideZooms var_InsideZooms = var_Chart.InsideZooms;
    var_InsideZooms.SplitBaseLevel = false;
    var_InsideZooms.Add("2/3/2008");
axG2antt1.EndUpdate();
```

The following VFP sample shows how can I change the scale unit when doing inside zoom (the chart displays weeks, and we want week days):

```
with thisform.G2antt1
  .BeginUpdate
  with .Chart
    .ShowNonworkingDates = .F.
    .PaneWidth(0) = 0
    .LevelCount = 2
    with .Level(0)
      .Label = "<%mmmm%>"
      .Unit = 16
    endwith
    with .Level(1)
      .Label = "<%ww%>"
      .Unit = 256
    endwith
    .FirstVisibleDate = {^2008-1-1}
    .AllowInsideZoom = .T.
    with .DefaultInsideZoomFormat
      .OwnerLabel = "<font ;7><%mmm%> Week: <%ww%>"
```

```
        .InsideLabel = "<font ;7><b><%d1%></b>"
        .InsideUnit = 4096
      endwith
      with .InsideZooms
        .SplitBaseLevel = .F.
        .Add({^2008-2-3})
      endwith
    endwith
    .EndUpdate
endwith
```

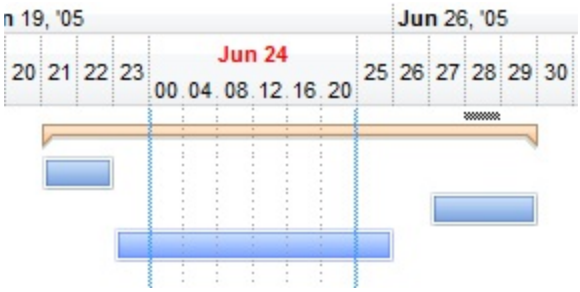
property InsideZoomFormat.OwnerLabel as String

Retrieves or sets a value that indicates the format of the original level's label.

Type	Description
String	A String expression that specifies the label being displayed by the original level. It supports built-in HTML format as well.

By default, the OwnerLabel property is "", ie the Label property is used instead, so the original format is used to paint the owner of the inside zoom level. The OwnerLabel property indicates the label being displayed in the time unit, once it gets magnified. The DisplayOwnerLabel property specifies whether the owner's label is shown for a time scale unit. The InsideLabel property defines the label for inside units being displayed. Use the SplitBaseLevel property specifies whether the chart's base level gets divided when inside zoom are shown.

The following sample shows the owner level in bold and red color: (*OwnerLevel property is "<fgcolor=FF0000><%mmm%> <%d%></fgcolor>"*)



The OwnerLabel property supports the following:

- <%**d**%> - Day of the month in one or two numeric digits, as needed (1 to 31).
- <%**dd**%> - Day of the month in two numeric digits (01 to 31).
- <%**d1**%> - First letter of the weekday (S to S). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**d2**%> - First two letters of the weekday (Su to Sa). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**d3**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**ddd**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays

property to specify the name of the days in the week). You can use the <%**loc_ddd**%> that indicates the day of week as a three-letter abbreviation using the current user regional and language settings.

- <%**loc_ddd**%> - Indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%**dddd**%> - Full name of the weekday (Sunday to Saturday). (Use the WeekDays property to specify the name of the days in the week). You can use the <%**loc_dddd**%> that indicates day of week as its full name using the current user regional and language settings.
- <%**loc_dddd**%> - Indicates day of week as its full name using the current user regional and language settings.
- <%**i**%> - Displays the number instead the date. For instance, you can display numbers as 1000, 1001, 1002, 1003, instead dates. (the valid range is from -647,434 to 2,958,465)
- <%**w**%> - Day of the week (1 to 7).
- <%**ww**%> - Week of the year (1 to 53).
- <%**m**%> - Month of the year in one or two numeric digits, as needed (1 to 12).
- <%**mr**%> - Month of the year in Roman numerals, as needed (I to XII).
- <%**mm**%> - Month of the year in two numeric digits (01 to 12).
- <%**m1**%> - First letter of the month (J to D). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**m2**%> - First two letters of the month (Ja to De). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**m3**%> - First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**mmm**%> - First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmm**%> that indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%**loc_mmm**%> - Indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%**mmmm**%> - Full name of the month (January to December). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmmm**%> that indicates month as its full name using the current user regional and language settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user regional and language settings.

- <%**q**%> - Date displayed as the quarter of the year (1 to 4).
- <%**y**%> - Number of the day of the year (1 to 366).
- <%**yy**%> - Last two digits of the year (01 to 99).
- <%**yyyy**%> - Full year (0100 to 9999).
- <%**hy**%> - Date displayed as the half of the year (1 to 2).
- <%**loc_gg**%> - Indicates period/era using the current user regional and language settings.
- <%**loc_sdate**%> - Indicates the date in the short format using the current user regional and language settings.
- <%**loc_ldate**%> - Indicates the date in the long format using the current user regional and language settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user regional and language settings (/).
- <%**h**%> - Hour in one or two digits, as needed (0 to 23).
- <%**hh**%> - Hour in two digits (00 to 23).
- <%**h12**%> - Hour in 12-hour time format, in one or two digits - [0(12),11]
- <%**hh12**%> - hour in 12-hour time format, in two digits - [00(12),11]
- <%**n**%> - Minute in one or two digits, as needed (0 to 59).
- <%**nn**%> - Minute in two digits (00 to 59).
- <%**s**%> - Second in one or two digits, as needed (0 to 59).
- <%**ss**%> - Second in two digits (00 to 59).
- <%**AM/PM**%> - Twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate. (Use the AMPM property to specify the name of the AM and PM indicators). You can use the <%**loc_AM/PM**%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%**loc_A/P**%> that indicates the one character time marker such as A or P using the current user regional and language settings
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user regional and language settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user regional and language settings.
- <%**loc_time**%> - Indicates the time using the current user regional and language settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%**loc_tsep**%> - indicates the time separator using the current user regional and language settings (:)
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the

calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

The following tags are displayed based on the user's Regional and Language Options:

- <%**loc_sdate**%> - Indicates the date in the short format using the current user settings.
- <%**loc_ldate**%> - Indicates the date in the long format using the current user settings.
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**loc_ddd**%> - Indicates day of week as a three-letters abbreviation using the current user settings.
- <%**loc_dddd**%> - Indicates day of week as its full name using the current user settings.
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**loc_mmm**%> - Indicates month as a three-letters abbreviation using the current user settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user settings.
- <%**loc_gg**%> - Indicates period/era using the current user settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user settings.
- <%**loc_time**%> - Indicates the time using the current user settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user settings.
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user settings.
- <%**loc_tsep**%> - Indicates the time separator using the current user settings
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional

settings. A leading zero is added for single-digit years.

- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

The OwnerLabel property supports:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.

  The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using <a ;exp=> or <a ;e64=> anchor tags. The exp/e64 field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption.

  - exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
  - e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

  Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold

- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with $^{\text{subscript}}$
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

shadow

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

outline anti-aliasing

The following VB sample shows how can I change the label for a specified unit:

```
With G2antt1
   .BeginUpdate
   With .Chart
      .PaneWidth(0) = 0
      .LevelCount = 2
      .FirstVisibleDate = #1/1/2008#
      .AllowInsideZoom = True
      .AllowResizeInsideZoom = False
      .InsideZoomOnDblClick = False
      .DefaultInsideZoomFormat.OwnerLabel = "<b><%d%></b> <%d2%>"
      With .InsideZooms
         .SplitBaseLevel = False
         .DefaultWidth = 32
         .Add(#1/4/2008#).AllowInsideFormat = False
      End With
   End With
   .EndUpdate
End With
```

The following VB.NET sample shows how can I change the label for a specified unit:

```
With AxG2antt1
   .BeginUpdate
   With .Chart
      .PaneWidth(0) = 0
      .LevelCount = 2
      .FirstVisibleDate = #1/1/2008#
      .AllowInsideZoom = True
      .AllowResizeInsideZoom = False
      .InsideZoomOnDblClick = False
      .DefaultInsideZoomFormat.OwnerLabel = "<b><%d%></b> <%d2%>"
      With .InsideZooms
         .SplitBaseLevel = False
         .DefaultWidth = 32
         .Add(#1/4/2008#).AllowInsideFormat = False
      End With
```

```
    End With
    .EndUpdate
End With
```

The following C++ sample shows how can I change the label for a specified unit:

```
/*
    Copy and paste the following directives to your header file as
    it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

    #import <ExG2antt.dll>
    using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
    var_Chart->PutPaneWidth(0,0);
    var_Chart->PutLevelCount(2);
    var_Chart->PutFirstVisibleDate("1/1/2008");
    var_Chart->PutAllowInsideZoom(VARIANT_TRUE);
    var_Chart->PutAllowResizeInsideZoom(VARIANT_FALSE);
    var_Chart->PutInsideZoomOnDblClick(VARIANT_FALSE);
    var_Chart->GetDefaultInsideZoomFormat()->PutOwnerLabel(L"<b><%d%></b>
<%d2%>");
    EXG2ANTTLib::IInsideZoomsPtr var_InsideZooms = var_Chart->GetInsideZooms();
        var_InsideZooms->PutSplitBaseLevel(VARIANT_FALSE);
        var_InsideZooms->PutDefaultWidth(32);
        var_InsideZooms->Add("1/4/2008")->PutAllowInsideFormat(VARIANT_FALSE);
spG2antt1->EndUpdate();
```

The following C# sample shows how can I change the label for a specified unit:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
    var_Chart.set_PaneWidth(0 != 0,0);
    var_Chart.LevelCount = 2;
    var_Chart.FirstVisibleDate = "1/1/2008";
```

```
    var_Chart.AllowInsideZoom = true;
    var_Chart.AllowResizeInsideZoom = false;
    var_Chart.InsideZoomOnDblClick = false;
    var_Chart.DefaultInsideZoomFormat.OwnerLabel = "<b><%d%></b> <%d2%>";
    EXG2ANTTLib.InsideZooms var_InsideZooms = var_Chart.InsideZooms;
        var_InsideZooms.SplitBaseLevel = false;
        var_InsideZooms.DefaultWidth = 32;
        var_InsideZooms.Add("1/4/2008").AllowInsideFormat = false;
axG2antt1.EndUpdate();
```

The following VFP sample shows how can I change the label for a specified unit:

```
with thisform.G2antt1
    .BeginUpdate
    with .Chart
        .PaneWidth(0) = 0
        .LevelCount = 2
        .FirstVisibleDate = {^2008-1-1}
        .AllowInsideZoom = .T.
        .AllowResizeInsideZoom = .F.
        .InsideZoomOnDblClick = .F.
        .DefaultInsideZoomFormat.OwnerLabel = "<b><%d%></b> <%d2%>"
        with .InsideZooms
            .SplitBaseLevel = .F.
            .DefaultWidth = 32
            .Add({^2008-1-4}).AllowInsideFormat = .F.
        endwith
    endwith
    .EndUpdate
endwith
```

property InsideZoomFormat.PatternChart as PatternEnum

Specifies the pattern to show on the chart.

Type	Description
PatternEnum	A PatternEnum expression that specifies the pattern to show on the specified time, on the chart area.

By default, the PatternChart property is exPatternEmpty. The PatternChartColor property determines the color to be used for pattern. The BackColorChart property determines the unit's background color. The BackColor property controls the background of the time scale unit being shown in the chart's header.

The following VB sample changes the pattern for a specified unit, in the chart area:

```
With G2antt1
   .BeginUpdate
   With .Chart
      .PaneWidth(False) = 0
      .LevelCount = 2
      .FirstVisibleDate = #1/1/2008#
      .AllowInsideZoom = True
      .AllowResizeInsideZoom = False
      .InsideZoomOnDblClick = False
      With .DefaultInsideZoomFormat
         .PatternChart = exPatternBDiagonal
         .PatternColorChart = RGB(255,0,0)
      End With
      With .InsideZooms
         .SplitBaseLevel = False
         .DefaultWidth = 18
         .Add(#1/4/2008#).AllowInsideFormat = False
      End With
   End With
   .EndUpdate
End With
```

The following VB.NET sample changes the pattern for a specified unit, in the chart area:

```
With AxG2antt1
```

```
    .BeginUpdate
    With .Chart
        .PaneWidth(False) = 0
        .LevelCount = 2
        .FirstVisibleDate = #1/1/2008#
        .AllowInsideZoom = True
        .AllowResizeInsideZoom = False
        .InsideZoomOnDblClick = False
        With .DefaultInsideZoomFormat
            .PatternChart = EXG2ANTTLib.PatternEnum.exPatternBDiagonal
            .PatternColorChart = 255
        End With
        With .InsideZooms
            .SplitBaseLevel = False
            .DefaultWidth = 18
            .Add(#1/4/2008#).AllowInsideFormat = False
        End With
    End With
    .EndUpdate
End With
```

The following C++ sample changes the pattern for a specified unit, in the chart area:

```
/*
    Copy and paste the following directives to your header file as
    it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

    #import <ExG2antt.dll>
    using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
    var_Chart->PutPaneWidth(VARIANT_FALSE,0);
    var_Chart->PutLevelCount(2);
    var_Chart->PutFirstVisibleDate("1/1/2008");
```

```
   var_Chart->PutAllowInsideZoom(VARIANT_TRUE);
   var_Chart->PutAllowResizeInsideZoom(VARIANT_FALSE);
   var_Chart->PutInsideZoomOnDblClick(VARIANT_FALSE);
   EXG2ANTTLib::IInsideZoomFormatPtr var_InsideZoomFormat = var_Chart-
>GetDefaultInsideZoomFormat();
      var_InsideZoomFormat->PutPatternChart(EXG2ANTTLib::exPatternBDiagonal);
      var_InsideZoomFormat->PutPatternColorChart(RGB(255,0,0));
   EXG2ANTTLib::IInsideZoomsPtr var_InsideZooms = var_Chart->GetInsideZooms();
      var_InsideZooms->PutSplitBaseLevel(VARIANT_FALSE);
      var_InsideZooms->PutDefaultWidth(18);
      var_InsideZooms->Add("1/4/2008")->PutAllowInsideFormat(VARIANT_FALSE);
spG2antt1->EndUpdate();
```

The following C# sample changes the pattern for a specified unit, in the chart area:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.set_PaneWidth(false,0);
   var_Chart.LevelCount = 2;
   var_Chart.FirstVisibleDate = "1/1/2008";
   var_Chart.AllowInsideZoom = true;
   var_Chart.AllowResizeInsideZoom = false;
   var_Chart.InsideZoomOnDblClick = false;
   EXG2ANTTLib.InsideZoomFormat var_InsideZoomFormat =
var_Chart.DefaultInsideZoomFormat;
      var_InsideZoomFormat.PatternChart =
EXG2ANTTLib.PatternEnum.exPatternBDiagonal;
      var_InsideZoomFormat.PatternColorChart = 255;
   EXG2ANTTLib.InsideZooms var_InsideZooms = var_Chart.InsideZooms;
      var_InsideZooms.SplitBaseLevel = false;
      var_InsideZooms.DefaultWidth = 18;
      var_InsideZooms.Add("1/4/2008").AllowInsideFormat = false;
axG2antt1.EndUpdate();
```

The following VFP sample changes the pattern for a specified unit, in the chart area:

```
with thisform.G2antt1
   .BeginUpdate
```

```
  with .Chart
    .PaneWidth(.F.) = 0
    .LevelCount = 2
    .FirstVisibleDate = {^2008-1-1}
    .AllowInsideZoom = .T.
    .AllowResizeInsideZoom = .F.
    .InsideZoomOnDblClick = .F.
    with .DefaultInsideZoomFormat
      .PatternChart = 6
      .PatternColorChart = RGB(255,0,0)
    endwith
    with .InsideZooms
      .SplitBaseLevel = .F.
      .DefaultWidth = 18
      .Add({^2008-1-4}).AllowInsideFormat = .F.
    endwith
  endwith
  .EndUpdate
endwith
```

The following Delphi sample changes the pattern for a specified unit, in the chart area:

```
with AxG2antt1 do
begin
  BeginUpdate();
  with Chart do
  begin
    PaneWidth[False] := 0;
    LevelCount := 2;
    FirstVisibleDate := '1/1/2008';
    AllowInsideZoom := True;
    AllowResizeInsideZoom := False;
    InsideZoomOnDblClick := False;
    with DefaultInsideZoomFormat do
    begin
      PatternChart := EXG2ANTTLib.PatternEnum.exPatternBDiagonal;
      PatternColorChart := 255;
```

```
    end;
    with InsideZooms do
    begin
      SplitBaseLevel := False;
      DefaultWidth := 18;
      Add('1/4/2008').AllowInsideFormat := False;
    end;
  end;
  EndUpdate();
end
```

property InsideZoomFormat.PatternColorChart as Color

Specifies the color of the pattern to show on the chart.

Type	Description
Color	A Color expression that specifies the color to show the pattern within the chart area.

By default, the PatternColorChart property is black (0). The PatternChart property specifies the pattern to show on the chart. The BackColorChart property determines the unit's background color. The BackColor property controls the background of the time scale unit being shown in the chart's header.

The following VB sample changes the pattern for a specified unit, in the chart area:

```
With G2antt1
   .BeginUpdate
   With .Chart
      .PaneWidth(False) = 0
      .LevelCount = 2
      .FirstVisibleDate = #1/1/2008#
      .AllowInsideZoom = True
      .AllowResizeInsideZoom = False
      .InsideZoomOnDblClick = False
      With .DefaultInsideZoomFormat
         .PatternChart = exPatternBDiagonal
         .PatternColorChart = RGB(255,0,0)
      End With
      With .InsideZooms
         .SplitBaseLevel = False
         .DefaultWidth = 18
         .Add(#1/4/2008#).AllowInsideFormat = False
      End With
   End With
   .EndUpdate
End With
```

The following VB.NET sample changes the pattern for a specified unit, in the chart area:

```
With AxG2antt1
```

```
    .BeginUpdate
    With .Chart
        .PaneWidth(False) = 0
        .LevelCount = 2
        .FirstVisibleDate = #1/1/2008#
        .AllowInsideZoom = True
        .AllowResizeInsideZoom = False
        .InsideZoomOnDblClick = False
        With .DefaultInsideZoomFormat
            .PatternChart = EXG2ANTTLib.PatternEnum.exPatternBDiagonal
            .PatternColorChart = 255
        End With
        With .InsideZooms
            .SplitBaseLevel = False
            .DefaultWidth = 18
            .Add(#1/4/2008#).AllowInsideFormat = False
        End With
    End With
    .EndUpdate
End With
```

The following C++ sample changes the pattern for a specified unit, in the chart area:

```
/*
    Copy and paste the following directives to your header file as
    it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

    #import <ExG2antt.dll>
    using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
    var_Chart->PutPaneWidth(VARIANT_FALSE,0);
    var_Chart->PutLevelCount(2);
    var_Chart->PutFirstVisibleDate("1/1/2008");
```

```
   var_Chart->PutAllowInsideZoom(VARIANT_TRUE);
   var_Chart->PutAllowResizeInsideZoom(VARIANT_FALSE);
   var_Chart->PutInsideZoomOnDblClick(VARIANT_FALSE);
   EXG2ANTTLib::IInsideZoomFormatPtr var_InsideZoomFormat = var_Chart-
>GetDefaultInsideZoomFormat();
      var_InsideZoomFormat->PutPatternChart(EXG2ANTTLib::exPatternBDiagonal);
      var_InsideZoomFormat->PutPatternColorChart(RGB(255,0,0));
   EXG2ANTTLib::IInsideZoomsPtr var_InsideZooms = var_Chart->GetInsideZooms();
      var_InsideZooms->PutSplitBaseLevel(VARIANT_FALSE);
      var_InsideZooms->PutDefaultWidth(18);
      var_InsideZooms->Add("1/4/2008")->PutAllowInsideFormat(VARIANT_FALSE);
spG2antt1->EndUpdate();
```

The following C# sample changes the pattern for a specified unit, in the chart area:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.set_PaneWidth(false,0);
   var_Chart.LevelCount = 2;
   var_Chart.FirstVisibleDate = "1/1/2008";
   var_Chart.AllowInsideZoom = true;
   var_Chart.AllowResizeInsideZoom = false;
   var_Chart.InsideZoomOnDblClick = false;
   EXG2ANTTLib.InsideZoomFormat var_InsideZoomFormat =
var_Chart.DefaultInsideZoomFormat;
      var_InsideZoomFormat.PatternChart =
EXG2ANTTLib.PatternEnum.exPatternBDiagonal;
      var_InsideZoomFormat.PatternColorChart = 255;
   EXG2ANTTLib.InsideZooms var_InsideZooms = var_Chart.InsideZooms;
      var_InsideZooms.SplitBaseLevel = false;
      var_InsideZooms.DefaultWidth = 18;
      var_InsideZooms.Add("1/4/2008").AllowInsideFormat = false;
axG2antt1.EndUpdate();
```

The following VFP sample changes the pattern for a specified unit, in the chart area:

```
with thisform.G2antt1
   .BeginUpdate
```

```
  with .Chart
    .PaneWidth(.F.) = 0
    .LevelCount = 2
    .FirstVisibleDate = {^2008-1-1}
    .AllowInsideZoom = .T.
    .AllowResizeInsideZoom = .F.
    .InsideZoomOnDblClick = .F.
    with .DefaultInsideZoomFormat
      .PatternChart = 6
      .PatternColorChart = RGB(255,0,0)
    endwith
    with .InsideZooms
      .SplitBaseLevel = .F.
      .DefaultWidth = 18
      .Add({^2008-1-4}).AllowInsideFormat = .F.
    endwith
  endwith
  .EndUpdate
endwith
```

The following Delphi sample changes the pattern for a specified unit, in the chart area:

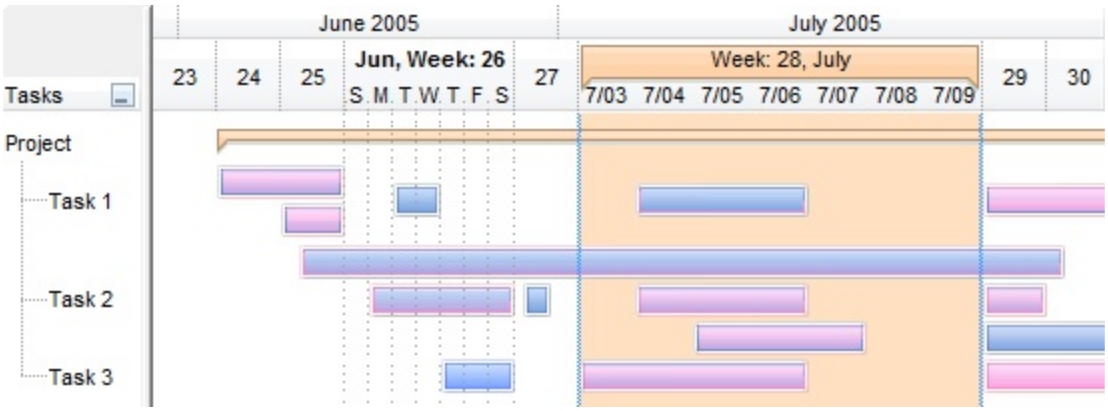
```
with AxG2antt1 do
begin
  BeginUpdate();
  with Chart do
  begin
    PaneWidth[False] := 0;
    LevelCount := 2;
    FirstVisibleDate := '1/1/2008';
    AllowInsideZoom := True;
    AllowResizeInsideZoom := False;
    InsideZoomOnDblClick := False;
    with DefaultInsideZoomFormat do
    begin
      PatternChart := EXG2ANTTLib.PatternEnum.exPatternBDiagonal;
      PatternColorChart := 255;
```

```
    end;
    with InsideZooms do
    begin
      SplitBaseLevel := False;
      DefaultWidth := 18;
      Add('1/4/2008').AllowInsideFormat := False;
    end;
  end;
  EndUpdate();
end
```

InsideZooms object

The InsideZooms holds a collection of InsideZoom objects. The InsideZooms property retrieves the chart's collection of units being displayed in a different way, or getting zoomed or magnified. The chart may display this kind of units only if the AllowInsideZoom property is True. The InsideZoomFormat object defines the format of units being zoomed ie background, foreground colors, grid lines, labels, and so on. The inside zoom feature allows displaying portions of the chart with different time scale units. For instance, you can display the bars on hours, while the chart still displays days. Once the AllowInsideZoom property is True, the user can double clicks the chart's header, so this portion gets magnified. Also, at runtime, the user can resize the time scale units, so the unit gets magnified.

The following chart displays weeks, and the week 26 and 28 gets displayed in a different way:



The InsideZooms object supports the following properties and methods:

Name	Description
Add	Adds a InsideZoom object to the collection and returns a reference to the newly created object.
Clear	Removes all objects in a collection.
Contains	Returns the InsideZoom object that contains the specified date-time.
Count	Returns the number of objects in a collection.
DefaultWidth	Specifies the default width in pixels for inside zoom units.
Item	Returns a specific InsideZoom of the InsideZooms collection.
Remove	Removes a specific member from the InsideZooms collection.
	Gets or sets a value that indicates whether the chart's

SplitBaseLevel	base level is splitted when inside zoom units are shown.

method InsideZooms.Add (DateTime as Variant)

Adds a InsideZoom object to the collection and returns a reference to the newly created object.

Type	Description
DateTime as Variant	A Date expression being zoomed.
Return	**Description**
InsideZoom	An InsideZoom object being created.

The Add method magnifies a date, and retrieves the InsideZoom object that may be used to customize the zoomed date. In other words, you can use the Add method to programmatically zoom, magnify, or change the visual appearance for a specified time unit. The control fires the InsideZoom event once a new date gets magnified. The Add method retrieves nothing, if the AllowInsideZoom property is False. An inside zoom unit may display a different label, background and so on. The DefaultInsideZoomFormat retrieves an InsideZoomFormat object that customizes the dates being magnified. Use the Width property to specify the width of the time scale unit being changed. Once a new date is added to InsideZooms property, the DefaultWidth property specifies the default width being used.

The following VB sample shows how can I change the background color for a time unit, in the chart area:

```
With G2antt1
  .BeginUpdate
  With .Chart
    .LevelCount = 2
    .FirstVisibleDate = #1/1/2008#
    .AllowInsideZoom = True
    .AllowResizeInsideZoom = False
    .InsideZoomOnDblClick = False
    .DefaultInsideZoomFormat.BackColorChart = 255
    With .InsideZooms
      .SplitBaseLevel = False
      .DefaultWidth = 18
      .Add(#1/4/2008#).AllowInsideFormat = False
    End With
  End With
  .EndUpdate
```

```
End With
```

The following VB.NET sample shows how can I change the background color for a time unit, in the chart area:

```
With AxG2antt1
   .BeginUpdate
   With .Chart
      .LevelCount = 2
      .FirstVisibleDate = #1/1/2008#
      .AllowInsideZoom = True
      .AllowResizeInsideZoom = False
      .InsideZoomOnDblClick = False
      .DefaultInsideZoomFormat.BackColorChart = 255
      With .InsideZooms
         .SplitBaseLevel = False
         .DefaultWidth = 18
         .Add(#1/4/2008#).AllowInsideFormat = False
      End With
   End With
   .EndUpdate
End With
```

The following C++ sample shows how can I change the background color for a time unit, in the chart area:

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutLevelCount(2);
```

```
    var_Chart->PutFirstVisibleDate("1/1/2008");
    var_Chart->PutAllowInsideZoom(VARIANT_TRUE);
    var_Chart->PutAllowResizeInsideZoom(VARIANT_FALSE);
    var_Chart->PutInsideZoomOnDblClick(VARIANT_FALSE);
    var_Chart->GetDefaultInsideZoomFormat()->PutBackColorChart(255);
    EXG2ANTTLib::IInsideZoomsPtr var_InsideZooms = var_Chart->GetInsideZooms();
        var_InsideZooms->PutSplitBaseLevel(VARIANT_FALSE);
        var_InsideZooms->PutDefaultWidth(18);
        var_InsideZooms->Add("1/4/2008")->PutAllowInsideFormat(VARIANT_FALSE);
spG2antt1->EndUpdate();
```

The following C# sample shows how can I change the background color for a time unit, in the chart area:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
    var_Chart.LevelCount = 2;
    var_Chart.FirstVisibleDate = "1/1/2008";
    var_Chart.AllowInsideZoom = true;
    var_Chart.AllowResizeInsideZoom = false;
    var_Chart.InsideZoomOnDblClick = false;
    var_Chart.DefaultInsideZoomFormat.BackColorChart = 255;
    EXG2ANTTLib.InsideZooms var_InsideZooms = var_Chart.InsideZooms;
        var_InsideZooms.SplitBaseLevel = false;
        var_InsideZooms.DefaultWidth = 18;
        var_InsideZooms.Add("1/4/2008").AllowInsideFormat = false;
axG2antt1.EndUpdate();
```

The following VFP sample shows how can I change the background color for a time unit, in the chart area:

```
with thisform.G2antt1
    .BeginUpdate
    with .Chart
        .LevelCount = 2
        .FirstVisibleDate = {^2008-1-1}
        .AllowInsideZoom = .T.
        .AllowResizeInsideZoom = .F.
```

```
    .InsideZoomOnDblClick = .F.
    .DefaultInsideZoomFormat.BackColorChart = 255
    with .InsideZooms
      .SplitBaseLevel = .F.
      .DefaultWidth = 18
      .Add({^2008-1-4}).AllowInsideFormat = .F.
    endwith
  endwith
  .EndUpdate
endwith
```

method InsideZooms.Clear ()

Removes all objects in a collection.

Type	Description

The Clear method clears the inside zoom units. Use the AllowInsideZoom property to show or hide the inside zoom units. Use the Remove method to remove a specific date from InsideZooms collection. Use the Width property on 0, to hide a specified date. Use the AllowResize property to specify whether the user can resize an inside zoom unit.

property InsideZooms.Contains (DateTime as Variant) as InsideZoom

Returns the InsideZoom object that contains the specified date-time.

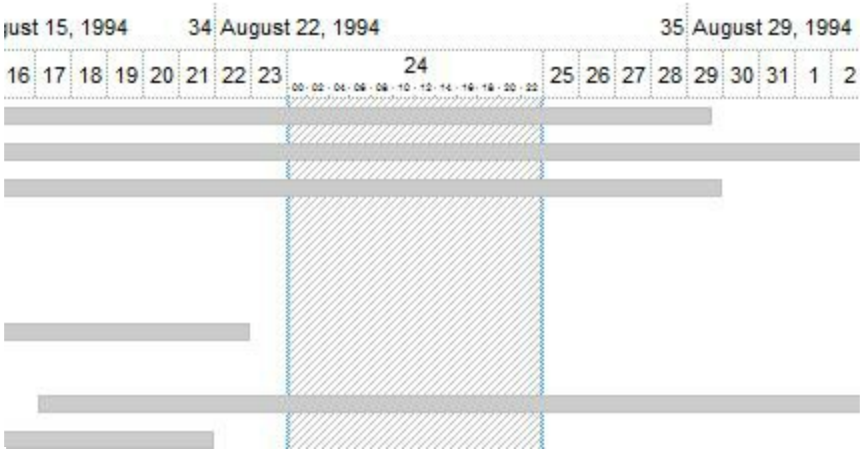
Type	Description
DateTime as Variant	A Date-Time expression that specifies the inside zoom unit being searched
InsideZoom	An InsideZoom object being found, or nothing if the date-time is not found

The Contains property returns the InsideZoom object that contains the specified date-time. The Item property looks for the exact date-time, while the Contains search for the giving date-time. The Count property specifies the number of inside zoom units. The InsideZooms collection may be enumerated using *for each* statements.

The following sample shows the Contains implementation:

```
Private Function InsideZoomContains(ByVal g As EXG2ANTTLibCtl.G2antt, ByVal d As
Date) As EXG2ANTTLibCtl.InsideZoom
   Dim i As EXG2ANTTLibCtl.InsideZoom
   With g.Chart
      For Each i In .InsideZooms
         If (d >= i.StartDate) Then
            If (d < i.EndDate) Then
               Set InsideZoomContains = i
               Exit Function
            End If
         End If
      Next
   End With
   Set InsideZoomContains = Nothing
End Function
```

For instance, in the following screen shot, the August 24, 1994 is zoomed, and so the InsideZoom object starts from #08/24/1994# (StartDate), and ends on #08/25/1994# (EndDate)



In this case, the Item(#08/24/1994#) returns the InsideZoom object, while any other date between start and end returns nothing. In case, you are using any mid-date, you have to use the Contains property that searches for the specified date-time.

property InsideZooms.Count as Long

Returns the number of objects in a collection.

Type	Description
Long	A long expression that specifies the number of inside zoom units.

The Count property specifies the number of inside zoom units. Use the Add method to add new inside zoom units. Use the Item property to access an inside zoom unit. The InsideZooms collection may be enumerated using *for each* statements.

property InsideZooms.DefaultWidth as Long

Specifies the default width in pixels for inside zoom units.

Type	Description
Long	A long expression that specifies the default width in pixels for newly added inside zoom units.

By default, the DefaultWidth property is 128 pixels. Changing the DefaultWidth property has no effect for already added inside zoom units. In this case, you can use the Width property to change the width for units. Use the AllowResize property to specify whether a specified unit is resizable. Use the CondInsideZoom property to specify a formula to define the time units that may be resized if the AllowResizeInsideZoom property is True.

The following VB sample shows how can I change the label for a specified unit:

```
With G2antt1
   .BeginUpdate
   With .Chart
      .PaneWidth(0) = 0
      .LevelCount = 2
      .FirstVisibleDate = #1/1/2008#
      .AllowInsideZoom = True
      .AllowResizeInsideZoom = False
      .InsideZoomOnDblClick = False
      .DefaultInsideZoomFormat.OwnerLabel = "<b><%d%></b> <%d2%>"
      With .InsideZooms
         .SplitBaseLevel = False
         .DefaultWidth = 32
         .Add(#1/4/2008#).AllowInsideFormat = False
      End With
   End With
   .EndUpdate
End With
```

The following VB.NET sample shows how can I change the label for a specified unit:

```
With AxG2antt1
   .BeginUpdate
   With .Chart
```

```
    .PaneWidth(0) = 0
    .LevelCount = 2
    .FirstVisibleDate = #1/1/2008#
    .AllowInsideZoom = True
    .AllowResizeInsideZoom = False
    .InsideZoomOnDblClick = False
    .DefaultInsideZoomFormat.OwnerLabel = "<b><%d%></b> <%d2%>"
    With .InsideZooms
      .SplitBaseLevel = False
      .DefaultWidth = 32
      .Add(#1/4/2008#).AllowInsideFormat = False
    End With
  End With
  .EndUpdate
End With
```

The following C++ sample shows how can I change the label for a specified unit:

```
/*
  Copy and paste the following directives to your header file as
  it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

  #import <ExG2antt.dll>
  using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
  var_Chart->PutPaneWidth(0,0);
  var_Chart->PutLevelCount(2);
  var_Chart->PutFirstVisibleDate("1/1/2008");
  var_Chart->PutAllowInsideZoom(VARIANT_TRUE);
  var_Chart->PutAllowResizeInsideZoom(VARIANT_FALSE);
  var_Chart->PutInsideZoomOnDblClick(VARIANT_FALSE);
  var_Chart->GetDefaultInsideZoomFormat()->PutOwnerLabel(L"<b><%d%></b>
<%d2%>");
```

```
    EXG2ANTTLib::IInsideZoomsPtr var_InsideZooms = var_Chart->GetInsideZooms();
        var_InsideZooms->PutSplitBaseLevel(VARIANT_FALSE);
        var_InsideZooms->PutDefaultWidth(32);
        var_InsideZooms->Add("1/4/2008")->PutAllowInsideFormat(VARIANT_FALSE);
spG2antt1->EndUpdate();
```

The following C# sample shows how can I change the label for a specified unit:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
    var_Chart.set_PaneWidth(0 != 0,0);
    var_Chart.LevelCount = 2;
    var_Chart.FirstVisibleDate = "1/1/2008";
    var_Chart.AllowInsideZoom = true;
    var_Chart.AllowResizeInsideZoom = false;
    var_Chart.InsideZoomOnDblClick = false;
    var_Chart.DefaultInsideZoomFormat.OwnerLabel = "<b><%d%></b> <%d2%>";
    EXG2ANTTLib.InsideZooms var_InsideZooms = var_Chart.InsideZooms;
        var_InsideZooms.SplitBaseLevel = false;
        var_InsideZooms.DefaultWidth = 32;
        var_InsideZooms.Add("1/4/2008").AllowInsideFormat = false;
axG2antt1.EndUpdate();
```

The following VFP sample shows how can I change the label for a specified unit:

```
with thisform.G2antt1
    .BeginUpdate
    with .Chart
        .PaneWidth(0) = 0
        .LevelCount = 2
        .FirstVisibleDate = {^2008-1-1}
        .AllowInsideZoom = .T.
        .AllowResizeInsideZoom = .F.
        .InsideZoomOnDblClick = .F.
        .DefaultInsideZoomFormat.OwnerLabel = "<b><%d%></b> <%d2%>"
        with .InsideZooms
            .SplitBaseLevel = .F.
            .DefaultWidth = 32
```

```
        .Add({^2008-1-4}).AllowInsideFormat = .F.
      endwith
    endwith
    .EndUpdate
endwith
```

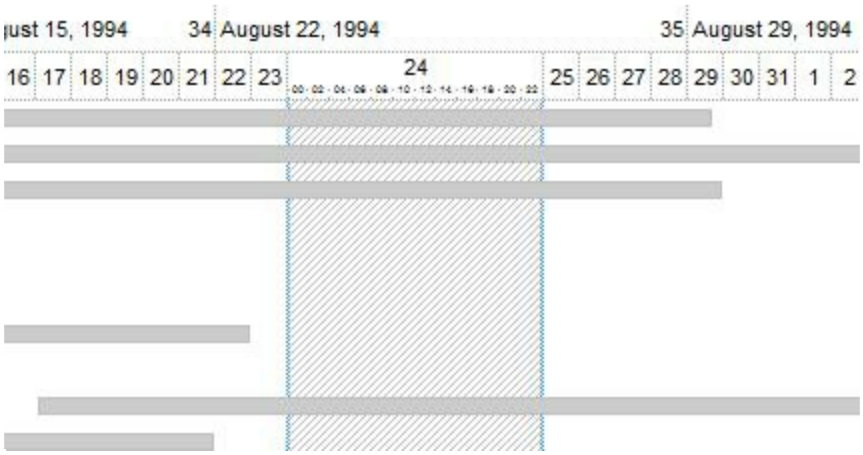
property InsideZooms.Item (DateTime as Variant) as InsideZoom

Returns a specific InsideZoom of the InsideZooms collection.

Type	Description
DateTime as Variant	A Date-Time expression that specifies the inside zoom unit being requested, or a long expression that indicates the index of the inside zoom object being requested
InsideZoom	An InsideZoom object being requested

The Item property retrieves nothing, if the specified inside zoom unit does not exists. The Item property looks for the exact date-time, while the Contains search for the giving date-time. The Count property specifies the number of inside zoom units. The InsideZooms collection may be enumerated using *for each* statements.

For instance, in the following screen shot, the August 24, 1994 is zoomed, and so the InsideZoom object starts from #08/24/1994# (StartDate), and ends on #08/25/1994# (EndDate)



In this case, the Item(#08/24/1994#) returns the InsideZoom object, while any other date between start and end returns nothing. In case, you are using any mid-date, you have to use the Contains property that searches for the specified date-time.

method InsideZooms.Remove (DateTime as Variant)

Removes a specific member from the InsideZooms collection.

Type	Description
DateTime as Variant	A Date-Time expression that specifies the inside zoom unit being deleted, or a long expression that indicates the index of the inside zoom object being deleted

Use the Remove method to remove a specified inside zoom unit. Use the Item property to check if a specified Date-Time is an inside zoom unit, or if the InsideZooms collection contains it. Use the Clear method to clears the InsideZooms collection. Use the AllowInsideZoom property to show or hide the inside zoom units. Use the Width property on 0, to hide a specified date. Use the AllowResize property to specify whether the user can resize an inside zoom unit.

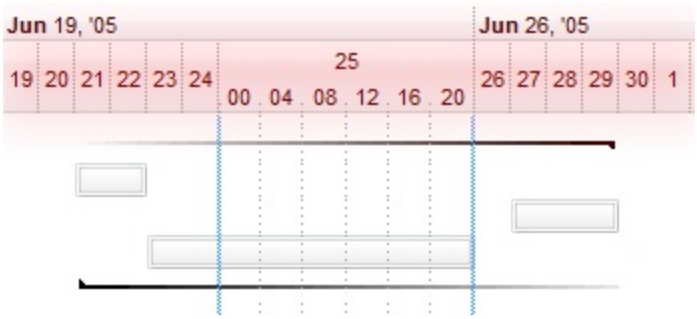
property InsideZooms.SplitBaseLevel as Boolean

Gets or sets a value that indicates whether the chart's base level is splitted when inside zoom units are shown.

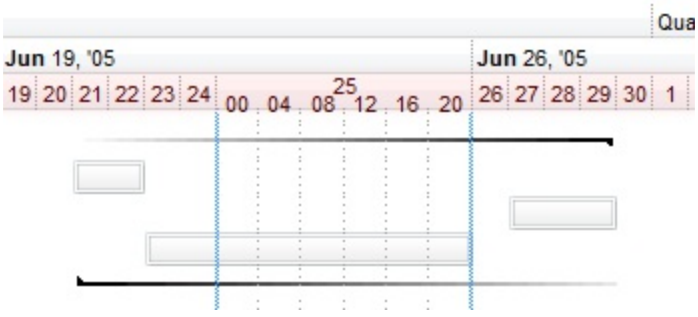
Type	Description
Boolean	A Boolean expression that specifies whether the base level of the chart is divided to display the inside zoom units.

By default, the SplitBaseLevel property is True, which means that once the chart shows inside zoom units, the base level is divided to display the owner and the inside levels. The AllowInsideZoom property specifies whether the chart may display inside zoom units. Use the AllowInsideFormat property to show the inside level. Use the DisplayOwnerLabel property to specify whether the owner level is displayed or not.

The following screen shot shows levels if the SplitBaseLevel property is True (the base level is shown in two lines):



The following screen shot shows levels if the SplitBaseLevel property is False (the base level is shown in one line):



The following VB sample shows how can I change the background color for a time unit:

```
With G2antt1
   .BeginUpdate
   With .Chart
      .LevelCount = 2
```

```
      .FirstVisibleDate = #1/1/2008#
      .AllowInsideZoom = True
      .AllowResizeInsideZoom = False
      .InsideZoomOnDblClick = False
      .DefaultInsideZoomFormat.BackColor = 255
      With .InsideZooms
         .SplitBaseLevel = False
         .DefaultWidth = 18
         .Add(#1/4/2008#).AllowInsideFormat = False
      End With
   End With
   .EndUpdate
End With
```

The following VB.NET sample shows how can I change the background color for a time unit:

```
With AxG2antt1
   .BeginUpdate
   With .Chart
      .LevelCount = 2
      .FirstVisibleDate = #1/1/2008#
      .AllowInsideZoom = True
      .AllowResizeInsideZoom = False
      .InsideZoomOnDblClick = False
      .DefaultInsideZoomFormat.BackColor = 255
      With .InsideZooms
         .SplitBaseLevel = False
         .DefaultWidth = 18
         .Add(#1/4/2008#).AllowInsideFormat = False
      End With
   End With
   .EndUpdate
End With
```

The following C++ sample shows how can I change the background color for a time unit:

```
/*
```

```
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutLevelCount(2);
   var_Chart->PutFirstVisibleDate("1/1/2008");
   var_Chart->PutAllowInsideZoom(VARIANT_TRUE);
   var_Chart->PutAllowResizeInsideZoom(VARIANT_FALSE);
   var_Chart->PutInsideZoomOnDblClick(VARIANT_FALSE);
   var_Chart->GetDefaultInsideZoomFormat()->PutBackColor(255);
   EXG2ANTTLib::IInsideZoomsPtr var_InsideZooms = var_Chart->GetInsideZooms();
      var_InsideZooms->PutSplitBaseLevel(VARIANT_FALSE);
      var_InsideZooms->PutDefaultWidth(18);
      var_InsideZooms->Add("1/4/2008")->PutAllowInsideFormat(VARIANT_FALSE);
spG2antt1->EndUpdate();
```

The following C# sample shows how can I change the background color for a time unit:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.LevelCount = 2;
   var_Chart.FirstVisibleDate = "1/1/2008";
   var_Chart.AllowInsideZoom = true;
   var_Chart.AllowResizeInsideZoom = false;
   var_Chart.InsideZoomOnDblClick = false;
   var_Chart.DefaultInsideZoomFormat.BackColor = 255;
   EXG2ANTTLib.InsideZooms var_InsideZooms = var_Chart.InsideZooms;
      var_InsideZooms.SplitBaseLevel = false;
      var_InsideZooms.DefaultWidth = 18;
      var_InsideZooms.Add("1/4/2008").AllowInsideFormat = false;
axG2antt1.EndUpdate();
```

The following VFP sample shows how can I change the background color for a time unit:

```
with thisform.G2antt1
   .BeginUpdate
   with .Chart
      .LevelCount = 2
      .FirstVisibleDate = {^2008-1-1}
      .AllowInsideZoom = .T.
      .AllowResizeInsideZoom = .F.
      .InsideZoomOnDblClick = .F.
      .DefaultInsideZoomFormat.BackColor = 255
      with .InsideZooms
         .SplitBaseLevel = .F.
         .DefaultWidth = 18
         .Add({^2008-1-4}).AllowInsideFormat = .F.
      endwith
   endwith
   .EndUpdate
endwith
```

Items object

The Items object contains a collection of items. Each item is identified by a handle HITEM. The HITEM is of long type. Each item contains a collection of cells. The number of cells is determined by the number of Column objects in the control. To access the Items collection use Items property of the control. Using the Items collection you can add, remove or change the control items. The Items collection can be organized as a hierarchy or as a tabular data.

The following screen shot shows the items of the control:

The Items collection supports the following properties and methods:

Name	Description
AcceptSetParent	Retrieves a value indicating whether the SetParent method can be accomplished..
AddBar	Adds a bar to an item.
AddItem	Adds a new item, and returns a handle to the newly created item.
AddLink	Links a bar to another.
AllowCellValueToItemBar	Retrieves or sets a value that indicates whether the cells display associated properties of the bars in the item.
CellBackColor	Retrieves or sets the cell's background color.
CellBold	Retrieves or sets a value that indicates whether the cell's caption should appear in bold.
CellButtonAutoWidth	Retrieves or sets a value indicating whether the cell's button fits the cell's caption.

CellCaption	Gets the cell's display value.
CellChecked	Retrieves the cell's handle that is checked on a specific radio group.
CellData	Retrieves or sets the extra data for a specific cell.
CellEditor	Creates and gets the cell's built-in editor.
CellEditorVisible	Specifies whether column's editor is visible or hidden in the cell.
CellEnabled	Returns or sets a value that determines whether a cell can respond to user-generated events.
CellFont	Retrieves or sets the cell's font.
CellForeColor	Retrieves or sets the cell's foreground color.
CellFormatLevel	Specifies the arrangement of the fields inside the cell.
CellHAlignment	Retrieves or sets a value that indicates the alignment of the cell's caption.
CellHasButton	Retrieves or sets a value indicating whether the cell has associated a push button or not.
CellHasCheckBox	Retrieves or sets a value indicating whether the cell has associated a checkbox or not.
CellHasRadioButton	Retrieves or sets a value indicating whether the cell has associated a radio button or not.
CellHyperLink	Specifies whether the cell's is highlighted when the cursor mouse is over the cell.
CellImage	Retrieves or sets an Image that is displayed on the cell's area.
CellImages	Specifies an additional list of icons shown in the cell.
CellItalic	Retrieves or sets a value that indicates whether the cell's caption should appear in italic.
CellItem	Retrieves the handle of item that is the owner of a specific cell.
CellMerge	Retrieves or sets a value that indicates the index of the cell that's merged to.
CellParent	Retrieves the parent of an inner cell.
CellPicture	Retrieves or sets a value that indicates the Picture object displayed by the cell.
	Retrieves or sets a value that indicates the height of the

CellPictureHeight	cell's picture.
CellPictureWidth	Retrieves or sets a value that indicates the width of the cell's picture.
CellRadioGroup	Retrieves or sets a value indicating the radio group where the cell is contained.
CellSingleLine	Retrieves or sets a value indicating whether the cell's caption is painted using one or more lines.
CellState	Retrieves or sets the cell's state. Has effect only for check and radio cells.
CellStrikeOut	Retrieves or sets a value that indicates whether the cell's caption should appear in strikeout.
CellToolTip	Retrieves or sets a text that is used to show the tooltip's cell.
CellUnderline	Retrieves or sets a value that indicates whether the cell's caption should appear in underline.
CellVAlignment	Retrieves or sets a value that indicates how the cell's caption is vertically aligned.
CellValue	Specifies the cell's value.
CellValueFormat	Specifies how the cell's caption is displayed.
CellValueToItemBar	Indicates whether the cell displays the specified property of the bar.
CellWidth	Retrieves or sets a value that indicates the width of the inner cell.
ChildCount	Retrieves the number of children items.
ClearBars	Clears the bars from the item.
ClearCellBackColor	Clears the cell's background color.
ClearCellForeColor	Clears the cell's foreground color.
ClearCellHAlignment	Clears the cell's alignment.
ClearItemBackColor	Clears the item's background color.
ClearItemForeColor	Clears the item's foreground color.
ClearLinks	Clears all links in the chart.
ComputeValue	Computes the value of a specified formula.
DefaultItem	Retrieves or sets the default item.
DefineSummaryBars	Defines the bars that belongs to a summary bar.

DefSchedulePDM	Retrieves or sets an option for SchedulePDM method.
DeleteCellEditor	Deletes the cell's built-in editor created by CellEditor property.
EnableItem	Returns or sets a value that determines whether a item can respond to user-generated events.
EndBlockUndoRedo	Ends recording the UI operations and adds the undo/redo operations as a block, so they all can be restored at once, if Undo method is performed.
EndUpdateBar	Adds programmatically updated properties of the bar to undo/redo queue.
EndUpdateLink	Adds programmatically updated properties of the link to undo/redo queue.
EnsureVisibleBar	Ensures that the given item-bar fits the chart's visible area.
EnsureVisibleItem	Ensures that the given item fits the control's visible area
ExpandItem	Expands, or collapses, the child items of the specified item.
FindBar	Finds the item that hosts the specified bar.
FindItem	Finds an item, looking for Caption in ColIndex colum. The searching starts at StartIndex item.
FindItemData	Finds the item giving its data.
FindPath	Finds the item, given its path. The control searches the path on the SearchColumnIndex column.
FirstItemBar	Gets the key of the first bar in the item.
FirstLink	Gets the key of the first link.
FirstVisibleItem	Retrieves the handle of the first visible item into control.
FocusItem	Retrieves the handle of item that has the focus.
FormatCell	Specifies the custom format to display the cell's content.
FullPath	Returns the fully qualified path of the referenced item in the control. The caption is taken from the column SearchColumnIndex.
GroupBars	Groups two bars.
GroupItem	Indicates a group item if positive, and the value specifies the index of the column that has been grouped.
HasCellEditor	Specifies whether a cell has a built-in editor.

InnerCell	Retrieves the inner cell.
InsertControlItem	Inserts a new item of ActiveX type, and returns a handle to the newly created item.
InsertItem	Inserts a new item, and returns a handle to the newly created item.
IntersectBars	Specifies whether two bars intersect if returns 0, if 1 A is before B and -1 if A is after bar B.
IsItemLocked	Returns a value that indicates whether the item is locked or unlocked.
IsItemVisible	Checks if the specific item is in the visible client area.
ItemAllowSizing	Retrieves or sets a value that indicates whether a user can resize the item at run-time.
ItemAppearance	Specifies the item's appearance when the item hosts an ActiveX control.
ItemBackColor	Retrieves or sets a background color for a specific item.
ItemBar	Gets or sets a bar property.
ItemBarEx	Gets or sets the property's bar that matches the criteria.
ItemBold	Retrieves or sets a value that indicates whether the item should appear in bold.
ItemByIndex	Retrieves the handle of the item given its index in Items collection..
ItemCell	Retrieves the cell's handle based on a specific column.
ItemChild	Retrieves the child of a specified item.
ItemControlID	Retrieves the item's control identifier that was used by InsertControlItem.
ItemCount	Retrieves the number of items.
ItemData	Retrieves or sets the extra data for a specific item.
ItemDivider	Specifies whether the item acts like a divider item. The value indicates the index of column used to define the divider's title.
ItemDividerLine	Defines the type of line in the divider item.
ItemDividerLineAlignment	Specifies the alignment of the line in the divider item.
ItemFiltered	Checks whether the item is included in the control's filter.
ItemFont	Retrieves or sets the item's font.

ItemForeColor	Retrieves or sets a foreground color for a specific item.
ItemHasChildren	Adds an expand button to left side of the item even if the item has no child items.
ItemHeight	Retrieves or sets the item's height.
ItemItalic	Retrieves or sets a value that indicates whether the item should appear in italic.
ItemMaxHeight	Retrieves or sets a value that indicates the maximum height when the item's height is variable.
ItemMinHeight	Retrieves or sets a value that indicates the minimum height when the item's height is sizing.
ItemNonworkingUnits	Gets or sets a value that indicates the formula to specify the use non-working units for the item.
ItemObject	Retrieves the ActiveX object associated, if the item was created using InsertControlItem method.
ItemParent	Returns the handle of parent item.
ItemPosition	Retrieves or sets a value that indicates the item's position in the children list.
ItemStrikeOut	Retrieves or sets a value that indicates whether the item should appear in strikeout.
ItemToIndex	Retrieves the index of item into Items collection given its handle.
ItemUnderline	Retrieves or sets a value that indicates whether the item should appear in underline.
ItemWidth	Retrieves or sets a value that indicates the item's width while it contains an ActiveX control.
ItemWindowHost	Retrieves the window's handle that hosts an ActiveX control when the item was created using InsertControlItem.
ItemWindowHostCreateStyle	Retrieves or sets a value that indicates a combination of window styles used to create the ActiveX window host.
LastVisibleItem	Retrieves the handle of the last visible item.
Link	Gets or sets a property for a link.
LockedItem	Retrieves the handle of the locked/fixed item.
LockedItemCount	Specifies the number of items fixed on the top or bottom side of the control.

MatchItemCount	Retrieves the number of items that match the filter.
MergeCells	Merges a list of cells.
NextItemBar	Gets the key of the next bar in the item.
NextLink	Gets the key of the next link.
NextSiblingItem	Retrieves the next sibling of the item in the parent's child list.
NextVisibleItem	Retrieves the handle of next visible item.
PathSeparator	Returns or sets the delimiter character used for the path returned by the FullPath property.
PrevSiblingItem	Retrieves the previous sibling of the item in the parent's child list.
PrevVisibleItem	Retrieves the handle of previous visible item.
RemoveAllItems	Removes all items from the control.
RemoveBar	Removes a bar from an item.
RemoveItem	Removes a specific item.
RemoveLink	Removes a link.
RemoveLinksOf	Removes the links that goes or ends on the specified bar.
RemoveSelection	Removes the selected items (including the descendents).
RootCount	Retrieves the number of root objects into Items collection.
RootItem	Retrieves the handle of the root item giving its index into the root items collection.
SchedulePDM	Schedules the chart using the Precedence Diagram Method.
SelectableItem	Specifies whether the user can select the item.
SelectAll	Selects all items.
SelectCount	Retrieves the handle of selected item giving its index in selected items collection.
SelectedItem	Retrieves the selected item's handle given its index in selected items collection.
SelectedObjects	Retrieves a collection of selected objects in the chart.
SelectItem	Selects or unselects a specific item.
SelectPos	Selects items by position.
SetParent	Changes the parent of the given item.

SortableItem	Specifies whether the item is sortable.
SortChildren	Sorts the child items of the given parent item in the control. SortChildren will not recurse through the tree, only the immediate children of Item will be sorted.
SplitCell	Splits a cell, and returns the inner created cell.
StartBlockUndoRedo	Starts recording the UI operations as a block of undo/redo operations.
StartUpdateBar	Starts changing properties of the bar, so EndUpdateBar method adds programmatically updated properties to undo/redo queue.
StartUpdateLink	Starts changing properties of the link, so EndUpdateLink method adds programmatically updated properties to undo/redo queue.
UndefineSummaryBars	Undefines the bars in a summary bar
UngroupBars	Ungroups two bars.
UnmergeCells	Unmerges a list of cells.
UnselectAll	Unselects all items.
UnsplitCell	Unsplits a cell.
VisibleCount	Retrieves the number of visible items.
VisibleItemCount	Retrieves the number of visible items.

property Items.AcceptSetParent (Item as HITEM, NewParent as HITEM) as Boolean

Retrieves a value indicating whether the SetParent method can be accomplished.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item being moved.
NewParent as HITEM	A long expression that indicates the handle of the parent item where the item should be moved.
Boolean	A boolean expression that indicates whether the item can be child of the NewParent item.

Use this property to make sure that SetParent can be called. The AcceptSetParent property checks if an item can be child of another item.

method Items.AddBar (Item as HITEM, BarName as Variant, DateStart as Variant, DateEnd as Variant, [Key as Variant], [Text as Variant])

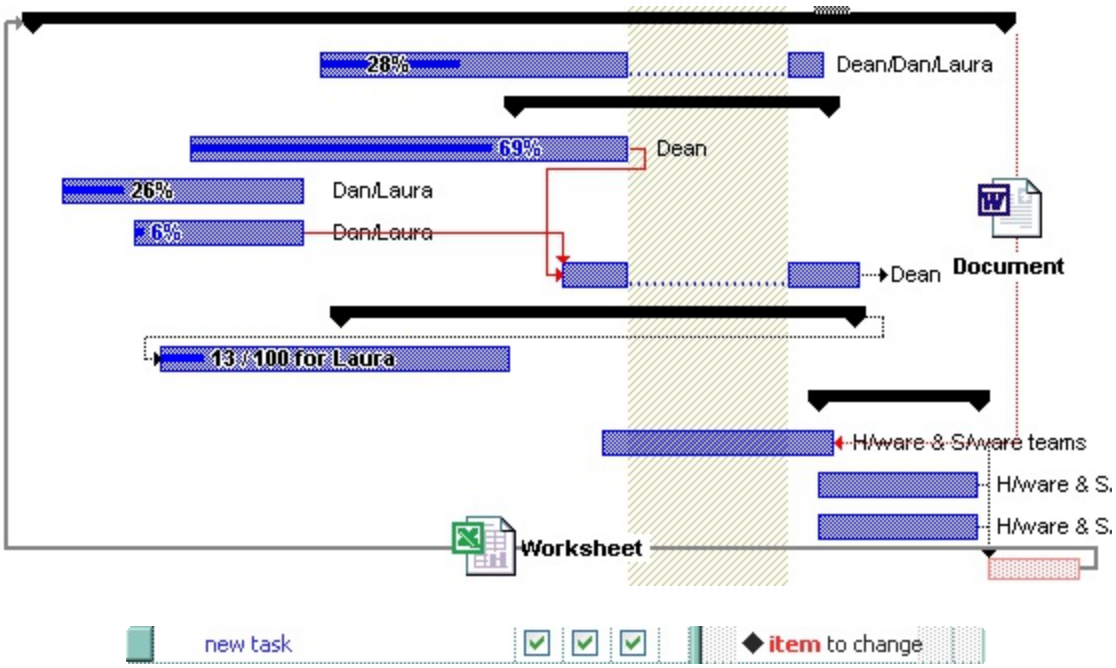
Adds a bar to an item.

Type	Description
Item as HITEM	A long expression that indicates the the handle of the item where the bar is inserted. Use the ItemBar(exBarParent) property to access later the handle of the item that hosts the bar.
BarName as Variant	A String expression that indicates the name of the bar being inserted, or a long expression that indicates the index of the bar being inserted. You can find a list of predefined bars here. Use the ItemBar(exBarName) property to access later the bar's name.
DateStart as Variant	A Date expression that indicates the date/time where the bar starts, or a string expression that indicates the start date and time. For instance, the "6/10/2003 10:13", indicates the date and the time. Use the ItemBar(exBarStart) property to access later the start date of the bar.
DateEnd as Variant	A Date expression that indicates the date where the bar ends, or a string expression that indicates the end date and time. For instance, the "6/10/2003 10:13", indicates the date and the time. Use the ItemBar(exBarEnd) property to access later the end point of the bar.
Key as Variant	Optional. A String expression that indicates the key of the bar being inserted. If missing, the Key parameter is empty. If the Item has only a single Bar you can not use the Key parameter, else an unique key should be used. Use the ItemBar(exBarKey) property to access later the key of the bar.
Text as Variant	Optional. A String expression that indicates the text being displayed. The Text may include built-in HTML format. Use the ItemBar(exBarCaption) property to access later the caption of the bar. Use the ItemBar(exBarHAlignCaption/exBarVAlignCaption) to display and align the caption of the bar inside or outside of the bar.

Use the AddBar method to *add or move* a bar to an item. The Key parameter indicates the

key of the bar being added or moved(replaced). If the item contains no bars with the giving key, a new bar is added. If the bar already exists, the new coordinates are updated (the bar is moved, so the new starting and ending points of the bar are indicated by DateStart and DateEnd parameters). In other words, an item can not contain several bars with the same key, or adding several bars require unique keys per item.

Use the ShowEmptyBars property to show the bars, even if the start and end dates are identical. If you want to assign multiple bars to the same items, you have to use different keys (Key parameter), else the default bar is overwritten. Use the Add method to add new types of bars to the Bars collection. Use the AddLink method to link a bar with another. Use the FirstVisibleDate property to specify the first visible date in the chart area. Use the Key parameter to identify a bar inside an item. If the AddBar method is called multiple time with the same item, the bar is moved. Use the ItemBar property to access a bar inside the item. Use the RemoveBar method to remove a bar from an item. Use the ClearBars method to remove all bars in the item. Use the PaneWidth property to specify the width of the chart. Use the NonworkingDays property to specify the non-working days. Use the NextDate property to compute the next or previous date based on a time unit. Use the ItemBar(exBarToolTip) property to assign a tooltip to a bar. Use the ItemBar(exBarsCount) property to count the number of bars inside in the specified item. Use the ItemBar(exBarData) property to associate an extra data to a bar in the specified item . Use the ItemBar(exBarBackColor) property to change the background or the visual appearance for the portion delimited by the start and end points. Use the AllowCreateBar property to specify whether the user can create new bars using the mouse. *Use the GroupBars method to group two or more bars, so when a bar in the group is moved or resized the other bars in the group and related are resized or moved accordingly.* The FindBar method looks for the item that hosts a specified bar.



The Color property of the Bar object specifies the color being used to paint the bar. This property changes the colors for all bars with the same name. For instance, if you have 3

"Task" bars, and you are changing the color for the "Task" bar, the color is applied to all "Task" bars in the chart. For instance, in order to provide "Task" bars with different colors, you can use the Copy method to copy the Task bar to a new bar, and use the Color to change the color of the bar. The AllowCellValueToItemBar property allows the cells to display properties of the bars.

The following function generates a Task bar with specified color:

```
Private Function AddTask(ByVal gantt As EXG2ANTTLibCtl.G2antt, ByVal clr As Long) As String
   Dim sT As String
   sT = "Task:" & clr
   With gantt.Chart.Bars.Copy("Task", sT)
      .color = clr
   End With
   AddTask = sT
End Function
```

The function generates a new bar with the name "Task:color", where the color is the color being used, and retrieves the name of the new bar being added. The Copy method retrieves the bar being found with specified name, or creates a new bar if the name is not found in the Bars collection, so AddTask function gets you the name of the bar you should use to specify the color for the bar being added as in the following sample:

```
With G2antt1.Items
   Dim d As Date
   d = G2antt1.Chart.FirstVisibleDate
   .AddBar .FirstVisibleItem, AddTask(G2antt1, vbRed), d, d + 4, "Red"
End With
```

The following VB sample adds a "Milestone" bar and a text beside:

```
With G2antt1.Items
   h = .AddItem("new task")
   .AddBar h, "Milestone", "5/30/2005 10:00", "5/31/2005"
   .AddBar h, "", "5/31/2005", "6/10/2005", "beside", "<fgcolor=FF0000><b>item</b></fgcolor> to change"
End With
```

or

```
With G2antt1.Items
   .AddBar .AddItem("new task"), "Milestone", "5/30/2005 10:00", "6/10/2005", , "
<fgcolor=FF0000><b>item</b></fgcolor> to change"
End With
```

The following VB sample adds an item with a single "Task" bar:

```
Dim h As HITEM, d As Date
With G2antt1.Items
   d = G2antt1.Chart.FirstVisibleDate
   h = .AddItem("new task")
   .AddBar h, "Task", G2antt1.Chart.NextDate(d, exDay, 2), G2antt1.Chart.NextDate(d,
exDay, 4)
End With
```

The following VB sample adds an item with three bars (two "Task" bars, and one "Split" bar) that looks like ):

```
Dim h As HITEM, d As Date
With G2antt1.Items
   d = G2antt1.Chart.FirstVisibleDate
   h = .AddItem("new task ")
   .AddBar h, "Task", d + 2, d + 4, "K1"
   .AddBar h, "Split", d + 4, d + 5, "K2"
   .AddBar h, "Task", d + 5, d + 9, "K3"
End With
```

The  bar is composed by three parts: K1, K2 and K3.

The following C++ sample adds a "Milestone" bar and a text beside:

```
#include "Items.h"
COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
CItems items = m_g2antt.GetItems();
long h = items.AddItem( COleVariant( "new task" ) );
items.AddBar( h, COleVariant("Milestone"), COleVariant( "5/30/2005 10:00" ), COleVariant(
"5/31/2005" ), vtMissing, vtMissing );
items.AddBar( h, COleVariant(""), COleVariant( "5/31/2005" ), COleVariant( "6/10/2005" ),
COleVariant( _T("just a key") ), COleVariant( "<fgcolor=FF0000><b>item</b></fgcolor>
```

```
to change" ) );
```

or

```
#include "Items.h"
COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
CItems items = m_g2antt.GetItems();
long h = items.AddItem( COleVariant( "new task" ) );
items.AddBar( h, COleVariant("Milestone"), COleVariant( "5/30/2005 10:00" ), COleVariant( "6/10/2005" ), vtMissing, COleVariant( "    <fgcolor=FF0000><b>item</b></fgcolor> to change" ) );
```

The following C++ sample adds an item with a single "Task" bar:

```
COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
CItems items = m_g2antt.GetItems();
CChart chart = m_g2antt.GetChart();
DATE d = V2D( &chart.GetFirstVisibleDate() );
long h = items.AddItem( COleVariant("new task") );
items.AddBar( h, COleVariant( "Task"), COleVariant( (double)chart.GetNextDate( d, 4096, COleVariant((long)2) ) ), COleVariant( (double)chart.GetNextDate( d, 4096, COleVariant((long)4) ) ), vtMissing , vtMissing );
```

The following C++ sample adds an item with three bars (two "Task" bars, and one "Split" bar) that looks like above:

```
COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
CItems items = m_g2antt.GetItems();
DATE d = V2D( &m_g2antt.GetChart().GetFirstVisibleDate() );
long h = items.AddItem( COleVariant("new task") );
items.AddBar( h, COleVariant( "Task"), COleVariant( d + 2 ), COleVariant( d + 4 ), COleVariant( "K1" ), vtMissing );
items.AddBar( h, COleVariant( "Split"), COleVariant( d + 4 ), COleVariant( d + 5 ), COleVariant( "K2" ), vtMissing );
items.AddBar( h, COleVariant( "Task"), COleVariant( d + 5 ), COleVariant( d + 9 ), COleVariant( "K3" ), vtMissing );
```

where the V2D function converts a Variant expression to a DATE expression and may look like follows:

```
static DATE V2D( VARIANT* pvtDate )
{
   COleVariant vtDate;
   vtDate.ChangeType( VT_DATE, pvtDate );
   return V_DATE( &vtDate );
}
```

The following VB.NET sample adds a "Milestone" bar and a text beside:

```
With AxG2antt1.Items
   Dim h As Integer = .AddItem("new task")
   .AddBar(h, "Milestone", "5/30/2005 10:00", "5/31/2005")
   .AddBar(h, "", "5/31/2005", "6/10/2005", "beside", "<fgcolor=FF0000><b>item</b>
</fgcolor> to change")
End With
```

or

```
With AxG2antt1.Items
   Dim h As Integer = .AddItem("new task")
   .AddBar(h, "Milestone", "5/30/2005 10:00", "6/10/2005", , "   <fgcolor=FF0000>
<b>item</b></fgcolor> to change")
End With
```

The following VB.NET sample adds an item with a single "Task" bar:

```
With AxG2antt1.Items
   Dim d As DateTime = AxG2antt1.Chart.FirstVisibleDate
   Dim h As Integer = .AddItem("new task")
   .AddBar(h, "Task", AxG2antt1.Chart.NextDate(d, EXG2ANTTLib.UnitEnum.exDay, 2),
AxG2antt1.Chart.NextDate(d, EXG2ANTTLib.UnitEnum.exDay, 4))
End With
```

The following VB.NET sample adds an item with three bars (two "Task" bars, and one "Split" bar) that looks like above:

```
With AxG2antt1.Items
   Dim d As DateTime = AxG2antt1.Chart.FirstVisibleDate
   Dim h As Integer = .AddItem("new task ")
```

```
    .AddBar(h, "Task", d.AddDays(2), d.AddDays(4), "K1")
    .AddBar(h, "Split", d.AddDays(4), d.AddDays(5), "K2")
    .AddBar(h, "Task", d.AddDays(5), d.AddDays(9), "K3")
End With
```

The following C# sample adds a "Milestone" bar and a text beside:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
int h = items.AddItem("new task");
items.AddBar(h, "Milestone", "5/30/2005 10:00", "5/31/2005", null, null);
items.AddBar(h, "", "5/31/2005", "6/10/2005", "just a new key", "<fgcolor=FF0000>
<b>item</b></fgcolor> to change");
```

or

```
EXG2ANTTLib.Items items = axG2antt1.Items;
int h = items.AddItem("new task");
items.AddBar(h, "Milestone", "5/30/2005 10:00", "6/10/2005", null, "    <fgcolor=FF0000>
<b>item</b></fgcolor> to change");
```

The following C# sample adds an item with a single "Task" bar:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
int h = items.AddItem("new task");
DateTime d = Convert.ToDateTime(axG2antt1.Chart.FirstVisibleDate);
items.AddBar(h, "Task", axG2antt1.Chart.get_NextDate(d, EXG2ANTTLib.UnitEnum.exDay,
2), axG2antt1.Chart.get_NextDate(d, EXG2ANTTLib.UnitEnum.exDay, 4), null, null);
```

The following C# sample adds an item with three bars (two "Task" bars, and one "Split" bar) that looks like above:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
int h = items.AddItem("new task");
DateTime d = Convert.ToDateTime( axG2antt1.Chart.FirstVisibleDate );
items.AddBar(h, "Task", d.AddDays(2), d.AddDays(4), "K1", null );
items.AddBar(h, "Split", d.AddDays(4), d.AddDays(5), "K2", null);
items.AddBar(h, "Task", d.AddDays(5), d.AddDays(9), "K3", null);
```

The following VFP sample adds an item with a single "Task" bar:

```
With thisform.G2antt1.Items
   d = thisform.G2antt1.Chart.FirstVisibleDate
   .DefaultItem = .AddItem("new task")
   .AddBar(0, "Task", thisform.G2antt1.Chart.NextDate(d,4096,2),
thisform.G2antt1.Chart.NextDate(d,4096,4))
EndWith
```

The following VFP sample adds an item with three bars (two "Task" bars, and one "Split" bar) that looks like above:

```
With thisform.G2antt1.Items
   thisform.G2antt1.Chart.FirstVisibleDate = "5/29/2005"
   .DefaultItem = .AddItem("new task")
   .AddBar(0, "Task", "5/31/2005", "6/2/2005", "K1", "")
   .AddBar(0, "Split", "6/2/2005", "6/4/2005", "K2", "")
   .AddBar(0, "Task", "6/4/2005", "6/9/2005", "K3", "")
EndWith
```

method Items.AddItem ([Caption as Variant])

Adds a new item, and returns a handle to the newly created item.

Type	Description
Caption as Variant	A string expression that indicates the cell's caption for the first column. or a safe array that contains the captions for each column. The Caption accepts HTML format, if the CellValueFormat property is exHTML, or a formula if the CellValueFormat property is exComputedField.
Return	**Description**
HITEM	A long expression that indicates the handle of the newly created item.

Use the Add method to add new columns to the control. If the control contains no columns, the AddItem method fails. Use the LoadXML/SaveXML methods to load/save the control's data from/to XML files. Use the AddItem property to add new items to the control. Use the AddBar method to add bars to the item. Use the AddLink method to link a bar with another. The bars are always shown in the chart area. Use the PaneWidth property to specify the width of the chart. Use InsertItem method to insert child items to the list. Use the InsertControlItem property to insert and ActiveX control. Use the LockedItemCount property to add or remove items locked to the top or bottom side of the control. Use the MergeCells method to combine two or multiple cells in a single cell. Use the SplitCell property to split a cell. Use the BeginUpdate and EndUpdate methods to maintain performance while adding new columns and items. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula. Use the FormatColumn property to format the column.

The AddItem property adds a new item that has no parent. When a new item is added (inserted) to the Items collection, the control fires the AddItem event. If the control contains more than one column use the CellValue property to set the cell's caption. If there are no columns AddItem method fails.

The following VB6 sample uses the VB Array function to add two items:

```
With G2antt1
  .BeginUpdate

    .Columns.Add "Column 1"
    .Columns.Add "Column 2"
    .Columns.Add "Column 3"
```

```
    With .Items
       .AddItem Array("Item 1.1", "Item 1.2", "Item 1.3")
       .AddItem Array("Item 2.1", "Item 2.2", "Item 2.3")
    End With

  .EndUpdate
End With
```

In VB/NET using the /NET assembly, the Array equivalent is New Object such as follows:

```
With G2antt1
  .BeginUpdate()

  .Columns.Add("Column 1")
  .Columns.Add("Column 2")
  .Columns.Add("Column 3")

  With .Items
     .AddItem(New Object() {"Item 1.1", "Item 1.2", "Item 1.3"})
     .AddItem(New Object() {"Item 2.1", "Item 2.2", "Item 2.3"})
  End With

  .EndUpdate()
End With
```

In C# using the /NET assembly, the Array equivalent is new object such as follows:

```
exg2antt1.BeginUpdate();

exg2antt1.Columns.Add("Column 1");
exg2antt1.Columns.Add("Column 2");
exg2antt1.Columns.Add("Column 3");

exg2antt1.Items.AddItem(new object[] { "Item 1.1", "Item 1.2", "Item 1.3" });
exg2antt1.Items.AddItem(new object[] { "Item 2.1", "Item 2.2", "Item 2.3" });

exg2antt1.EndUpdate();
```

Use the PutItems method to load an array, like in the following VB sample:

```
Set rs = CreateObject("ADODB.Recordset")
rs.Open "Orders", "Provider=Microsoft.Jet.OLEDB.3.51;Data Source= D:\Program
Files\Microsoft Visual Studio\VB98\NWIND.MDB", 3 ' Opens the table using static mode
G2antt1.BeginUpdate
' Add the columns
With G2antt1.Columns
For Each f In rs.Fields
   .Add f.Name
Next
End With
G2antt1.PutItems rs.getRows()
G2antt1.EndUpdate
```

The following C++ sample adds new items to the control:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
long iNewItem = items.AddItem( COleVariant( "Item 1" ) );
items.SetCellValue( COleVariant( iNewItem ), COleVariant( (long)1 ), COleVariant( "SubItem
1" ) );
iNewItem = items.AddItem( COleVariant( "Item 2" ) );
items.SetCellValue( COleVariant( iNewItem ), COleVariant( (long)1 ), COleVariant( "SubItem
2" ) );
```

The following VB.NET sample adds new items to the control:

```
With AxG2antt1.Items
   Dim iNewItem As Integer
   iNewItem = .AddItem("Item 1")
   .CellValue(iNewItem, 1) = "SubItem 1"
   iNewItem = .AddItem("Item 2")
   .CellValue(iNewItem, 1) = "SubItem 2"
End With
```

The following C# sample adds new items to the control:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
```

```
int iNewItem = items.AddItem( "Item 1" );
items.set_CellValue( iNewItem, 1, "SubItem 1" );
iNewItem = items.AddItem( "Item 2" );
items.set_CellValue( iNewItem, 1, "SubItem 2" );
```

The following VFP sample adds new items to the control:

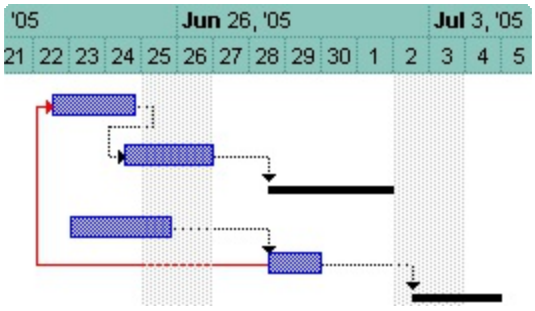
```
with thisform.G2antt1.Items
    .DefaultItem = .AddItem("Item 1")
    .CellValue(0, 1) = "SubItem 1"
endwith
```

method Items.AddLink (LinkKey as Variant, StartItem as HITEM, StartBarKey as Variant, EndItem as HITEM, EndBarKey as Variant)

Links a bar to another.

Type	Description
LinkKey as Variant	A String expression that indicates the key of the link. This value is used to identify the link.
StartItem as HITEM	A HITEM expression that indicates the handle of the item where the link starts.
StartBarKey as Variant	A String expression that indicates the key of the bar in the StartItem where the link starts.
EndItem as HITEM	A HITEM expression that indicates the handle of the item where the link ends.
EndBarKey as Variant	A String expression that indicates the key of the bar in the EndItem where the link ends.

Use the AddLink method to draw a line between two bars, or two create a link between two bars. The AddLink method adds the link not matter of the ItemBar(exBarCanBeLinked), ItemBar(exBarCanStartLink), ItemBar(exBarCanEndLink) properties. Use the Link(exLinkGroupBars) to group two linked bars. Use the GroupBars method to group two bars so you can move or resize together when a change occurs in the group. Use the AllowLinkBars property to specify whether the user can link bars using the mouse. By default, the bar is drawn from the right side of the starting bar, to the left side of the ending bar. Use the Link(exLinkText) property to display a HTML text/icon or picture on the link. Use the Link(exLinkStartPos) property to change where the link starts in the starting bar. Use the Link(exLinkEndPos) property to change where the link starts in the starting bar. Use the AddBar method to add new bars to an item. Use the Link property to change the appearance of the line between bars. Use the ShowLinks property to hide all links in the chart area. Use the ClearLinks method to clear the links collection. The AddLink method fails, if the StartItem or EndItem item is not valid, or if the StartBarKey or EndBarKey bar does not exist. Use the LinkColor property to change the color for all links between bars. Use the Link(exLinkShowDir) property to hide the link's arrow. Use the RemoveLink method to remove a specific link. Use the BeginUpdate and EndUpdate methods to maintain performance while adding columns, items, bars or links. Use the FirstLink and NextLink properties to enumerate the links in the control.



The following VB sample adds a link between two bars:

```
G2antt1.BeginUpdate
With G2antt1.Items
   Dim h1 As HITEM
   h1 = .AddItem("Item 1")
   .AddBar h1, "Task", G2antt1.Chart.FirstVisibleDate + 2, G2antt1.Chart.FirstVisibleDate +
4
   Dim h2 As HITEM
   h2 = .AddItem("Item 2")
   .AddBar h2, "Task", G2antt1.Chart.FirstVisibleDate + 1, G2antt1.Chart.FirstVisibleDate +
2, "A"
   .AddLink "Link11", h1, "", h2, "A"
End With
G2antt1.EndUpdate
```

The following C++ sample adds a link between two bars:

```
COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
m_g2antt.BeginUpdate();
CItems items = m_g2antt.GetItems();
CChart chart = m_g2antt.GetChart();
long h1 = items.AddItem( COleVariant( "Item1" ) );
items.AddBar( h1, COleVariant( "Task" ), COleVariant( V_DATE(&chart.GetFirstVisibleDate())
+ 2 ), COleVariant( V_DATE(&chart.GetFirstVisibleDate()) + 4 ), vtMissing, vtMissing );
long h2 = items.AddItem( COleVariant( "Item2" ) );
items.AddBar( h2, COleVariant( "Task" ), COleVariant( V_DATE(&chart.GetFirstVisibleDate())
+ 1 ), COleVariant( V_DATE(&chart.GetFirstVisibleDate()) + 2 ), COleVariant("JustAKey"),
vtMissing );
items.AddLink( COleVariant( "Link1" ), h1, vtMissing, h2, COleVariant("JustAKey") );
m_g2antt.EndUpdate();
```

The following VB.NET sample adds a link between two bars:

```
AxG2antt1.BeginUpdate()
Dim d As Date = AxG2antt1.Chart.FirstVisibleDate
With AxG2antt1.Items
   Dim h1 As Integer = .AddItem("Item 1")
   .AddBar(h1, "Task", d.AddDays(2), d.AddDays(4))
   Dim h2 As Integer = .AddItem("Item 2")
   .AddBar(h2, "Task", d.AddDays(1), d.AddDays(2), "A")
   .AddLink("Link11", h1, "", h2, "A")
End With
AxG2antt1.EndUpdate()
```

The following C# sample adds a link between two bars:

```
axG2antt1.BeginUpdate();
DateTime d = Convert.ToDateTime(axG2antt1.Chart.FirstVisibleDate);
EXG2ANTTLib.Items spItems = axG2antt1.Items;
int h1 = spItems.AddItem("Item 1");
spItems.AddBar(h1, "Task", d.AddDays(2), d.AddDays(4) , null, null);
int h2 = spItems.AddItem("Item 2");
spItems.AddBar(h2, "Task", d.AddDays(1), d.AddDays(2), "A", null);
spItems.AddLink("Link1", h1, null, h2, "A");
axG2antt1.EndUpdate();
```

The following VFP sample adds a link between two bars:

```
thisform.G2antt1.BeginUpdate
local d
d = thisform.G2antt1.Chart.FirstVisibleDate
With thisform.G2antt1.Items
   local h1
   .DefaultItem = .AddItem("Item 1")
   h1 = .DefaultItem
   .AddBar(0, "Task", thisform.G2antt1.Chart.NextDate(d,4096,2),
thisform.G2antt1.Chart.NextDate(d,4096,4))
   local h2
   .DefaultItem = .AddItem("Item 2")
```

```
    h2 = .DefaultItem
    .AddBar(0, "Task", thisform.G2antt1.Chart.NextDate(d,4096,1),
thisform.G2antt1.Chart.NextDate(d,4096,2), "A")
    .AddLink("Link11", h1, "", h2, "A")
EndWith
thisform.G2antt1.EndUpdate
```

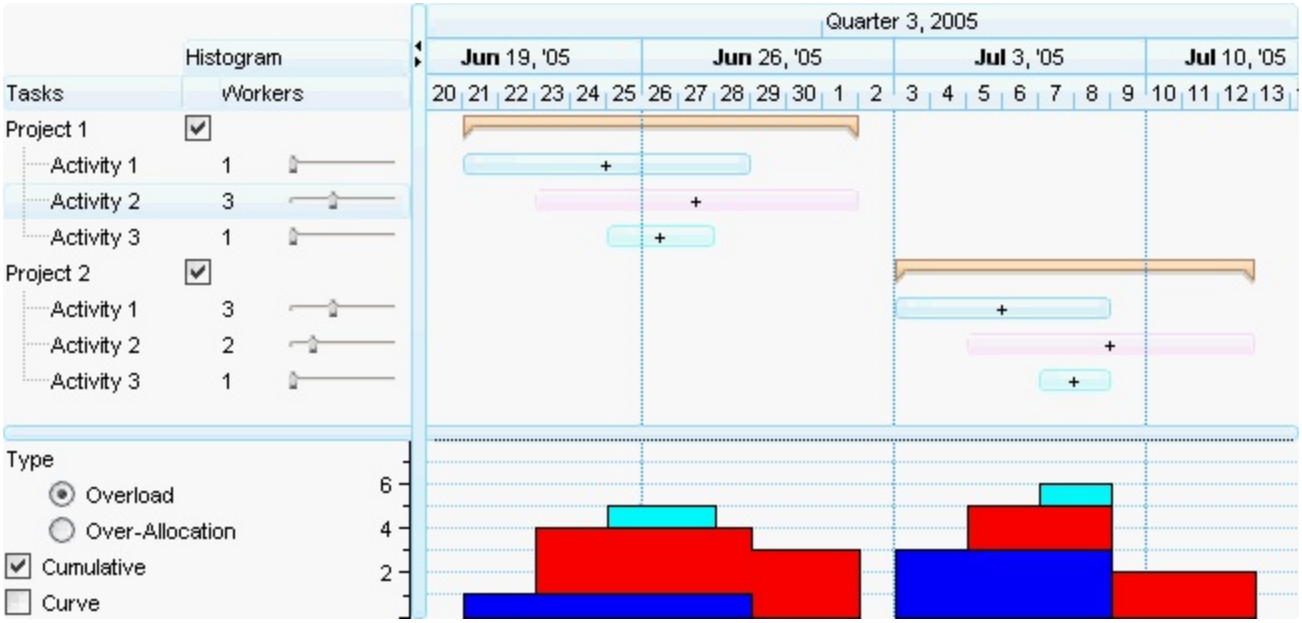
property Items.AllowCellValueToItemBar as Boolean

Retrieves or sets a value that indicates whether the cells display associated properties of the bars in the item.

Type	Description
Boolean	A Boolean expression that specifies whether the control handles properties of the bars in the column's section.

By default, the AllowCellValueToItemBar property is False. In other words, if there is any association between a cell and a property bar they do not have any effect, while the AllowCellValueToItemBar property is False. The AllowCellValueToItemBar property allows the cells to display properties of the bars, and the property of the bar to be specified by the cell's value. For instance, you can use the CellValueToItemBar feature to display the start and ending date of any bar in the item, for the entire column, or in any cell. The values in the cells are automatically changed once the bar is resized or moved and reverse, if the cell's value is changed the associated bar is moved or resized.

The following screen shot shows the association between ItemBar(exBarEffort) property of the bar with the cell in last column, so changing it is automatically reflected in the chart's histogram :



The cell's value can be associated with any property of the bar, using the:

- Def(exCellValueToItemBarProperty/exCellValueToItemBarKey) property of the Column object, that defines a relation/association between specified property bar and the cells in the column. For instance, the .Def(exCellValueToItemBarProperty) = 1 indicates that the column displays the property of the bar with the index 1, which is exBarStart or .Def(exCellValueToItemBarProperty) = 543, displays the exBarEndInclusive property of the bar.

- CellValueToItemBar method of the Items object, that associates the cell's value with any property of the bar in the item.

Once an association between a cell and a bar is made, the CellValue property and ItemBar property returns the same result, or in other words, changing the cell's value will be reflected in the bar's property, and back, so changing the bar's property will change the cell's value.

The following VB sample display automatically the start and end dates of the bars in the Start and End columns:

```
With G2antt1
   .BeginUpdate
   With .Columns
      .Add "Tasks"
      With .Add("Start")
         .Def(exCellValueToItemBarProperty) = 1
         .Editor.EditType = DateType
      End With
      With .Add("End")
         .Def(exCellValueToItemBarProperty) = 2
         .Editor.EditType = DateType
      End With
   End With
   With .Chart
      .FirstVisibleDate = #9/20/2006#
      .AllowLinkBars = True
      .AllowCreateBar = exNoCreateBar
      .LevelCount = 2
      .PaneWidth(0) = 196
   End With
   With .Items
      .AllowCellValueToItemBar = True
      .AddBar .AddItem("Task 1"),"Task",#9/21/2006#,#9/24/2006#
      .AddBar .AddItem("Task 2"),"Task",#9/22/2006#,#9/25/2006#
      .AddBar .AddItem("Task 3"),"Task",#9/23/2006#,#9/26/2006#
   End With
   .EndUpdate
End With
```

The following VB.NET sample display automatically the start and end dates of the bars in the Start and End columns:

```
With AxG2antt1
   .BeginUpdate
   With .Columns
      .Add "Tasks"
      With .Add("Start")
         .Def(EXG2ANTTLib.DefColumnEnum.exCellValueToItemBarProperty) = 1
         .Editor.EditType = EXG2ANTTLib.EditTypeEnum.DateType
      End With
      With .Add("End")
         .Def(EXG2ANTTLib.DefColumnEnum.exCellValueToItemBarProperty) = 2
         .Editor.EditType = EXG2ANTTLib.EditTypeEnum.DateType
      End With
   End With
   With .Chart
      .FirstVisibleDate = #9/20/2006#
      .AllowLinkBars = True
      .AllowCreateBar = EXG2ANTTLib.CreateBarEnum.exNoCreateBar
      .LevelCount = 2
      .PaneWidth(0) = 196
   End With
   With .Items
      .AllowCellValueToItemBar = True
      .AddBar .AddItem("Task 1"),"Task",#9/21/2006#,#9/24/2006#
      .AddBar .AddItem("Task 2"),"Task",#9/22/2006#,#9/25/2006#
      .AddBar .AddItem("Task 3"),"Task",#9/23/2006#,#9/26/2006#
   End With
   .EndUpdate
End With
```

The following C++ sample display automatically the start and end dates of the bars in the Start and End columns:

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'
```

```
   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
EXG2ANTTLib::IColumnsPtr var_Columns = spG2antt1->GetColumns();
   var_Columns->Add(L"Tasks");
   EXG2ANTTLib::IColumnPtr var_Column = ((EXG2ANTTLib::IColumnPtr)(var_Columns-
>Add(L"Start")));
      var_Column->PutDef(EXG2ANTTLib::exCellValueToItemBarProperty,long(1));
      var_Column->GetEditor()->PutEditType(EXG2ANTTLib::DateType);
   EXG2ANTTLib::IColumnPtr var_Column1 = ((EXG2ANTTLib::IColumnPtr)(var_Columns-
>Add(L"End")));
      var_Column1->PutDef(EXG2ANTTLib::exCellValueToItemBarProperty,long(2));
      var_Column1->GetEditor()->PutEditType(EXG2ANTTLib::DateType);
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutFirstVisibleDate("9/20/2006");
   var_Chart->PutAllowLinkBars(VARIANT_TRUE);
   var_Chart->PutAllowCreateBar(EXG2ANTTLib::exNoCreateBar);
   var_Chart->PutLevelCount(2);
   var_Chart->PutPaneWidth(0,196);
EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems();
   var_Items->PutAllowCellValueToItemBar(VARIANT_TRUE);
   var_Items->AddBar(var_Items->AddItem("Task
1"),"Task","9/21/2006","9/24/2006",vtMissing,vtMissing);
   var_Items->AddBar(var_Items->AddItem("Task
2"),"Task","9/22/2006","9/25/2006",vtMissing,vtMissing);
   var_Items->AddBar(var_Items->AddItem("Task
3"),"Task","9/23/2006","9/26/2006",vtMissing,vtMissing);
spG2antt1->EndUpdate();
```

The following C# sample display automatically the start and end dates of the bars in the Start and End columns:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Columns var_Columns = axG2antt1.Columns;
```

```
   var_Columns.Add("Tasks");
   EXG2ANTTLib.Column var_Column = (var_Columns.Add("Start") as
EXG2ANTTLib.Column);

var_Column.set_Def(EXG2ANTTLib.DefColumnEnum.exCellValueToItemBarProperty,1);
      var_Column.Editor.EditType = EXG2ANTTLib.EditTypeEnum.DateType;
   EXG2ANTTLib.Column var_Column1 = (var_Columns.Add("End") as
EXG2ANTTLib.Column);

var_Column1.set_Def(EXG2ANTTLib.DefColumnEnum.exCellValueToItemBarProperty,2);
      var_Column1.Editor.EditType = EXG2ANTTLib.EditTypeEnum.DateType;
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.FirstVisibleDate = "9/20/2006";
   var_Chart.AllowLinkBars = true;
   var_Chart.AllowCreateBar = EXG2ANTTLib.CreateBarEnum.exNoCreateBar;
   var_Chart.LevelCount = 2;
   var_Chart.set_PaneWidth(0 != 0,196);
EXG2ANTTLib.Items var_Items = axG2antt1.Items;
   var_Items.AllowCellValueToItemBar = true;
   var_Items.AddBar(var_Items.AddItem("Task 1"),"Task","9/21/2006","9/24/2006",null,null);
   var_Items.AddBar(var_Items.AddItem("Task 2"),"Task","9/22/2006","9/25/2006",null,null);
   var_Items.AddBar(var_Items.AddItem("Task 3"),"Task","9/23/2006","9/26/2006",null,null);
axG2antt1.EndUpdate();
```

The following VFP sample display automatically the start and end dates of the bars in the Start and End columns:

```
with thisform.G2antt1
   .BeginUpdate
   with .Columns
      .Add("Tasks")
      with .Add("Start")
         .Def(18) = 1
         .Editor.EditType = 7
      endwith
      with .Add("End")
         .Def(18) = 2
         .Editor.EditType = 7
```

```
    endwith
  endwith
  with .Chart
    .FirstVisibleDate = {^2006-9-20}
    .AllowLinkBars = .T.
    .AllowCreateBar = 0
    .LevelCount = 2
    .PaneWidth(0) = 196
  endwith
  with .Items
    .AllowCellValueToItemBar = .T.
    .AddBar(.AddItem("Task 1"),"Task",{^2006-9-21},{^2006-9-24})
    .AddBar(.AddItem("Task 2"),"Task",{^2006-9-22},{^2006-9-25})
    .AddBar(.AddItem("Task 3"),"Task",{^2006-9-23},{^2006-9-26})
  endwith
  .EndUpdate
endwith
```

The following Delphi sample display automatically the start and end dates of the bars in the Start and End columns:

```
with AxG2antt1 do
begin
  BeginUpdate();
  with Columns do
  begin
    Add('Tasks');
    with (Add('Start') as EXG2ANTTLib.Column) do
    begin
      Def[EXG2ANTTLib.DefColumnEnum.exCellValueToItemBarProperty] := TObject(1);
      Editor.EditType := EXG2ANTTLib.EditTypeEnum.DateType;
    end;
    with (Add('End') as EXG2ANTTLib.Column) do
    begin
      Def[EXG2ANTTLib.DefColumnEnum.exCellValueToItemBarProperty] := TObject(2);
      Editor.EditType := EXG2ANTTLib.EditTypeEnum.DateType;
    end;
  end;
```

```
  with Chart do
  begin
    FirstVisibleDate := '9/20/2006';
    AllowLinkBars := True;
    AllowCreateBar := EXG2ANTTLib.CreateBarEnum.exNoCreateBar;
    LevelCount := 2;
    PaneWidth[0 <> 0] := 196;
  end;
  with Items do
  begin
    AllowCellValueToItemBar := True;
    AddBar(AddItem('Task 1'),'Task','9/21/2006','9/24/2006',Nil,Nil);
    AddBar(AddItem('Task 2'),'Task','9/22/2006','9/25/2006',Nil,Nil);
    AddBar(AddItem('Task 3'),'Task','9/23/2006','9/26/2006',Nil,Nil);
  end;
  EndUpdate();
end
```

property Items.CellBackColor([Item as Variant], [ColIndex as Variant]) as Color

Retrieves or sets the cell's background color.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
Color	A color expression that indicates the cell's background color. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

To change the background color for the entire item you can use ItemBackColor property. Use the ClearCellBackColor method to clear the cell's background color. Use the BackColor property to specify the control's background color. Use the CellForeColor property to specify the cell's foreground color. Use the ItemForeColor property to specify the item's foreground color. Use the Def(exCellBackColor) property to specify the background color for all cells in the column. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula.

In VB.NET or C# you require the following functions until the .NET framework will support them:

You can use the following VB.NET function:

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
   Dim i As Long
i = c.R
i = i + 256 * c.G
i = i + 256 * 256 * c.B
ToUInt32 = Convert.ToUInt32(i)
End Function
```

You can use the following C# function:

```
private UInt32 ToUInt32(Color c)
{
   long i;
   i = c.R;
   i = i + 256 * c.G;
   i = i + 256 * 256 * c.B;
   return Convert.ToUInt32(i);
}
```

The following C# sample changes the background color for the focused cell:

```
axG2antt1.Items.set_CellBackColor(axG2antt1.Items.FocusItem, 0, ToUInt32(Color.Red));
```

The following VB.NET sample changes the background color for the focused cell:

```
With AxG2antt1.Items
   .CellBackColor(.FocusItem, 0) = ToUInt32(Color.Red)
End With
```

The following C++ sample changes the background color for the focused cell:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetCellBackColor( COleVariant( items.GetFocusItem() ), COleVariant( (long)0 ),
RGB(255,0,0) );
```

The following VFP sample changes the background color for the focused cell:

```
with thisform.G2antt1.Items
   .DefaultItem = .FocusItem
   .CellBackColor( 0, 0 ) = RGB(255,0,0)
endwith
```

For instance, the following VB code changes background color of the left top cell of your control: G2antt1.Items.CellBackColor(G2antt.Items(0), 0) = vbBlue

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell (see ItemCell property). Here's few hints

how to use properties with Item and ColIndex parameters:

G2antt1.Items.CellBold(, G2antt1.Items.ItemCell(G2antt1.Items(0), 0)) = True

G2antt1.Items.CellBold(G2antt1.Items(0), 0) = True

G2antt1.Items.CellBold(G2antt1.Items(0), "ColumnName") = True

property Items.CellBold([Item as Variant], [ColIndex as Variant]) as Boolean

Retrieves or sets a value that indicates whether the cell's caption should appear in bold.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
Boolean	A boolean expression that indicates whether the cell should appear in bold.

Use the CellBold property to bold a cell. Use the ItemBold property to specify whether the item should appear in bold. Use the HeaderBold property of the Column object to bold the column's caption. Use the CellItalic, CellUnderline or CellStrikeOut property to apply different font attributes to the cell. Use the ItemItalic, ItemUnderline or ItemStrikeOut property to apply different font attributes to the item. Use the CellValueFormat property to specify an HTML caption. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula.

The following VB sample bolds the cells in the first column

```
Dim h As Variant
G2antt1.BeginUpdate
With G2antt1.Items
For Each h In G2antt1.Items
   .CellBold(h, 0) = True
Next
End With
G2antt1.EndUpdate
```

The following C++ sample bolds the focused cell:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetCellBold( COleVariant( items.GetFocusItem() ), COleVariant( (long)0 ), TRUE );
```

The following C# sample bolds the focused cell:

```
axG2antt1.Items.set_CellBold(axG2antt1.Items.FocusItem, 0, true);
```

The following VB.NET sample bolds the focused cell:

```
With AxG2antt1.Items
   .CellBold(.FocusItem, 0) = True
End With
```

The following VFP sample bolds the focused cell:

```
with thisform.G2antt1.Items
   .DefaultItem = .FocusItem
   .CellBold( 0, 0 ) = .t.
endwith
```

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell (see ItemCell property). Here's few hints how to use properties with Item and ColIndex parameters:

G2antt1.Items.CellBold(, G2antt1.Items.ItemCell(G2antt1.Items(0), 0)) = True

G2antt1.Items.CellBold(G2antt1.Items(0), 0) = True

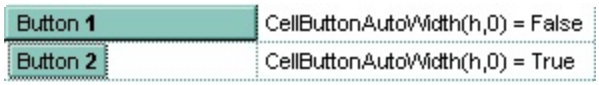
G2antt1.Items.CellBold(G2antt1.Items(0), "ColumnName") = True

property Items.CellButtonAutoWidth([Item as Variant], [ColIndex as Variant]) as Boolean

Retrieves or sets a value indicating whether the cell's button fits the cell's caption.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, or a string expression that indicates the column's caption or the column's key.
Boolean	A boolean expression indicating whether the cell's button fits the cell's caption.

By default, the CellButtonAutoWidth property is False. The CellButtonAutoWidth property has effect only if the CellHasButton property is true. Use the Def property to specify that all buttons in the column fit to the cell's content. If the CellButtonAutoWidth property is False, the width of the button is the same as the width of the column. If the CellButtonAutoWidth property is True, the button area covers only the cell's caption. Use the CellValue property to specify the button's caption. Use the CellValueFormat property to assign an HTML caption to the button. The control fires the ButtonClick property when the user clicks a button.



property Items.CellCaption ([Item as Variant], [ColIndex as Variant]) as String

Gets the cell's display value.

Type	Description
Item as Variant	A long expression that indicates the item's handle. During the ValidateValue event, you can uses -1 instead Item, to access to the modified value. In other words during ValidateValue event, the Items.CellValue(Item,ColIndex) and Items.CellCaption(Item,ColIndex) properties retrieve the original value/caption of the cell while the Items.CellValue(-1,ColIndex) and Items.CellCaption(-1,ColIndex) gets the modified value of the specified cell.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
String	A string expression that indicates the cell's value as it is displayed on the user interface.

The CellCaption property retrieves the cell's display value as it is displayed on the control's user interface. If the cell has no editor associated (no editor was assigned to the column and no editor was assigned to the cell), the CellCaption property gets the string representation of the cell's value. Use the CellValue property to change the cell's value. For instance, if a cell has a drop down list editor, the CellCaption property retrieves the caption of the predefined values. Use the CellImage property to assign an icon to a cell. Use the CellImages property to assign multiple icons to a cell. Use the CellPicture property to assign a custom size picture to a cell. Use the **<img>** HTML tag to insert icons inside the cell's caption, if the CellValueFormat property is exHTML.

property Items.CellChecked (RadioGroup as Long) as HCELL

Retrieves the cell's handle that is checked on a specific radio group.

Type	Description
RadioGroup as Long	A long expression that indicates the radio group identifier.
HCELL	A long expression that identifies the handle of the cell that's checked in the specified radio group. To retrieve the handle of the owner item you have to use CellItem property.

A radio group contains a set of cells of radio types. Use the CellHasRadioButton property to set the cell of radio type. To change the state for a cell you can use the CellState property. To add or remove a cell to a given radio group you have to use CellHasRadioButton property. Use the CellRadioGroup property to add cells in the same radio group. The control fires the CellStateChanged event when the check box or radio button state is changed.

The following VB sample groups all cells on the first column into a radio group, and display the cell's checked on the radio group when the state of a radio group is changed:

```
Private Sub G2antt1_AddItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
   G2antt1.Items.CellHasRadioButton(Item, 0) = True
   G2antt1.Items.CellRadioGroup(Item, 0) = 1234 ' The 1234 is arbirary and it represents the identifier for the radio group
End Sub

Private Sub G2antt1_CellStateChanged(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal ColIndex As Long)
   Debug.Print "In the 1234 radio group the """ & G2antt1.Items.CellValue(, G2antt1.Items.CellChecked(1234)) & """ is checked."
End Sub
```

The following C++ sample groups the radio cells on the first column, and displays the caption of the checked radio cell:

```
#include "Items.h"
COleVariant vtColumn( long(0) );
CItems items = m_g2antt.GetItems();
m_g2antt.BeginUpdate();
```

```
for ( long i = 0; i < items.GetItemCount(); i++ )
{
   COleVariant vtItem( items.GetItemByIndex( i ) );
   items.SetCellHasRadioButton( vtItem, vtColumn, TRUE );
   items.SetCellRadioGroup( vtItem, vtColumn, 1234 );
}
m_g2antt.EndUpdate();
```

```
static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
   if ( pv )
   {
      if ( pv->vt == VT_ERROR )
         return szDefault;

      COleVariant vt;
      vt.ChangeType( VT_BSTR, pv );
      return V_BSTR( &vt );
   }
   return szDefault;
}

void OnCellStateChangedG2antt1(long Item, long ColIndex)
{
   CItems items = m_g2antt.GetItems();
   long hCell = items.GetCellChecked( 1234 );
   COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
   OutputDebugString( V2S( &items.GetCellValue( vtMissing, COleVariant( hCell ) ) ) );
}
```

The following VB.NET sample groups the radio cells on the first column, and displays the caption of the checked radio cell:

```
With AxG2antt1
   .BeginUpdate()
   With .Items
      Dim k As Integer
      For k = 0 To .ItemCount - 1
```

```
        .CellHasRadioButton(.ItemByIndex(k), 0) = True
        .CellRadioGroup(.ItemByIndex(k), 0) = 1234
      Next
    End With
    .EndUpdate()
End With
```

```
Private Sub AxG2antt1_CellStateChanged(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_CellStateChangedEvent) Handles
AxG2antt1.CellStateChanged
    With AxG2antt1.Items
      Debug.WriteLine(.CellValue(, .CellChecked(1234)))
    End With
End Sub
```

The following C# sample groups the radio cells on the first column, and displays the caption of the checked radio cell:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Items items = axG2antt1.Items;
for (int i = 0; i < items.ItemCount; i++)
{
    items.set_CellHasRadioButton(items[i], 0, true);
    items.set_CellRadioGroup(items[i], 0, 1234);
}
axG2antt1.EndUpdate();
```

```
private void axG2antt1_CellStateChanged(object sender,
AxEXG2ANTTLib._IG2anttEvents_CellStateChangedEvent e)
{
    string strOutput = axG2antt1.Items.get_CellValue( 0,
axG2antt1.Items.get_CellChecked(1234) ).ToString();
    strOutput += " state = " + axG2antt1.Items.get_CellState(e.item, e.colIndex).ToString() ;
    System.Diagnostics.Debug.WriteLine( strOutput );
}
```

The following VFP sample groups the radio cells on the first column, and displays the caption of the checked radio cell:

```
thisform.G2antt1.BeginUpdate()
with thisform.G2antt1.Items
   local i
   for i = 0 to .ItemCount - 1
      .DefaultItem = .ItemByIndex(i)
      .CellHasRadioButton( 0,0 ) = .t.
      .CellRadioGroup(0,0) = 1234
   next
endwith
thisform.G2antt1.EndUpdate()
```

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell (see ItemCell property). Here's few hints how to use properties with Item and ColIndex parameters:

```
G2antt1.Items.CellBold(, G2antt1.Items.ItemCell(G2antt1.Items(0), 0)) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), 0) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), "ColumnName") = True
```

property Items.CellData([Item as Variant], [ColIndex as Variant]) as Variant

Retrieves or sets the extra data for a specific cell.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
Variant	A variant expression that indicates the cell's user data.

Use the CellData to associate an extra data to your cell. Use ItemData when you need to associate an extra data with an item. The CellData value is not used by the control, it is only for user use. Use the Data property to assign an extra data to a column. Use the SortUserData or SortUserDataString type to sort the column based on the CellData value. Use the CellValue property to specify the cell's caption.

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell (see ItemCell property). Here's few hints how to use properties with Item and ColIndex parameters:

```
G2antt1.Items.CellBold(, G2antt1.Items.ItemCell(G2antt1.Items(0), 0)) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), 0) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), "ColumnName") = True
```

property Items.CellEditor ([Item as Variant], [ColIndex as Variant]) as Editor

Creates an gets the cell's built-in editor.

Type	Description
Item as Variant	A long expression that indicates the item's handle
ColIndex as Variant	A long expression that indicates the column's index or the cell's handle, a string expression that indicates the column's caption or key.
Editor	An Editor object being created or accessed

The CellEditor property gets you the custom cell editor if it exists, **else it creates it**. The CellEditor property **creates an empty editor if it wasn't created before**, so please pay attention when creating custom cell editors on the fly. You can use the HasCellEditor property to check whether a cell has associated a custom editor (created using the CellEditor property). You can have different type of editors in the same column using the CellEditor property. The CellEditor property builds a new editor for a specific cell. By default, the cell's editor is the default column's editor. Use the EditType property to specify an editor for the column. Use the DeleteCellEditor method to clear a particular cell editor created using the CellEditor property. Use the CellEditorVisible property to hide the cell's editor. You can use the BeginUpdate, EndUpdate method to refresh the control.

The following VB sample assigns a date type editor to the first cell:

```
With G2antt1.Items
  Dim h As EXG2ANTTLibCtl.HITEM
  h = .ItemByIndex(0)
  If Not .HasCellEditor(h, 0) Then
    With .CellEditor(h, 0)
      .EditType = DateType
    End With
  End If
End With
```

The following VB sample assigns a spin type editor with min and max values:

```
With G2antt1.Items
  With .CellEditor(.FirstVisibleItem, 0)
    .EditType = SliderType
```

```
    .Option(exSliderWidth) = 0
    .Option(exSliderMin) = 5
    .Option(exSliderMax) = 10
  End With
End With
```

The following C++ sample assigns a date type editor to the first cell:

```
#include "Items.h"
#include "Editor.h"
void OnButton1()
{
  CItems items = m_g2antt.GetItems();
  COleVariant vtItem( items.GetItemByIndex( 0 ) ), vtColumn( (long)0 );
  if ( !items.GetHasCellEditor( vtItem, vtColumn ) )
  {
    CEditor editor = items.GetCellEditor( vtItem, vtColumn );
    editor.SetEditType( 7 /*DateType*/ );
  }
}
```

The following C++ sample assigns a spin type editor with min and max values:

```
#include "Items.h"
#include "Editor.h"
CItems items = m_g2antt.GetItems();
CEditor editor = items.GetCellEditor( COleVariant( items.GetFirstVisibleItem() ),
COleVariant( long(0) ) );
editor.SetEditType( 20 /*SliderType*/ );
editor.SetOption( 41 /*exSliderWidth */, COleVariant( long(0) ) );
editor.SetOption( 43 /*exSliderMin*/, COleVariant( long(5) ) );
editor.SetOption( 44 /*exSliderMax*/, COleVariant( long(10) ) );
```

The following VB.NET sample assigns a date type editor to the first cell:

```
With AxG2antt1.Items
  Dim h As Integer = .ItemByIndex(0)
  If Not .HasCellEditor(h, 0) Then
    With .CellEditor(h, 0)
```

```
        .EditType = EXG2ANTTLib.EditTypeEnum.DateType
    End With
  End If
End With
```

The following VB.NET sample assigns a spin type editor with min and max values:

```
With AxG2antt1.Items
  With .CellEditor(.FirstVisibleItem, 0)
    .EditType = EXG2ANTTLib.EditTypeEnum.SliderType
    .Option(EXG2ANTTLib.EditorOptionEnum.exSliderWidth) = 0
    .Option(EXG2ANTTLib.EditorOptionEnum.exSliderMin) = 5
    .Option(EXG2ANTTLib.EditorOptionEnum.exSliderMax) = 10
  End With
End With
```

The following C# sample assigns a date type editor to the first cell:

```
int h = axG2antt1.Items[0];
if ( !axG2antt1.Items.get_HasCellEditor( h, 0 ) )
{
  EXG2ANTTLib.Editor editor = axG2antt1.Items.get_CellEditor( h, 0 );
  if ( editor != null )
    editor.EditType = EXG2ANTTLib.EditTypeEnum.DateType;
}
```

The following C# sample assigns a spin type editor with min and max values:

```
EXG2ANTTLib.Editor editor =
axG2antt1.Items.get_CellEditor(axG2antt1.Items.FirstVisibleItem, 0);
editor.EditType = EXG2ANTTLib.EditTypeEnum.SliderType;
editor.set_Option(EXG2ANTTLib.EditorOptionEnum.exSliderWidth, 0);
editor.set_Option(EXG2ANTTLib.EditorOptionEnum.exSliderMin, 5);
editor.set_Option(EXG2ANTTLib.EditorOptionEnum.exSliderMax, 10);
```

The following VFP sample assigns a date type editor to the first cell:

```
with thisform.G2antt1.Items
  thisform.G2antt1.BeginUpdate()
```

```
    .DefaultItem = .ItemByIndex( 0 )
    if ( !.HasCellEditor( 0, 0 ) )
        .CellEditor( 0, 0 ).EditType = 7
    endif
    thisform.G2antt1.EndUpdate()
endwith
```

The following VFP sample assigns a spin type editor with min and max values:

```
with thisform.G2antt1.Items
    with .CellEditor(.FirstVisibleItem, 0)
        .EditType = 20 && SliderType
        .Option(41) = 0  && exSliderWidth
        .Option(43) = 5 && exSliderMin
        .Option(44) = 10 && exSliderMax
    endwith
endwith
```

property Items.CellEditorVisible([Item as Variant], [ColIndex as Variant]) as EditorVisibleEnum

Specifies whether a column's editor is visible or hidden into the cell.

Type	Description
Item as Variant	A long expression that indicates the item's handle
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key
EditorVisibleEnum	An EditorVisibleEnum expression that specifies whether the cell's editor is visible or hidden

Use the CellEditorVisible property to hide the cell's editor. Use the Editor or CellEditor property to assign an editor to the entire column or to a specific cell. Use the Locked property to lock an editor. If the cell's editor is hidden, the cell displays the CellValue property as a plain text, if the CellValueFormat property is exText, else if the CellValueFormat property is exHTML the cell displays the CellValue using built-in HTML format.

The following VB sample hides the editor for the focused cell:

```
With G2antt1.Items
   .CellEditorVisible(.FocusItem, G2antt1.FocusColumnIndex) = False
End With
```

The following C++ sample hides the editor for the focused cell:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetCellEditorVisible( COleVariant( items.GetFocusItem() ), COleVariant(
m_g2antt.GetFocusColumnIndex() ), FALSE );
```

The following VB.NET sample hides the editor for the focused cell:

```
With AxG2antt1.Items
   .CellEditorVisible(.FocusItem, AxG2antt1.FocusColumnIndex) = False
End With
```

The following C# sample hides the editor for the focused cell:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
items.set_CellEditorVisible(items.FocusItem, axG2antt1.FocusColumnIndex, false);
```

The following VFP sample hides the editor for the focused cell:

```
with thisform.G2antt1.Items
    .DefaultItem = .FocusItem
    .CellEditorVisible( 0, thisform.G2antt1.FocusColumnIndex ) = .f.
endwith
```

property Items.CellEnabled([Item as Variant], [ColIndex as Variant]) as Boolean

Returns or sets a value that determines whether a cell can respond to user-generated events.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
Boolean	A boolean expression that indicates whether the cell is enabled or disabled.

Use the CellEnabled property to disable a cell. A disabled cell looks grayed. Use the EnableItem property to disable an item. Once that one cell is disabled it cannot be checked or clicked. Use the SelectableItem property to specify the user can select an item. To disable a column you can use Enabled property of the Column object.

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell (see ItemCell property). Here's few hints how to use properties with Item and ColIndex parameters:

```
G2antt1.Items.CellBold(, G2antt1.Items.ItemCell(G2antt1.Items(0), 0)) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), 0) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), "ColumnName") = True
```

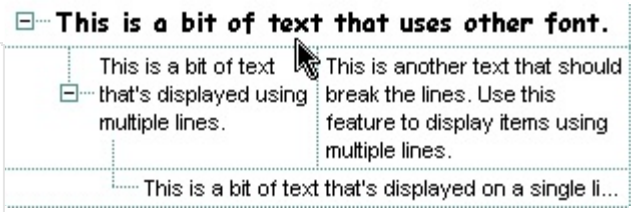
property Items.CellFont ([Item as Variant], [ColIndex as Variant]) as IFontDisp

Retrieves or sets the cell's font.

Type	Description
Item as Variant	A long expression that indicates the item's handle, or optional if the cell's handle is passed to ColIndex parameter
ColIndex as Variant	A long expression that indicates the column's index or cell's handle, or a string expression that indicates the column's caption.
IFontDisp	A Font object that indicates the cell's font.

By default, the CellFont property is nothing. If the CellFont property is noting, the cell uses the item's font. Use the CellFont and ItemFont properties to specify different fonts for cells or items. Use the CellBold, CellItalic, CellUnderline, CellStrikeout, ItemBold, ItemUnderline, ItemStrikeout, ItemItalic or CellValueFormat to specify different font attributes. Use the Refresh method to refresh the control's content on the fly. Use the BeginUpdate and EndUpdate methods if you are doing multiple changes, so no need for an update each time a change is done. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula.

The following VB sample changes the font for the focused cell:




```
With G2antt1.Items
    .CellFont(.FocusItem, 0) = G2antt1.Font
    With .CellFont(.FocusItem, 0)
        .Name = "Comic Sans MS"
        .Size = 10
        .Bold = True
    End With
End With
G2antt1.Refresh
```

The following C++ sample changes the font for the focused cell:

```
#include "Items.h"
#include "Font.h"
```

```
CItems items = m_g2antt.GetItems();
COleVariant vtItem(items.GetFocusItem()), vtColumn( (long)0 );
items.SetCellFont( vtItem, vtColumn, m_g2antt.GetFont().m_lpDispatch );
COleFont font = items.GetCellFont( vtItem, vtColumn );
font.SetName( "Comic Sans MS" );
font.SetBold( TRUE );
m_g2antt.Refresh();
```

The following VB.NET sample changes the font for the focused cell:

```
With AxG2antt1.Items
   .CellFont(.FocusItem, 0) = IFDH.GetIFontDisp(AxG2antt1.Font)
   With .CellFont(.FocusItem, 0)
      .Name = "Comic Sans MS"
      .Bold = True
   End With
End With
AxG2antt1.CtlRefresh()
```

where the IFDH class is defined like follows:

```
Public Class IFDH
   Inherits System.Windows.Forms.AxHost

   Sub New()
      MyBase.New("")
   End Sub

   Public Shared Function GetIFontDisp(ByVal font As Font) As Object
      GetIFontDisp = AxHost.GetIFontFromFont(font)
   End Function

End Class
```

The following C# sample changes the font for the focused cell:

```
axG2antt1.Items.set_CellFont( axG2antt1.Items.FocusItem, 0, IFDH.GetIFontDisp(
axG2antt1.Font ) );
stdole.IFontDisp spFont = axG2antt1.Items.get_CellFont(axG2antt1.Items.FocusItem, 0 );
```

```
spFont.Name = "Comic Sans MS";
spFont.Bold = true;
axG2antt1.CtlRefresh();
```

where the IFDH class is defined like follows:

```
internal class IFDH : System.Windows.Forms.AxHost
{
    public IFDH() : base("")
    {
    }

    public static stdole.IFontDisp GetIFontDisp(System.Drawing.Font font)
    {
        return System.Windows.Forms.AxHost.GetIFontFromFont(font) as stdole.IFontDisp;
    }
}
```

The following VFP sample changes the font for the focused cell:

```
with thisform.G2antt1.Items
    .DefaultItem = .FocusItem
    .CellFont(0,0) = thisform.G2antt1.Font
    with .CellFont(0,0)
        .Name = "Comic Sans MS"
        .Bold = .t.
    endwith
endwith
thisform.G2antt1.Object.Refresh()
```

property Items.CellForeColor([Item as Variant], [ColIndex as Variant]) as Color

Retrieves or sets the cell's foreground color.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
Color	A color expression that indicates the cell's foreground color.

The CellForeColor property identifies the cell's foreground color. Use the ClearCellForeColor property to clear the cell's foreground color. Use the ItemForeColor property to specify the the item's foreground color. Use the Def(exCellForeColor) property to specify the foreground color for all cells in the column.

For instance, the following VB code changes the left top cell of your control:
G2antt1.Items.CellForeColor(G2antt1.Items(0), 0) = vbBlue

In VB.NET or C# you require the following functions until the .NET framework will provide:

You can use the following VB.NET function:

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
   Dim i As Long
i = c.R
i = i + 256 * c.G
i = i + 256 * 256 * c.B
ToUInt32 = Convert.ToUInt32(i)
End Function
```

You can use the following C# function:

```
private UInt32 ToUInt32(Color c)
{
   long i;
   i = c.R;
   i = i + 256 * c.G;
```

```
    i = i + 256 * 256 * c.B;
    return Convert.ToUInt32(i);
}
```

The following C# sample changes the foreground color for the focused cell:

```
axG2antt1.Items.set_CellForeColor(axG2antt1.Items.FocusItem, 0, ToUInt32(Color.Red) );
```

The following VB.NET sample changes the foreground color for the focused cell:

```
With AxG2antt1.Items
   .CellForeColor(.FocusItem, 0) = ToUInt32(Color.Red)
End With
```

The following C++ sample changes the foreground color for the focused cell:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetCellForeColor( COleVariant( items.GetFocusItem() ), COleVariant( (long)0 ),
RGB(255,0,0) );
```

The following VFP sample changes the foreground color for the focused cell:

```
with thisform.G2antt1.Items
   .DefaultItem = .FocusItem
   .CellForeColor( 0, 0 ) = RGB(255,0,0)
endwith
```

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell. Here's few hints how to use properties with Item and ColIndex parameters:

```
G2antt1.Items.CellBold(, G2antt1.Items.ItemCell(G2antt1.Items(0), 0)) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), 0) = True
```

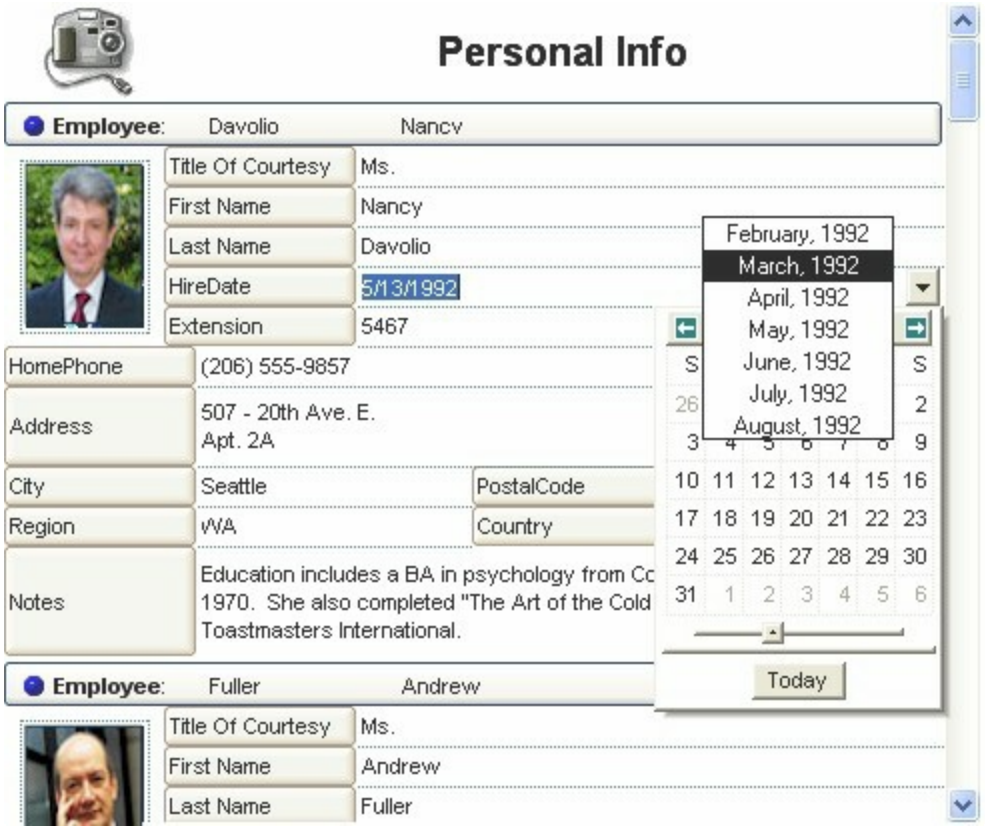
```
G2antt1.Items.CellBold(G2antt1.Items(0), "ColumnName") = True
```

property Items.CellFormatLevel([Item as Variant], [ColIndex as Variant]) as String

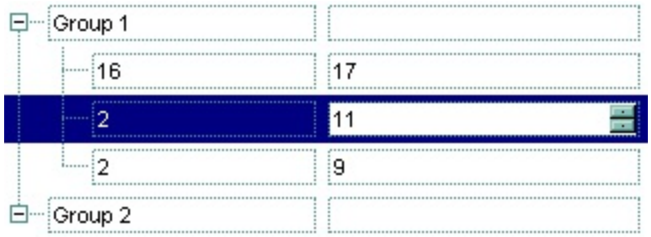
Specifies the arrangement of the fields inside the cell.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key.
String	A CRD string expression that indicates the layout of the cell. The Index elements in the CRD string indicates the index of the column being displayed.

By default, the CellFormatLevel property is empty. If the CellFormatLevel property is empty, the cell displays it's caption. Use the CellValue property to assign a value to a cell. If the CellFormatLevel property is not empty, it indicates the layout being displayed in the cell's area. For instance, the CellFormatLevel = "1/2" indicates that the cell's area is vertically divided such as the up part displays the caption of the cell in the first column, and the down part displays the caption of the cell in the second column. The height of the item is NOT changed, after calling the CellFormatLevel property. Use the ItemHeight property to specify the height of the item. Use the DefaultItemHeight property to specify the default height of the items before inserting them. Use the Def(exCellFormatLevel) property to specify the layout for all cells in the same column. For instance, you can have a specify layout for some cells using the Def(exCellFormatLevel) property (by default it is applied to all cells in the column), and for other cells you can use the CellFormatLevel property to specify different layouts, or to remove the default layout. Use the FormatLevel property to arrange the columns in the control's header bar.



For instance, this layout [dgl=1]""[b=0]:4,(4;""[b=4]/0/4;""[b=1]),""[b=0]:4 adds a 4 pixels-borders around any object its applies (in this case all columns), like in the following picture:



property Items.CellHAlignment ([Item as Variant], [ColIndex as Variant]) as AlignmentEnum

Retrieves or sets a value that indicates the alignment of the cell's caption.

Type	Description
Item as Variant	A long expression that indicates the handle of the item.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's key or the column's caption.
AlignmentEnum	An AlignmentEnum expression that indicates the alignment of the cell's caption.

The CellHAlignment property aligns a particular cell. Use the Alignment property of the Column object to align all the cells in the column. Use the CellVAlignment property to align vertically the caption of the cell, when the item displays its content using multiple lines. Use the ClearCellHAlignment method to clear the cell's alignment previously set by the CellHAlignment property. If the CellHAlignment property is not set, the Alignment property of the Column object indicates the cell's alignment. If the cell belongs to the column that displays the hierarchy (TreeColumnIndex property), the cell can be aligned to the left or to the right.

The following VB sample right aligns the focused cell:

```
With G2antt1.Items
   .CellHAlignment(.FocusItem, 0) = AlignmentEnum.RightAlignment
End With
```

The following C++ sample right aligns the focused cell:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetCellHAlignment( COleVariant( items.GetFocusItem() ), COleVariant( (long)0 ), 2
/*RightAlignment*/ );
```

The following VB.NET sample right aligns the focused cell:

```
With AxG2antt1.Items
   .CellHAlignment(.FocusItem, 0) = EXG2ANTTLib.AlignmentEnum.RightAlignment
End With
```

The following C# sample right aligns the focused cell:

```
axG2antt1.Items.set_CellHAlignment(axG2antt1.Items.FocusItem, 0,
EXG2ANTTLib.AlignmentEnum.RightAlignment);
```

The following VFP sample right aligns the focused cell:

```
with thisform.G2antt1.Items
    .DefaultItem = .FocusItem
    .CellHAlignment(0,0) = 2 && RightAlignment
endwith
```

property Items.CellHasButton([Item as Variant], [ColIndex as Variant]) as Boolean

Retrieves or sets a value indicating whether the cell has associated a push button or not.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
Boolean	A boolean expression that indicates whether the cell contains a button.

The CellHasButton property specifies whether the cell display a button inside. When the cell's button is clicked the control fires ButtonClick event. The caption of the push button is specified by the CellValue property. Use the Def property to assign buttons for all cells in the column. Use the Add method to add new skins to the control. Use the Background property to specify a background color or a visual appearance for specific parts in the control. See also: CellButtonAutoWidth property.

The following VB sample changes the appearance for buttons in the cells. The sample use the skin "  " when the button is up, and the skin "  " when the button is down:

```
With G2antt1
   With .VisualAppearance
      .Add &H20, App.Path + "\buttonu.ebn"
      .Add &H21, App.Path + "\buttond.ebn"
   End With
   .Background(exCellButtonUp) = &H20000000
   .Background(exCellButtonDown) = &H21000000
End With
```

The following C++ sample changes the appearance for buttons in the cells:

```
#include "Appearance.h"
m_g2antt.GetVisualAppearance().Add( 0x20,
COleVariant(_T("D:\\Temp\\ExG2antt.Help\\buttonu.ebn")) );
m_g2antt.GetVisualAppearance().Add( 0x21,
COleVariant(_T("D:\\Temp\\ExG2antt.Help\\buttond.ebn")) );
m_g2antt.SetBackground( 2 /*exCellButtonUp*/, 0x20000000 );
```

```
m_g2antt.SetBackground( 3 /*exCellButtonDown*/, 0x21000000 );
```

The following VB.NET sample changes the appearance for buttons in the cells.

```
With AxG2antt1
   With .VisualAppearance
      .Add(&H20, "D:\Temp\ExG2antt.Help\buttonu.ebn")
      .Add(&H21, "D:\Temp\ExG2antt.Help\buttond.ebn")
   End With
   .set_Background(EXG2ANTTLib.BackgroundPartEnum.exCellButtonUp, &H20000000)
   .set_Background(EXG2ANTTLib.BackgroundPartEnum.exCellButtonDown, &H21000000)
End With
```

The following C# sample changes the appearance for buttons in the cells.

```
axG2antt1.VisualAppearance.Add(0x20, "D:\\Temp\\ExG2antt.Help\\buttonu.ebn");
axG2antt1.VisualAppearance.Add(0x21, "D:\\Temp\\ExG2antt.Help\\buttond.ebn");
axG2antt1.set_Background(EXG2ANTTLib.BackgroundPartEnum.exCellButtonUp,
0x20000000);
axG2antt1.set_Background(EXG2ANTTLib.BackgroundPartEnum.exCellButtonDown,
0x21000000);
```

The following VFP sample changes the appearance for buttons in the cells.

```
With thisform.G2antt1
   With .VisualAppearance
      .Add(32, "D:\Temp\ExG2antt.Help\buttonu.ebn")
      .Add(33, "D:\Temp\ExG2antt.Help\buttond.ebn")
   EndWith
   .Object.Background(2) = 536870912
   .Object.Background(3) = 553648128
endwith
```

the 536870912 indicates the 0x20000000 value in hexadecimal, and the 553648128 indicates the 0x21000000 value in hexadecimal

The following VB sample sets the cells of the first column to be of button type, and displays a message if the button is clicked:

```
Private Sub G2antt1_AddItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
```

```
   G2antt1.Items.CellHasButton(Item, 0) = True
End Sub

Private Sub G2antt1_ButtonClick(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal ColIndex As Long)
   MsgBox "The cell of button type has been clicked"
End Sub
```

The following VB sample assigns a button to the focused cell:

```
With G2antt1.Items
   .CellHasButton(.FocusItem, 0) = True
End With
```

The following C++ sample assigns a button to the focused cell:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetCellHasButton( COleVariant( items.GetFocusItem() ), COleVariant( (long)0 ), TRUE );
```

The following VB.NET sample assigns a button to the focused cell:

```
With AxG2antt1.Items
   .CellHasButton(.FocusItem, 0) = True
End With
```

The following C# sample assigns a button to the focused cell:

```
axG2antt1.Items.set_CellHasButton(axG2antt1.Items.FocusItem, 0, true);
```

The following VFP sample assigns a button to the focused cell:

```
with thisform.G2antt1.Items
   .DefaultItem = .FocusItem
   .CellHasButton(0,0) = .t.
endwith
```

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of

an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell. Here's few hints how to use properties with Item and ColIndex parameters:

```
G2antt1.Items.CellBold(, G2antt1.Items.ItemCell(G2antt1.Items(0), 0)) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), 0) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), "ColumnName") = True
```

property Items.CellHasCheckBox([Item as Variant], [ColIndex as Variant]) as Boolean

Retrieves or sets a value indicating whether the cell has associated a checkbox or not.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
Boolean	A boolean expression that indicates whether the cell contains a check box button.

To change the state for a check cell you have to use CellState property. The cell cannot display in the same time a radio and a check button. The control fires CellStateChanged event when the cell's state has been changed. To set the cell of radio type you have call CellHasRadioButton property. Use the Def property to assign check boxes for all cells in the column. Use the CellImage property to add a single icon to a cell. Use the CellImages property to assign multiple icons to a cell. Use the CellPicture property to load a custom size picture to a cell. Use the PartialCheck property to allow partial check feature within the column. Use the CheckImage property to change the check box appearance. Use the FilterType property on exCheck to filter for checked or unchecked items. Use the Def(exCellDrawPartsOrder) property to change the positions of drawing elements in the cell.

The following sample enumerates the cells in the first column and assign a checkbox to all of them:

```
Dim h As Variant
G2antt1.BeginUpdate
With G2antt1.Items
For Each h In G2antt1.Items
   .CellHasCheckBox(h, 0) = True
Next
End With
G2antt1.EndUpdate
```

The same thing we can do using the Def property like follows:

```
With G2antt1.Columns(0)
   .Def(exCellHasCheckBox) = True
```

```
End With
```

The following sample shows how how set the type of cells to radio type while adding new items:

```
Private Sub G2antt1_AddItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
   G2antt1.Items.CellHasCheckBox(Item, 0) = True
End Sub
```

The following sample shows how to use the CellStateChanged event to display a message when a cell of radio or check type has changed its state:

```
Private Sub G2antt1_CellStateChanged(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal ColIndex As Long)
   Debug.Print "The cell """ & G2antt1.Items.CellValue(Item, ColIndex) & """ has changed its state. The new state is " & IIf(G2antt1.Items.CellState(Item, ColIndes) = 0, "Unchecked", "Checked")
End Sub
```

The following VB sample adds a checkbox to the focused cell:

```
With G2antt1.Items
   .CellHasCheckBox(.FocusItem, 0) = True
End With
```

The following C++ sample adds a checkbox to the focused cell:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetCellHasCheckBox( COleVariant( items.GetFocusItem() ), COleVariant( (long)0 ), TRUE );
```

The following C# sample adds a checkbox to the focused cell:

```
axG2antt1.Items.set_CellHasCheckBox(axG2antt1.Items.FocusItem, 0, true);
```

The following VB.NET sample adds a checkbox to the focused cell:

```
With AxG2antt1.Items
   .CellHasCheckBox(.FocusItem, 0) = True
End With
```

The following VFP sample adds a checkbox to the focused cell:

```
with thisform.G2antt1.Items
   .DefaultItem = .FocusItem
   .CellHasCheckBox(0,0) = .t.
endwith
```

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell. Here's few hints how to use properties with Item and ColIndex parameters:

```
G2antt1.Items.CellBold(, G2antt1.Items.ItemCell(G2antt1.Items(0), 0)) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), 0) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), "ColumnName") = True
```

property Items.CellHasRadioButton([Item as Variant], [ColIndex as Variant]) as Boolean

Retrieves or sets a value indicating whether the cell has associated a radio button or not.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
Boolean	A boolean expression that indicates whether the cell contains a radio button.

Retrieves or sets a value indicating whether the cell has associated a radio button or not. To change the state for a radio cell you have to use CellState property. The cell cannot display in the same time a radio and a check button. The control fires CellStateChanged event when the cell's state has been changed. To set the cell of check type you have call CellHasCheckBox property. To add or remove a cell to a given radio group you have to use CellRadioGroup property. Use the Def property to assign radio buttons for all cells in the column. Use the CellImage property to add a single icon to a cell. Use the CellImages property to assign multiple icons to a cell. Use the CellPicture property to load a custom size picture to a cell. Use the RadioImage property to change the radio button appearance. Use the Def(exCellDrawPartsOrder) property to change the positions of drawing elements in the cell.

The following VB sample sets the radio type for all cells in the first column, and group all of them in the same radio group (1234):

```
Dim h As Variant
G2antt1.BeginUpdate
With G2antt1.Items
For Each h In G2antt1.Items
   .CellHasRadioButton(h, 0) = True
   .CellRadioGroup(h, 0) = 1234
Next
End With
G2antt1.EndUpdate
```

or

```
Private Sub G2antt1_AddItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
```

```
    G2antt1.Items.CellHasRadioButton(Item, 0) = True
    G2antt1.Items.CellRadioGroup(Item, 0) = 1234
End Sub
```

To find out the radio cell that is checked in the radio group 1234 you have to call: MsgBox G2antt1.Items.CellValue(, G2antt1.Items.CellChecked(1234))

The following sample group all cells of the first column into a radio group, and display the cell's checked on the radio group when the state of a radio group has been changed:

```
Private Sub G2antt1_AddItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
    G2antt1.Items.CellHasRadioButton(Item, 0) = True
    G2antt1.Items.CellRadioGroup(Item, 0) = 1234 ' The 1234 is arbirary and it represents the identifier for the radio group
End Sub

Private Sub G2antt1_CellStateChanged(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal ColIndex As Long)
    Debug.Print "In the 1234 radio group the """ & G2antt1.Items.CellValue(, G2antt1.Items.CellChecked(1234)) & """ is checked."
End Sub
```

The following VB sample assigns a radio button to the focused cell:

```
With G2antt1.Items
    .CellHasRadioButton(.FocusItem, 0) = True
    .CellRadioGroup(.FocusItem, 0) = 1234
End With
```

The following C++ sample assigns a radio button to the focused cell:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetCellHasRadioButton( COleVariant( items.GetFocusItem() ), COleVariant( (long)0 ), TRUE );
items.SetCellRadioGroup( COleVariant( items.GetFocusItem() ), COleVariant( (long)0 ), 1234 );
```

The following VB.NET sample assigns a radio button to the focused cell:

```
With AxG2antt1.Items
   .CellHasRadioButton(.FocusItem, 0) = True
   .CellRadioGroup(.FocusItem, 0) = 1234
End With
```

The following C# sample assigns a radio button to the focused cell:

```
axG2antt1.Items.set_CellHasRadioButton(axG2antt1.Items.FocusItem, 0, true);
axG2antt1.Items.set_CellRadioGroup(axG2antt1.Items.FocusItem, 0, 1234);
```

The following VFP sample assigns a radio button to the focused cell:

```
with thisform.G2antt1.Items
   .DefaultItem = .FocusItem
   .CellHasRadioButton(0,0) = .t.
   .CellRadioGroup(0,0) = 1234
endwith
```

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell. Here's few hints how to use properties with Item and ColIndex parameters:

```
G2antt1.Items.CellBold(, G2antt1.Items.ItemCell(G2antt1.Items(0), 0)) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), 0) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), "ColumnName") = True
```

property Items.CellHyperLink ([Item as Variant], [ColIndex as Variant]) as Boolean

Specifies whether the cell's is highlighted when the cursor mouse is over the cell.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, or a string expression that indicates the column's caption.
Boolean	A boolean expression that indicates whether the cell is highlighted when the cursor is over the cell.

Use the CellHyperLink property to add hyperlink cells to your list/tree. Use the HyperLinkClick event to notify your application when a hyperlink cell is clicked. Use the CellForeColor property to specify the cell's foreground color. Use the HyperLinkColor property to specify the hyperlink color. Use the <a> anchor elements to insert hyperlinks to any cell, bar or link.

property Items.CellImage ([Item as Variant], [ColIndex as Variant]) as Long

Retrieves or sets an Image that is displayed on the cell's area.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
Long	A long value that indicates the image index. The last 7 bits in the high significant byte of the long expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part.

Use the CellImage property to assign a single icon to a cell. Use the CellImages property to assign multiple icons to a cell. Use the Images method to assign icons to the control at runtime. You can add images at design time by dragging a file to image editor of the control. The CellImage = 0 removes the cell's image. The collection of Images is 1 based. The CellImageClick event occurs when the cell's image is clicked. Use the ItemFromPoint property to retrieve the part of the control being clicked. Use the CellHasCheckBox property to add a check box to a cell. Use the CellHasRadioButton property to assign a radio button to a cell. Use the CellPicture property to load a custom size picture to a cell. Use the **<img>** HTML tag to insert icons inside the cell's caption, if the CellValueFormat property is exHTML. Use the FilterType property on exImage to filter items by icons. Use the Def(exCellDrawPartsOrder) property to change the positions of drawing elements in the cell.

The following VB sample sets cell's image for the first column while new items are added (to run the sample make sure that control's images collection is not empty):

```
Private Sub G2antt1_AddItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
  G2antt1.Items.CellImage(Item, 0) = 1
End Sub
```

The following VB sample changes the cell's image when the user has clicked on the cell's image (to run the following sample you have to add two images to the g2antt's images collection.),

```
Private Sub G2antt1_AddItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
   G2antt1.Items.CellImage(Item, 0) = 1
End Sub

Private Sub G2antt1_CellImageClick(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal ColIndex
As Long)
   G2antt1.Items.CellImage(Item, ColIndex) = G2antt1.Items.CellImage(Item, ColIndex)
Mod 2 + 1
End Sub
```

The following C++ sample displays the first icon in the focused cell:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetCellImage( COleVariant( items.GetFocusItem() ), COleVariant( (long)0 ), 1 );
```

The following C# sample displays the first icon in the focused cell:

```
axG2antt1.Items.set_CellImage(axG2antt1.Items.FocusItem, 0, 1);
```

The following VB.NET sample displays the first icon in the focused cell:

```
With AxG2antt1.Items
   .CellImage(.FocusItem, 0) = 1
End With
```

The following VFP sample displays the first icon in the focused cell:

```
with thisform.G2antt1.Items
   .DefaultItem = .FocusItem
   .CellImage(0,0) = 1
endwith
```

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell. Here's few hints how to use properties with Item and ColIndex parameters:

```
G2antt1.Items.CellBold(, G2antt1.Items.ItemCell(G2antt1.Items(0), 0)) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), 0) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), "ColumnName") = True
```

property Items.CellImages ([Item as Variant], [ColIndex as Variant]) as Variant

Specifies an additional list of icons shown in the cell.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
Variant	A string expression that indicates the list of icons shown in the cell.

The CellImages property assigns multiple icons to a cell. The CellImage property assign a single icon to the cell. Instead if multiple icons need to be assigned to a single cell you have to use the CellImages property. The CellImages property takes a list of additional icons and display them in the cell. The list is separated by ',' and should contain numbers that represent indexes to Images list collection. Use the ItemFromPoint property to retrieve the part of the control being clicked. Use the CellHasCheckBox property to add a check box to a cell. Use the CellHasRadioButton property to assign a radio button to a cell. Use the CellPicture property to load a custom size picture to a cell. Use the **<img>** HTML tag to insert icons inside the cell's caption, if the CellValueFormat property is exHTML. Use the Def(exCellDrawPartsOrder) property to change the positions of drawing elements in the cell.

The following VB sample assigns the first and third icon to the cell:

```
With G2antt1.Items
    .CellImages(.ItemByIndex(0), 1) = "1,3"
End With
```

The following VB sample displays the index of icon being clicked:

```
Private Sub G2antt1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
  Dim i As HITEM, h As HitTestInfoEnum, c As Long
  With G2antt1
    i = .ItemFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY, c, h)
  End With
  If (i <> 0) Then
    If exHTCellIcon = (h And exHTCellIcon) Then
      Debug.Print "The index of icon being clicked is: " & (h And &HFFFF0000) / 65536
```

```
    End If
  End If
End Sub
```

The following C++ sample assigns the first and the third icon to the cell:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetCellImages( COleVariant( items.GetFocusItem() ), COleVariant( (long)0 ),
COleVariant( "1,3" ) );
```

The following C++ sample displays the index of icon being clicked:

```
#include "Items.h"
void OnMouseUpG2antt1(short Button, short Shift, long X, long Y)
{
  CItems items = m_g2antt.GetItems();
  long c = 0, hit = 0, h = m_g2antt.GetItemFromPoint( X, Y, &c, &hit);
  if ( h != 0 )
  {
    if ( ( hit & 0x44 /*exHTCellIcon*/ ) == 0x44 )
    {
      CString strFormat;
      strFormat.Format( "The index of icon being clicked is: %i\n", (hit >> 16) );
      OutputDebugString( strFormat );
    }
  }
}
```

The following VB.NET sample assigns the first and the third icon to the cell:

```
With AxG2antt1.Items
  .CellImages(.FocusItem, 0) = "1,3"
End With
```

The following VB.NET sample displays the index of icon being clicked:

```
Private Sub AxG2antt1_MouseUpEvent(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_MouseUpEvent) Handles AxG2antt1.MouseUpEvent
```

```
  With AxG2antt1
    Dim i As Integer, c As Integer, hit As EXG2ANTTLib.HitTestInfoEnum
    i = .get_ItemFromPoint(e.x, e.y, c, hit)
    If (Not (i = 0)) Then
      Debug.WriteLine("The index of icon being clicked is: " & (hit And &HFFFF0000) /
65536)
    End If
  End With
End Sub
```

The following C# sample assigns the first and the third icon to the cell:

```
axG2antt1.Items.set_CellImages(axG2antt1.Items.FocusItem, 0, "1,3");
```

The following C# sample displays the index of icon being clicked:

```
private void axG2antt1_MouseUpEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_MouseUpEvent e)
{
  int c = 0;
  EXG2ANTTLib.HitTestInfoEnum hit;
  int i = axG2antt1.get_ItemFromPoint(e.x, e.y, out c, out hit);
  if ((i != 0))
  {
    if ((Convert.ToUInt32(hit) &
Convert.ToUInt32(EXG2ANTTLib.HitTestInfoEnum.exHTCellIcon)) ==
Convert.ToUInt32(EXG2ANTTLib.HitTestInfoEnum.exHTCellIcon))
    {
      string s = axG2antt1.Items.get_CellValue(i, c).ToString();
      s = "Cell: " + s + ", Icon's Index: " + (Convert.ToUInt32(hit) >> 16).ToString();
      System.Diagnostics.Debug.WriteLine(s);
    }
  }
}
```

The following VFP sample assigns the first and the third icon to the cell:

```
with thisform.G2antt1.Items
  .DefaultItem = .FocusItem
```

```
    .CellImages(0,0) = "1,3"
endwith
```

The following VFP sample displays the index of icon being clicked:

```
*** ActiveX Control Event ***
LPARAMETERS button, shift, x, y

local c, hit
c = 0
hit = 0
with thisform.G2antt1
    .Items.DefaultItem = .ItemFromPoint( x, y, @c, @hit )
    if ( .Items.DefaultItem <> 0 )
        if ( bitand( hit, 68 )= 68 )
            wait window nowait .Items.CellValue( 0, c ) + " " + Str( Int((hit - 68)/65536) )
        endif
    endif
endwith
```

Add the code to the MouseUp, MouseMove or MouseDown event,

property Items.CellItalic([Item as Variant], [ColIndex as Variant]) as Boolean

Retrieves or sets a value that indicates whether the cell's caption should appear in italic.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
Boolean	A boolean expression that indicates whether the cell should appear in italic.

Use ItemBold, ItemItalic, ItemUnderline or ItemStrikeOut property to apply different font attributes to the item. Use the CellItalic, CellUnderline, CellBold or CellStrikeOut property to apply different font attributes to the cell. Use the CellValueFormat property to specify an HTML caption. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula.

The following VB sample makes italic the focused cell:

```
With G2antt1.Items
   .CellItalic(.FocusItem, 0) = True
End With
```

The following C++ sample makes italic the focused cell:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetCellItalic( COleVariant( items.GetFocusItem() ), COleVariant( (long)0 ), TRUE );
```

The following C# sample makes italic the focused cell:

```
axG2antt1.Items.set_CellItalic(axG2antt1.Items.FocusItem, 0, true);
```

The following VB.NET sample makes italic the focused cell:

```
With AxG2antt1.Items
   .CellItalic(.FocusItem, 0) = True
End With
```

The following VFP sample makes italic the focused cell:

```
with thisform.G2antt1.Items
   .DefaultItem = .FocusItem
   .CellItalic( 0, 0 ) = .t.
endwith
```

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell. Here's few hints how to use properties with Item and ColIndex parameters:

```
G2antt1.Items.CellBold(, G2antt1.Items.ItemCell(G2antt1.Items(0), 0)) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), 0) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), "ColumnName") = True
```

property Items.CellItem (Cell as HCELL) as HITEM

Retrieves the handle of the item that is owner for a specfic cell.

Type	Description
Cell as HCELL	A long expression that indicates the handle of the cell.
HITEM	A long expression that indicates the handle of the item.

Use the CellItem property to retrieve the item's handle. Use the ItemCell property to gets the cell's handle given an item and a column. Most of the properties of the Items object that have parameters [Item as Variant], [ColIndex as Variant], could use the handle of the cell to identify the cell, instead the ColIndex parameter. For instance the following statements are equivalents:

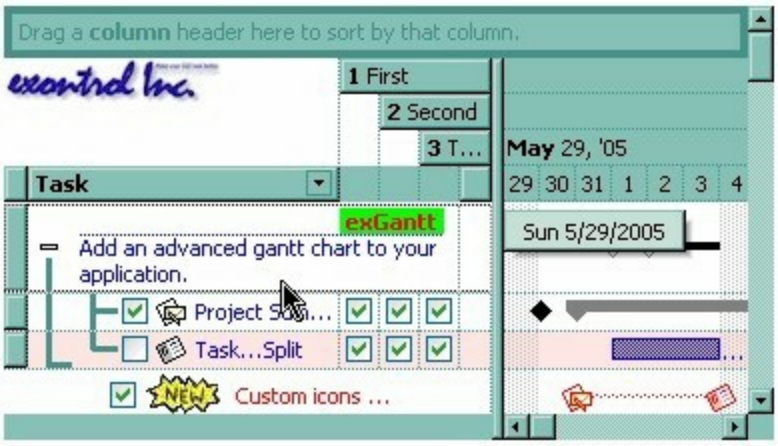
```
With G2antt1.Items
   .CellValue(.FocusItem, 0) = "this"
   .CellValue(, .ItemCell(.FocusItem, 0)) = "this"
End With
```

property Items.CellMerge([Item as Variant], [ColIndex as Variant]) as Variant

Retrieves or sets a value that indicates the index of the cell that's merged to.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
Variant	A long expression that indicates the index of the cell that's merged with, a safe array that holds the indexes of the cells being merged.

Use the CellMerge property to combine two or more cells in the same item in a single cell. The data of the source cell is displayed in the new larger cell. All the other cells' data is not lost. Use the ItemDivider property to display a single cell in the entire item (merging all cells in the same item). Use the UnmergeCells method to unmerge the merged cells. Use the CellMerge property to unmerge a single cell. Use the MergeCells method to combine one or more cells in a single cell. Use the Add method to add new columns to the control. Use the SplitCell property to split a cell.



You can merge the first three cells in the root item using any of the following methods:

```
With G2antt1
  With .Items
    .CellMerge(.RootItem(0), 0) = Array(1, 2)
  End With
End With
```

```
With G2antt1

```

```
   .BeginUpdate
   With .Items
      Dim r As Long
      r = .RootItem(0)
      .CellMerge(r, 0) = 1
      .CellMerge(r, 0) = 2
   End With
   .EndUpdate
End With
```

```
With G2antt1
   .BeginUpdate
   With .Items
      Dim r As Long
      r = .RootItem(0)
      .MergeCells .ItemCell(r, 0), .ItemCell(r, 1)
      .MergeCells .ItemCell(r, 0), .ItemCell(r, 2)
   End With
   .EndUpdate
End With
```

```
With G2antt1
   With .Items
      Dim r As Long
      r = .RootItem(0)
      .MergeCells .ItemCell(r, 0), Array(.ItemCell(r, 1), .ItemCell(r, 2))
   End With
End With
```

```
With G2antt1
   With .Items
      Dim r As Long
      r = .RootItem(0)
      .MergeCells Array(.ItemCell(r, 0), .ItemCell(r, 1), .ItemCell(r, 2))
   End With
End With
```

The following sample shows few methods to unmerge cells:

```
With G2antt1
   With .Items
      .UnmergeCells .ItemCell(.RootItem(0), 0)
   End With
End With
```

```
With G2antt1
   With .Items
      Dim r As Long
      r = .RootItem(0)
      .UnmergeCells Array(.ItemCell(r, 0), .ItemCell(r, 1))
   End With
End With
```

```
With G2antt1
   .BeginUpdate
   With .Items
      .CellMerge(.RootItem(0), 0) = -1
      .CellMerge(.RootItem(0), 1) = -1
      .CellMerge(.RootItem(0), 2) = -1
   End With
   .EndUpdate
End With
```

The following VB sample merges the first three cells in the focused item:

```
With G2antt1.Items
   .CellMerge(.FocusItem, 0) = 1
   .CellMerge(.FocusItem, 0) = 2
End With
```

The following C++ sample merges the first three cells in the focused item:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
COleVariant vtItem( items.GetFocusItem() ), vtColumn( long( 0 ) );
items.SetCellMerge( vtItem, vtColumn, COleVariant( long(1) ) );
items.SetCellMerge( vtItem, vtColumn, COleVariant( long(2) ) );
```

The following VB.NET sample merges the first three cells in the focused item:

```
With AxG2antt1.Items
   .CellMerge(.FocusItem, 0) = 1
   .CellMerge(.FocusItem, 0) = 2
End With
```

The following C# sample merges the first three cells in the focused item:

```
axG2antt1.Items.set_CellMerge(axG2antt1.Items.FocusItem, 0, 1);
axG2antt1.Items.set_CellMerge(axG2antt1.Items.FocusItem, 0, 2);
```

The following VFP sample merges the first three cells in the focused item:

```
with thisform.G2antt1.Items
   .DefaultItem = .FocusItem
   .CellMerge(0,0) = 1
   .CellMerge(0,0) = 2
endwith
```

In other words, the sample shows how to display the first cell using the space occupied by three cells.

property Items.CellParent ([Item as Variant], [ColIndex as Variant]) as Variant

Retrieves the parent of an inner cell.

Type	Description
Item as Variant	A long expression that indicates the handle of the item where the cell is, or 0. If the Item parameter is 0, the ColIndex parameter must indicate the handle of the cell.
ColIndex as Variant	A long expression that indicates the index of the column where a cell is divided, or a long expression that indicates the handle of the cell being divided, if the Item parameter is missing or it is zero.
Variant	A long expression that indicates the handle of the parent cell.

Use the CellParent property to get the parent of the inner cell. The SplitCell method splits a cell in two cells (the newly created cell is called inner cell). Use the InnerCell property to get the inner cell. Use the CellItem property to get the item that's the owner of the cell. The CellParent property gets 0 if the cell is not an inner cell. The parent cell is always displayed to the left side of the cell. The inner cell (InnerCell) is displayed to the right side of the cell.

The following VB sample determines whether the cell is a master cell or an inner cell:

```
Private Function isMaster(ByVal g As EXG2ANTTLibCtl.G2antt, ByVal h As
EXG2ANTTLibCtl.HITEM, ByVal c As Long) As Boolean
  With g.Items
    isMaster = .CellParent(h, c) = 0
  End With
End Function
```

The following VB sample determines the master cell (the cell from where the splitting starts):

```
Private Function getMaster(ByVal g As EXG2ANTTLibCtl.G2antt, ByVal h As
EXG2ANTTLibCtl.HITEM, ByVal c As Long) As EXG2ANTTLibCtl.HCELL
  With g.Items
    Dim r As EXG2ANTTLibCtl.HCELL
    r = c
    If Not (h = 0) Then
      r = .ItemCell(h, c)
```

```
    End If
    While Not (.CellParent(, r) = 0)
      r = .CellParent(, r)
    Wend
    getMaster = r
  End With
End Function
```

The following C++ sample determines whether the cell is a master cell or an inner cell:

```
#include "Items.h"

static long V2I( VARIANT* pv, long nDefault = 0 )
{
  if ( pv )
  {
    if ( pv->vt == VT_ERROR )
      return nDefault;

    COleVariant vt;
    vt.ChangeType( VT_I4, pv );
    return V_I4( &vt );
  }
  return nDefault;
}

BOOL isMaster( CG2antt g2antt, long hItem, long nColIndex )
{
  return V2I( &g2antt.GetItems().GetCellParent( COleVariant( hItem ), COleVariant(
nColIndex ) ) ) == 0;
}
```

The following C++ sample determines the master cell (the cell from where the splitting starts):

```
long getMaster( CG2antt g2antt, long hItem, long nColIndex )
{
  COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
```

```
    CItems items = g2antt.GetItems();
    long r = nColIndex;
    if ( hItem )
        r = items.GetItemCell( hItem, COleVariant( nColIndex ) );
    long r2 = 0;
    while ( r2 = V2I( &items.GetCellParent( vtMissing, COleVariant( r ) ) ) )
        r = r2;
    return r;
}
```

The following VB.NET sample determines whether the cell is a master cell or an inner cell:

```
Private Function isMaster(ByVal g As AxEXG2ANTTLib.AxG2antt, ByVal h As Long, ByVal c
As Long) As Boolean
    With g.Items
        isMaster = .CellParent(h, c) = 0
    End With
End Function
```

The following VB.NET sample determines the master cell (the cell from where the splitting starts):

```
Shared Function getMaster(ByVal g As AxEXG2ANTTLib.AxG2antt, ByVal h As Integer, ByVal
c As Integer) As Integer
    With g.Items
        Dim r As Integer
        r = c
        If Not (h = 0) Then
            r = .ItemCell(h, c)
        End If
        While Not (.CellParent(, r) = 0)
            r = .CellParent(, r)
        End While
        getMaster = r
    End With
End Function
```

The following C# sample determines whether the cell is a master cell or an inner cell:

```
private bool isMaster(AxEXG2ANTTLib.AxG2antt g2antt, int h, int c)
{
   return Convert.ToInt32(g2antt.Items.get_CellParent(h, c)) != 0;
}
```

The following C# sample determines the master cell (the cell from where the splitting starts):

```
private long getMaster(AxEXG2ANTTLib.AxG2antt g, int h, int c)
{
   int r = c, r2 = 0;
   if ( h != 0 )
      r = Convert.ToInt32( g.Items.get_ItemCell(h,c) );
   r2 = Convert.ToInt32( g.Items.get_CellParent(null, r));
   while ( r2 != 0)
   {
      r = r2;
      r2 = Convert.ToInt32( g.Items.get_CellParent(null, r));
   }
   return r;
}
```

property Items.CellPicture ([Item as Variant], [ColIndex as Variant]) as Variant

Retrieves or sets a value that indicates the Picture object displayed by the cell.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
Variant	A Picture object that indicates the cell's picture. (A Picture object implements IPicture interface), a string expression that indicates the base64 encoded string that holds a picture object. Use the eximages tool to save your picture as base64 encoded format.

The control can associate to a cell a check or radio button, an icon, multiple icons, a picture and a caption. Use the CellPicture property to associate a picture to a cell. You can use the CellPicture property when you want to display images with different widths into a cell. Use the CellImage property to associate an icon from Images collection. Use the CellImages property to assign multiple icons to a cell. Use the CellHasCheckBox property to add a check box to a cell. Use the CellHasRadioButton property to assign a radio button to a cell. Use the CellPictureWidth and CellPictureHeight properties to stretch the cell's picture to a specified size. Use the Def(exCellDrawPartsOrder) property to change the positions of drawing elements in the cell.

The following VB sample loads a picture from a file:

```
G2antt1.Items.CellPicture(h, 0) = LoadPicture("c:\winnt\logo.gif")
```

The following VB sample associates a picture to a cell by loading it from a base64 encoded string:

```
Dim s As String
s =
"gBCJr+BAAg0HGwEgwog4jg4ig4BAEFg4AZEKisZjUbAAzg5mg6Zg7Mg7/g0ek8oGcgjsijsk

s = s +
"XgBadlDXdYSXRb9wWBclK2taF1gAI5HiPaN8oPdlNWbaF23KAwyWkNYyXxg9p3WNYjU/c

With G2antt1
```

```
   .BeginUpdate
   .Columns.Add "Column 1"
   With .Items
      Dim h As HITEM
      h = .AddItem("Item 1")
      .CellPicture(h, 0) = s
      .ItemHeight(h) = 24
   End With
   .EndUpdate
End With
```

The following C++ loads a picture from a file:

```
#include
BOOL LoadPicture( LPCTSTR szFileName, IPictureDisp** ppPictureDisp )
{
   BOOL bResult = FALSE;
   if ( szFileName )
   {
      OFSTRUCT of;
      HANDLE hFile = NULL;;
#ifdef _UNICODE
      USES_CONVERSION;
      if ( (hFile = (HANDLE)OpenFile( W2A(szFileName), &of;, OF_READ |
OF_SHARE_COMPAT)) != (HANDLE)HFILE_ERROR )
#else
      if ( (hFile = (HANDLE)OpenFile( szFileName, &of;, OF_READ | OF_SHARE_COMPAT)) !=
(HANDLE)HFILE_ERROR )
#endif
      {
         *ppPictureDisp = NULL;
         DWORD dwHighWord = NULL, dwSizeLow = GetFileSize( hFile, &dwHighWord; );
         DWORD dwFileSize = dwSizeLow;
         HRESULT hResult = NULL;
         if ( HGLOBAL hGlobal = GlobalAlloc(GMEM_MOVEABLE, dwFileSize) )
            if ( void* pvData = GlobalLock( hGlobal ) )
            {
```

```
            DWORD dwReadBytes = NULL;
            BOOL bRead = ReadFile( hFile, pvData, dwFileSize, &dwReadBytes;, NULL );
            GlobalUnlock( hGlobal );
            if ( bRead )
            {
               CComPtr spStream;
               _ASSERTE( dwFileSize == dwReadBytes );
               if ( SUCCEEDED( CreateStreamOnHGlobal( hGlobal, TRUE, &spStream;) ) )
                  if ( SUCCEEDED( hResult = OleLoadPicture( spStream, 0, FALSE,
IID_IPictureDisp, (void**)ppPictureDisp ) ) )
                     bResult = TRUE;
            }
         }
         CloseHandle( hFile );
      }
   }
   return bResult;
}

IPictureDisp* pPicture = NULL;
if ( LoadPicture( "c:\\winnt\\zapotec.bmp", &pPicture; ) )
{
   COleVariant vtPicture;
   V_VT( &vtPicture; ) = VT_DISPATCH;
   pPicture->QueryInterface( IID_IDispatch, (LPVOID*)&V;_DISPATCH( &vtPicture; ) );
   CItems items = m_g2antt.GetItems();
   items.SetCellPicture( COleVariant( items.GetFocusItem() ), COleVariant(long(0)), vtPicture
);
   pPicture->Release();
}
```

The following VB.NET sample loads a picture from a file:

```
With AxG2antt1.Items
   .CellPicture(.FocusItem, 0) =
IPDH.GetIPictureDisp(Image.FromFile("c:\winnt\zapotec.bmp"))
End With
```

where the IPDH class is defined like follows:

```
Public Class IPDH
    Inherits System.Windows.Forms.AxHost

    Sub New()
        MyBase.New("")
    End Sub

    Public Shared Function GetIPictureDisp(ByVal image As Image) As Object
        GetIPictureDisp = AxHost.GetIPictureDispFromPicture(image)
    End Function

End Class
```

The following C# sample loads a picture from a file:

```
axG2antt1.Items.set_CellPicture(axG2antt1.Items.FocusItem, 0,
IPDH.GetIPictureDisp(Image.FromFile("c:\\winnt\\zapotec.bmp")));
```

where the IPDH class is defined like follows:

```
internal class IPDH : System.Windows.Forms.AxHost
{
    public IPDH() : base("")
    {
    }

    public static object GetIPictureDisp(System.Drawing.Image image)
    {
        return System.Windows.Forms.AxHost.GetIPictureDispFromPicture( image );
    }
}
```

The following VFP sample loads a picture from a file:

```
with thisform.G2antt1.Items
    .DefaultItem = .FocusItem
    .CellPicture( 0, 0 ) = LoadPicture("c:\winnt\zapotec.bmp")
```

endwith

property Items.CellPictureHeight ([Item as Variant], [ColIndex as Variant]) as Long

Retrieves or sets a value that indicates the height of the cell's picture.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
Long	A long expression that indicates the height of the cell's picture, or -1, if the property is ignored.

By default, the CellPictureHeight property is -1. Use the CellPicture property to assign a custom size picture to a cell. Use the CellPictureWidth property to specify the width of the cell's picture. The CellPictureWidth and CellPictureHeight properties specifies the size of the area where the cell's picture is stretched. If the CellPictureWidth and CellPictureHeight properties are -1 (by default), the cell displays the full size picture. If the CellPictureHeight property is greater than 0, it indicates the height of the area where the cell's picture is stretched. Use the ItemHeight property to specify the height of the item. Use the CellImage or CellImages property to assign one or more icons to the cell.

property Items.CellPictureWidth ([Item as Variant], [ColIndex as Variant]) as Long

Retrieves or sets a value that indicates the width of the cell's picture.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
Long	A long expression that indicates the width of the cell's picture, or -1, if the property is ignored.

By default, the CellPictureWidth property is -1. Use the CellPicture property to assign a custom size picture to a cell. Use the CellPictureHeight property to specify the height of the cell's picture. The CellPictureWidth and CellPictureHeight properties specifies the size of the area where the cell's picture is stretched. If the CellPictureWidth and CellPictureHeight properties are -1 (by default), the cell displays the full size picture. If the CellPictureWidth property is greater than 0, it indicates the width of the area where the cell's picture is stretched. Use the CellImage or CellImages property to assign one or more icons to the cell.

property Items.CellRadioGroup([Item as Variant], [ColIndex as Variant]) as Long

Retrieves or sets a value indicating the radio group where the cell is contained.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
Long	A long value that identifies the cell's radio group.

Use the CellRadioGroup property to add or remove a radio button from a group. In a radio group only one radio button can be checked. A radio cell cannot be contained by two different radio groups. Use the CellHasRadioButton property to add a radio button to a cell. When the cell's state is changed the control fires the CellStateChanged event. The CellState property specifies the cell's state. By default, when a cell of radio type is created the radio cell is not grouped to any of existent radio groups.

The following VB sample sets the radio type for all cells in the first column, and group all of them in the same radio group (1234):

```
Dim h As Variant
G2antt1.BeginUpdate
With G2antt1.Items
For Each h In G2antt1.Items
   .CellHasRadioButton(h, 0) = True
   .CellRadioGroup(h, 0) = 1234
Next
End With
G2antt1.EndUpdate
```

or

```
Private Sub G2antt1_AddItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
   G2antt1.Items.CellHasRadioButton(Item, 0) = True
   G2antt1.Items.CellRadioGroup(Item, 0) = 1234
End Sub
```

To find out the radio cell that is checked in the radio group 1234 you have to call: MsgBox

G2antt1.Items.CellValue(, G2antt1.Items.CellChecked(1234))

The following sample group all cells of the first column into a radio group, and display the cell's checked on the radio group when the state of a radio group has been changed:

```
Private Sub G2antt1_AddItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
   G2antt1.Items.CellHasRadioButton(Item, 0) = True
   G2antt1.Items.CellRadioGroup(Item, 0) = 1234 ' The 1234 is arbirary and it represents the identifier for the radio group
End Sub

Private Sub G2antt1_CellStateChanged(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal ColIndex As Long)
   Debug.Print "In the 1234 radio group the """ & G2antt1.Items.CellValue(, G2antt1.Items.CellChecked(1234)) & """ is checked."
End Sub
```

The following VB sample assigns a radio button to the focused cell:

```
With G2antt1.Items
   .CellHasRadioButton(.FocusItem, 0) = True
   .CellRadioGroup(.FocusItem, 0) = 1234
End With
```

The following C++ sample assigns a radio button to the focused cell:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetCellHasRadioButton( COleVariant( items.GetFocusItem() ), COleVariant( (long)0 ), TRUE );
items.SetCellRadioGroup( COleVariant( items.GetFocusItem() ), COleVariant( (long)0 ), 1234 );
```

The following VB.NET sample assigns a radio button to the focused cell:

```
With AxG2antt1.Items
   .CellHasRadioButton(.FocusItem, 0) = True
   .CellRadioGroup(.FocusItem, 0) = 1234
End With
```

The following C# sample assigns a radio button to the focused cell:

```
axG2antt1.Items.set_CellHasRadioButton(axG2antt1.Items.FocusItem, 0, true);
axG2antt1.Items.set_CellRadioGroup(axG2antt1.Items.FocusItem, 0, 1234);
```

The following VFP sample assigns a radio button to the focused cell:

```
with thisform.G2antt1.Items
   .DefaultItem = .FocusItem
   .CellHasRadioButton(0,0) = .t.
   .CellRadioGroup(0,0) = 1234
endwith
```

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell. Here's few hints how to use properties with Item and ColIndex parameters:

```
G2antt1.Items.CellBold(, G2antt1.Items.ItemCell(G2antt1.Items(0), 0)) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), 0) = True
```

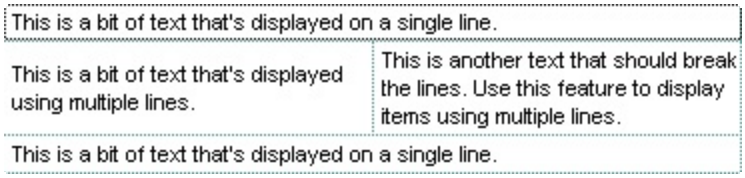
```
G2antt1.Items.CellBold(G2antt1.Items(0), "ColumnName") = True
```

property Items.CellSingleLine([Item as Variant], [ColIndex as Variant]) as CellSingleLineEnum

Retrieves or sets a value indicating whether the cell's caption is painted using one or more lines.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
CellSingleLineEnum	A CellSingleLineEnum expression that indicates whether the cell displays its caption using one or more lines.

By default, the CellSingleLine property is exCaptionSingleLine / True, which indicates that the cell's caption is displayed on a single line. Use the Def(exCellSingleLine) property to specify that all cells in the column display their content using multiple lines. The control can displays the cell's caption using more lines, if the CellSingleLine property is exCaptionWordWrap or exCaptionBreakWrap. The CellSingleLine property wraps the cell's caption so it fits in the cell's client area. If the text doesn't fit the cell's client area, the height of the item is increased or decreased. When the CellSingleLine is exCaptionWordWrap / exCaptionBreakWrap / False, the height of the item is computed based on each cell caption. *If the CellSingleLine property is* exCaptionWordWrap / exCaptionBreakWrap / *False, changing the ItemHeight property has no effect.* Use the ItemMaxHeight property to specify the maximum height of the item when its height is variable. Use the CellVAlignment property to align vertically a cell.



If using the CellSingleLine / Def(exCellSingleLine) property, we recommend to set the ScrollBySingleLine property on True so all items can be scrolled.

The following VB sample displays the caption of the focused cell using multiple lines:

```
With G2antt1.Items
   .CellSingleLine(.FocusItem, 0) = True
End With
```

The following C++ sample displays the caption of the focused cell using multiple lines:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetCellSingleLine( COleVariant( items.GetFocusItem() ), COleVariant( long(0) ), FALSE );
```

The following VB.NET sample displays the caption of the focused cell using multiple lines:

```
With AxG2antt1.Items
   .CellSingleLine(.FocusItem, 0) = False
End With
```

The following C# sample displays the caption of the focused cell using multiple lines:

```
axG2antt1.Items.set_CellSingleLine(axG2antt1.Items.FocusItem, 0, false);
```

The following VFP sample displays the caption of the focused cell using multiple lines:

```
with thisform.G2antt1.Items
   .DefaultItem = .FocusItem
   .CellSingleLine( 0, 0 ) = .f.
endwith
```

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell. Here's few hints how to use properties with Item and ColIndex parameters:

```
G2antt1.Items.CellBold(, G2antt1.Items.ItemCell(G2antt1.Items(0), 0)) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), 0) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), "ColumnName") = True
```

property Items.CellState([Item as Variant], [ColIndex as Variant]) as Long

Retrieves or sets the cell's state. Has effect only for check and radio cells.

Type	Description
Item as Variant	A long expression that indicates the item's handle that indicates the owner of the cell.
ColIndex as Variant	A long expression that identifies the column's index, or a string expression that specifies the column's caption or the column's key.
Long	A long value that indicates the cell's state.

Use the CellState property to change the cell's state. The CellState property has effect only for check and radio cells. Use the CellHasCheckBox property to assign a check box to a cell. Use the CellHasRadioButton property to add a radio button to a cell. The control fires the CellStateChanged event when user changes the cell's state. Use the PartialCheck property to allow partial check feature within the column. Use the CheckImage property to change the check box appearance. Use the RadioImage property to change the radio button appearance. Use the FilterType property on exCheck to filter for checked or unchecked items.

Once the user clicks a check-box, radio-button, the control fires the following events:

- CellStateChanging event, where the NewState parameter indicates the new state of the cell's checkbox / radio-button. You can change the NewState parameter during this event. For instance, NewState = Items.CellState(Item,ColIndex) un-changes the cell's state once the user tries to change it.
- CellStateChanged event notifies your application that the cell's check-box or radio-button has been changed. The CellState property determines the check-box/radio-button state of the cell.

The following VB sample adds a check box that's checked to the focused cell:

```
With G2antt1.Items
  .CellHasCheckBox(.FocusItem, 0) = True
  .CellState(.FocusItem, 0) = 1
End With
```

The following C++ sample adds a check box that's checked to the focused cell:

```
#include "Items.h"
```

```
CItems items = m_g2antt.GetItems();
COleVariant vtItem( items.GetFocusItem() ), vtColumn( long(0) );
items.SetCellHasCheckBox( vtItem, vtColumn, TRUE );
items.SetCellState( vtItem, vtColumn, 1 );
```

The following VB.NET sample adds a check box that's checked to the focused cell:

```
With AxG2antt1.Items
   .CellHasCheckBox(.FocusItem, 0) = True
   .CellState(.FocusItem, 0) = 1
End With
```

The following C# sample adds a check box that's checked to the focused cell:

```
axG2antt1.Items.set_CellHasCheckBox(axG2antt1.Items.FocusItem, 0, true);
axG2antt1.Items.set_CellState(axG2antt1.Items.FocusItem, 0, 1);
```

The following VFP sample adds a check box that's checked to the focused cell:

```
with thisform.G2antt1.Items
   .DefaultItem = .FocusItem
   .CellHasCheckBox( 0, 0 ) = .t.
   .CellState( 0,0 ) = 1
endwith
```

The following VB sample changes the state for a cell to checked state:
G2antt1.Items.CellState(G2antt1.Items(0), 0) = 1,

The following VB sample changes the state for a cell to to unchecked state:
G2antt1.Items.CellState(G2antt1.Items(0), 0) = 0,

The following VB sample changes the state for a cell to partial checked state:
G2antt1.Items.CellState(G2antt1.Items(0), 0) = 2

The following VB sample displays a message when a cell of radio or check type is changing its state:

```
Private Sub G2antt1_AddItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
   G2antt1.Items.CellHasCheckBox(Item, 0) = True
End Sub
```

```
Private Sub G2antt1_CellStateChanged(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal ColIndex As Long)
    Debug.Print "The cell """ & G2antt1.Items.CellValue(Item, ColIndex) & """ has changed its state. The new state is " & IIf(G2antt1.Items.CellState(Item, ColIndes) = 0, "Unchecked", "Checked")
End Sub
```

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell. Here's few hints how to use properties with Item and ColIndex parameters:

```
G2antt1.Items.CellBold(, G2antt1.Items.ItemCell(G2antt1.Items(0), 0)) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), 0) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), "ColumnName") = True
```

property Items.CellStrikeOut([Item as Variant], [ColIndex as Variant]) as Boolean

Retrieves or sets a value that indicates whether the cell's caption should appear in strikeout.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
Boolean	A boolean expression that indicates whether the cell's caption should appear in strikeout.

If the CellStrikeOut property is True, the cell's font is displayed with a horizontal line through it. Use ItemBold, ItemItalic, ItemUnderline or ItemStrikeOut property to apply different font attributes to the item. Use the CellItalic, CellUnderline, CellBold or CellStrikeOut property to apply different font attributes to the cell. Use the CellValueFormat property to specify an HTML caption. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula.

The following VB sample draws a horizontal line through the caption of the cell that has the focus:

```
With G2antt1.Items
   .CellStrikeOut(.FocusItem, 0) = True
End With
```

The following C++ sample draws a horizontal line through the caption of the cell that has the focus:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetCellStrikeOut( COleVariant( items.GetFocusItem() ), COleVariant( (long)0 ), TRUE );
```

The following C# sample draws a horizontal line through the caption of the cell that has the focus:

```
axG2antt1.Items.set_CellStrikeOut(axG2antt1.Items.FocusItem, 0, true);
```

The following VB.NET sample draws a horizontal line through the caption of the cell that has the focus:

```
With AxG2antt1.Items
   .CellStrikeOut(.FocusItem, 0) = True
End With
```

The following VFP sample draws a horizontal line through the caption of the cell that has the focus:

```
with thisform.G2antt1.Items
   .DefaultItem = .FocusItem
   .CellStrikeOut(0, 0 ) = .t.
endwith
```

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell. Here's few hints how to use properties with Item and ColIndex parameters:

```
G2antt1.Items.CellBold(, G2antt1.Items.ItemCell(G2antt1.Items(0), 0)) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), 0) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), "ColumnName") = True
```

property Items.CellToolTip([Item as Variant], [ColIndex as Variant]) as String

Retrieves or sets a text that is used to show the tooltip's cell.

Type	Description
Item as Variant	A long expression that indicates the handle of the item.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
String	A string expression that indicates the cell's tooltip.

By default, the CellToolTip property is "..." (three dots). If the CellToolTip property is "..." the control displays the cell's caption if it doesn't fit the cell's client area. If the CellToolTip property is different than "...", the control shows a tooltip that displays the CellToolTip value. The control fires the ToolTip event when the column's tooltip is about to be displayed. Use the ToolTipWidth property to specify the width of the tooltip window. The ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the <font> element to specify a different font or size for the tooltip, or use the ToolTipFont property to specify a different font or size for all tooltips in the control. Use the ItemBar(,,exBarToolTip) property to specify a tooltip for a bar. Use the Link(,exLinkToolTip) property to specify the tooltip to be shown when the cursor hovers the link.

The tooltip supports the following HTML tags:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.

  The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using <a ;exp=> or <a ;e64=> anchor tags. The exp/e64 field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption.

- exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
- e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to

your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.

- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with $^{\text{subscript}}$
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

outlined

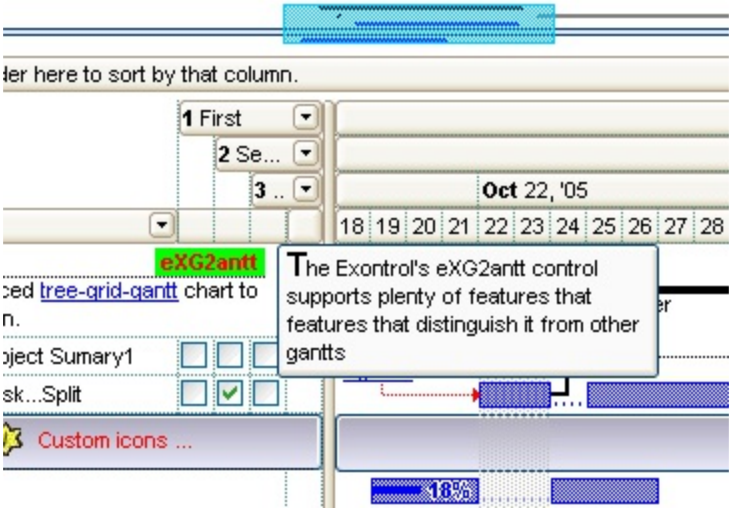
- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb

represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

shadow

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

outline anti-aliasing



Note: The intersection of an item with a column defines a cell. Each cell is uniquely represented by its handle. The cell's handle is of HCELL type, that's equivalent with a long type. All properties of Items object that have two parameters *Item* and *ColIndex,* that refers a cell.

property Items.CellUnderline([Item as Variant], [ColIndex as Variant]) as Boolean

Retrieves or sets a value that indicates whether the cell's caption should appear in underline.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's caption or the column's key.
Boolean	A boolean expression that indicates whether the cell is underlined.

Use ItemBold, ItemItalic, ItemUnderline or ItemStrikeOut property to apply different font attributes to the item. Use the CellItalic, CellUnderline, CellBold or CellStrikeOut property to apply different font attributes to the cell. Use the CellValueFormat property to specify an HTML caption. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula.

The following VB sample underlines the focused cell:

```
With G2antt1.Items
   .CellUnderline(.FocusItem, 0) = True
End With
```

The following C++ sample underlines the focused cell:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetCellUnderline( COleVariant( items.GetFocusItem() ), COleVariant( (long)0 ), TRUE );
```

The following C# sample underlines the focused cell:

```
axG2antt1.Items.set_CellUnderline(axG2antt1.Items.FocusItem, 0, true);
```

The following VB.NET sample underlines the focused cell:

```
With AxG2antt1.Items
   .CellUnderline(.FocusItem, 0) = True
```

```
End With
```

The following VFP sample underlines the focused cell:

```
with thisform.G2antt1.Items
   .DefaultItem = .FocusItem
   .CellUnderline(0, 0 ) = .t.
endwith
```

Note: A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell. Here's few hints how to use properties with Item and ColIndex parameters:

```
G2antt1.Items.CellBold(, G2antt1.Items.ItemCell(G2antt1.Items(0), 0)) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), 0) = True
```

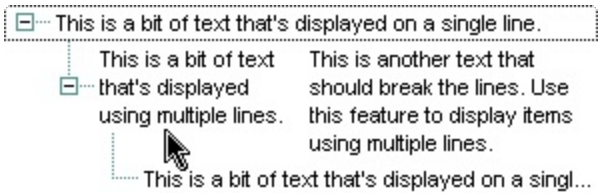
```
G2antt1.Items.CellBold(G2antt1.Items(0), "ColumnName") = True
```

property Items.CellVAlignment ([Item as Variant], [ColIndex as Variant]) as VAlignmentEnum

Retrieves or sets a value that indicates how the cell's caption is vertically aligned.

Type	Description
Item as Variant	A long expression that identifies the item's handle
ColIndex as Variant	A long expression that indicates the column's index or the cell's handle, a string expression that indicates the column's caption.
VAlignmentEnum	A VAlignmentEnum expression that indicates the cell's vertically alignment.

Use the CellVAlignment property to specify the vertically alignment for the cell's caption. Use the CellSingleLine property to specify whether a cell uses single or multiple lines. Use the CellHAlignment property to align horizontally the cell. The +/- button is aligned accordingly to the cell's caption. Use the Def(exCellVAlignment) property to specify the same vertical alignment for the entire column.



The following VB sample aligns the focused cell to the bottom:

```
With G2antt1.Items
   .CellVAlignment(.FocusItem, 0) = VAlignmentEnum.BottomAlignment
End With
```

The following C++ sample right aligns the focused cell:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetCellVAlignment( COleVariant( items.GetFocusItem() ), COleVariant( (long)0 ), 2
/*BottomAlignment*/ );
```

The following VB.NET sample right aligns the focused cell:

```
With AxG2antt1.Items
   .CellVAlignment(.FocusItem, 0) = EXG2ANTTLib.VAlignmentEnum.BottomAlignment
```

```
End With
```

The following C# sample right aligns the focused cell:

```
axG2antt1.Items.set_CellVAlignment(axG2antt1.Items.FocusItem, 0,
EXG2ANTTLib.VAlignmentEnum.BottomAlignment);
```

The following VFP sample right aligns the focused cell:

```
with thisform.G2antt1.Items
    .DefaultItem = .FocusItem
    .CellVAlignment(0,0) = 2 && BottomAlignment
endwith
```

property Items.CellValue([Item as Variant], [ColIndex as Variant]) as Variant

Specifies the cell's value.

Type	Description
Item as Variant	A long expression that indicates the item's handle. During the ValidateValue event, you can uses -1 instead Item, to access to the modified value. In other words during ValidateValue event, the Items.CellValue(Item,ColIndex) and Items.CellCaption(Item,ColIndex) properties retrieve the original value/caption of the cell while the Items.CellValue(-1,ColIndex) and Items.CellCaption(-1,ColIndex) gets the modified value of the specified cell.
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key. If the Item parameter is missing or it is zero (0), the ColIndex parameter is the handle of the cell being accessed.
Variant	A variant expression that indicates the cell's value. The cell's value supports built-in HTML format if the CellValueFormat property is exHTML.

Use the CellValue property to specify the value for cells in the second, third columns and so on. The Change event is called when the user changes the CellValue property. Use the CellData property to associate an user data to a cell. The AddItem or InsertItem method may specify the value for the first cell. Use the LockedItemCount property to lock or unlock items to the top or bottom side of the control. Use the ItemCell property to get the cell's handle based on the item and the column. Use the CellItem property to get the handle of the item that's the owner of the cell. Use the SplitCell property to split a cell. If the CauseValidateValue property is True, the control fires the ValidateValue property when the user changes the CellValue property. Use the AddItem method to add new predefined values to a drop down list editor. Use the CellEditor property to assign an editor to a single cell. Use the Editor property to assign the same editor to all cells in the column. Use the Add method to add new columns to the control. Use the **<img>** HTML tag to insert icons inside the cell's caption, if the CellValueFormat property is exHTML. Use the FormatColumn property to format the column. The AllowCellValueToItemBar property allows the cells to display properties of the bars.

The CellValue property indicates the formula being used to compute the field, if the CellValueFormat property is exComputedField. The ComputedField property specifies the

formula to compute the entire column.

The cell shows its text based on the CellValueFormat property as follows:

- **exText**, the CellValue indicates the text to be displayed without HTML formatting
- **exHTML**, the CellValue indicates the text to be displayed with HTML formatting, such as <b> to bold a portion of text.
- **exComputedField**, the CellValue property indicates a formula to display the cell's content based on the values of any cell in the current item. For instance, the %1 + %2 + %3 adds or concatenates the values from first 3 cells. The exComputedField can be combined with exHTML that indicates that the computed field may display HTML format. The ComputedField property specifies the formula to compute the entire column.

The CellValue property of the cell is being shown as:

- formatted using the FormatCell property, if it is valid
- formatted using the FormatColumn property, if it is valid

In other words, all cells applies the format of the FormatColumn property, excepts the cells with the FormatCell property being set. If the cell belongs to a column with the FireFormatColumn property on True, the Value parameter of the FormatColumn event shows the newly caption for the cell to be shown

The following VB sample displays an HTML cell on multiple lines:

```
With G2antt1.Items
    Dim h As HITEM
    h = .AddItem("Cell 1")
    .CellValue(h, 1) = "<r><dotline><b>HTML support</b><br>This is a bit of text
where built-in <b>HTML</b> support is enabled."
    .CellValueFormat(h, 1) = exHTML
    .CellSingleLine(h, 1) = False
    .CellEditorVisible(h, 1) = False
End With
```

The following C++ changes the value of the focused cell:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
COleVariant vtItem( items.GetFocusItem() ), vtColumn(
long(m_g2antt.GetFocusColumnIndex()) );
```

```
items.SetCellValue( vtItem, vtColumn, COleVariant("new value") );
```

The following VB.NET changes the value of the focused cell:

```
With AxG2antt1.Items
   .CellValue(.FocusItem, AxG2antt1.FocusColumnIndex) = "new value"
End With
```

The following C# changes the value of the focused cell:

```
axG2antt1.Items.set_CellValue(axG2antt1.Items.FocusItem, axG2antt1.FocusColumnIndex,
"new value");
```

The following VFP changes the value of the focused cell:

```
with thisform.G2antt1.Items
   .DefaultItem = .FocusItem
   .CellValue(0,thisform.G2antt1.FocusColumnIndex) = "new value"
endwith
```

You may include strings like [m¸], [mł], [180ş], źml, or ˝m¸, žmł, and so on. Copy the symbol from this page, and paste to your cell.

Note: The intersection of an item with a column defines a cell. Each cell is uniquely represented by its handle. The cell's handle is of HCELL type, that's equivalent with a long type. All properties of Items object that have two parameters *Item* and *ColIndex,* refer a cell.

The following lines are equivalents and each of them changes the bold font attribute of the first cell on the first item.

```
With G2antt1
   .Items.CellBold(, .Items.ItemCell(.Items(0), 0)) = True
   .Items.CellBold(.Items(0), 0) = True
   .Items.CellBold(.Items(0)) = True
   .Items.CellBold(.Items.ItemByIndex(0)) = True
   .Items.CellBold(.Items.ItemByIndex(0), 0) = True
   .Items.CellBold(.Items(0), G2antt1.Columns(0).Caption) = True
End With
```

property Items.CellValueFormat([Item as Variant], [ColIndex as Variant]) as ValueFormatEnum

Specifies how the cell's caption is displayed.

Type	Description
Item as Variant	A long expression that indicates the item's handle
ColIndex as Variant	A long expression that indicates the column's index, or a string expression that indicates the column's caption or column's key.
ValueFormatEnum	A long expression that defines the way how the cell's value is displayed

The component supports built-in HTML format. That means that you can use HTML tags when displays the cell's value . By default, the CellValueFormat property is exText. If the CellValueFormat is exText, the cell displays the CellValue property like it is. If the CellValueFormat is exHTML, the cell displays the CellValue property using the HTML tags specified in the ValueFormatEnum type. Use the Def property to specify whether all cells in the column display HTML format. Use the CellVAlignment property to align vertically a cell.

The CellValue property of the cell is being shown as:

- formatted using the FormatCell property, if it is valid
- formatted using the FormatColumn property, if it is valid

In other words, all cells applies the format of the FormatColumn property, excepts the cells with the FormatCell property being set. If the cell belongs to a column with the FireFormatColumn property on True, the Value parameter of the FormatColumn event shows the newly caption for the cell to be shown

method Items.CellValueToItemBar (Item as Variant, ColIndex as Variant, PropertyBar as ItemBarPropertyEnum, [BarKey as Variant])

Indicates whether the cell displays the specified property of the bar.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key. If the Item parameter is missing or it is zero (0), the ColIndex parameter is the handle of the cell being accessed.
PropertyBar as ItemBarPropertyEnum	An ItemBarPropertyEnum expression that specifies the property of the bar being displayed. If -1, the cell has no bar associated, or in other words removes the association between a cell and a bar. Though, if the CellValueToItemBar is not called, the Def(exCellValueToItemBarProperty) property defines the property of the bar being displayed, if it is greater than 0
BarKey as Variant	A VARIANT expression that specifies the key of the associated bar. If the CellValueToItemBar is not called, the Def(exCellValueToItemBarKey) property defines the key of the associated bar

The CellValueToItemBar method can be used to associate particular cells with bars in the items. The Def(exCellValueToItemBarProperty/exCellValueToItemBarKey) property of the Column object defines a relation/association between specified property bar and the cells in the column. Once an association between a cell and a bar is made, the CellValue property and ItemBar property returns the same result, or in other words, changing the cell's value will be reflected in the bar's property, and back, so changing the bar's property will change the cell's value.

property Items.CellWidth([Item as Variant], [ColIndex as Variant]) as Long

Retrieves or sets a value that indicates the width of the inner cell.

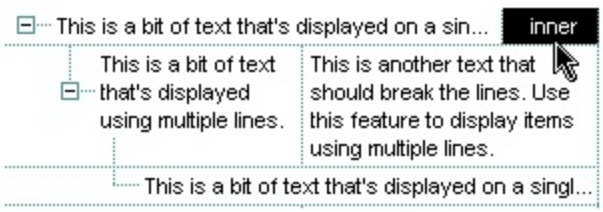
Type	Description
Item as Variant	A long expression that indicates the handle of the item where the cell is, or 0. If the Item parameter is 0, the ColIndex parameter must indicate the handle of the cell.
ColIndex as Variant	A long expression that indicates the index of the column where a cell is divided, or a long expression that indicates the handle of the cell being divided, if the Item parameter is missing or it is zero.
Long	A long expression that indicates the width of the cell.

The CellWidth property specifies the cell's width. The CellWidth property has effect only if the cell contains inner cells. The SplitCell method splits a cell in two cells (the newly created cell is called inner cell). Use the InnerCell property to get the inner cell. Use the CellParent property to get the parent of the inner cell. Use the CellItem property to get the item that's the owner of the cell. Use the BeginUpdate and EndUpdate methods to refresh the cell's width when changing it on the fly.

The CellWidth property specifies the width of the cell, where the cell is divided in two or multiple (inner) cells like follows:

- if the CellWidth property is less than zero, the master cell calculates the width of the inner cell, so all the inner cells with CellWidth less than zero have the same width in the master cell.
- if the CellWidth property is greater than zero, it indicates the width in pixels of the inner cell.

By default, the CellWidth property is -1, and so when the user splits a cell the inner cell takes the right half of the area occupied by the master cell.



The following VB sample splits the first visible cell in three cells:

```
With G2antt1
  .BeginUpdate
  .DrawGridLines = exAllLines
```

```
  With .Items
    Dim h As HITEM, f As HCELL
    h = .FirstVisibleItem
    f = .ItemCell(h, 0)
    f = .SplitCell(, f)
    .CellValue(, f) = "Split 1"
    f = .SplitCell(, f)
    .CellValue(, f) = "Split 2"
  End With
  .EndUpdate
End With
```

The following VB sample specifies that the inner cell should have 32 pixels:

```
With G2antt1
  .BeginUpdate
  .DrawGridLines = exAllLines
  With .Items
    Dim h As HITEM, f As HCELL
    h = .FirstVisibleItem
    f = .ItemCell(h, 0)
    f = .SplitCell(, f)
    .CellValue(, f) = "Split"
    .CellWidth(, f) = 32
  End With
  .EndUpdate
End With
```

The following VB sample adds an inner cell to the focused cell with 48 pixels width:

```
G2antt1.BeginUpdate
With G2antt1.Items
  Dim h As Long
  h = .SplitCell(.FocusItem, 0)
  .CellBackColor(, h) = vbBlack
  .CellForeColor(, h) = vbWhite
  .CellHAlignment(, h) = CenterAlignment
  .CellValue(, h) = "inner"
```

```
    .CellWidth(, h) = 48
End With
G2antt1.EndUpdate
```

The following C++ sample adds an inner cell to the focused cell with 48 pixels width:

```
#include "Items.h"
m_g2antt.BeginUpdate();
CItems items = m_g2antt.GetItems();
COleVariant vtItem( items.GetFocusItem() ), vtColumn( long(0) ), vtMissing; V_VT(
&vtMissing ) = VT_ERROR;
COleVariant vtInner = items.GetSplitCell( vtItem, vtColumn );
items.SetCellWidth( vtMissing, vtInner, 48 );
items.SetCellBackColor( vtMissing, vtInner, 0 );
items.SetCellForeColor( vtMissing, vtInner, RGB(255,255,255) );
items.SetCellValue( vtMissing, vtInner, COleVariant("inner") );
items.SetCellHAlignment( vtMissing, vtInner, 1 );
m_g2antt.EndUpdate();
```

The following VB.NET sample adds an inner cell to the focused cell with 48 pixels width:

```
With AxG2antt1
    .BeginUpdate()
    With .Items
        Dim iInner As Integer
        iInner = .SplitCell(.FocusItem, 0)
        .CellValue(, iInner) = "inner"
        .CellHAlignment(, iInner) = EXG2ANTTLib.AlignmentEnum.CenterAlignment
        .CellWidth(, iInner) = 48
        .CellBackColor(, iInner) = 0
        .CellForeColor(, iInner) = ToUInt32(Color.White)
    End With
    .EndUpdate()
End With
```

The following C# sample adds an inner cell to the focused cell with 48 pixels width:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
axG2antt1.BeginUpdate();
```

```
object iInner = items.get_SplitCell(axG2antt1.Items.FocusItem, 0);
items.set_CellValue(null, iInner, "inner");
items.set_CellHAlignment(null, iInner, EXG2ANTTLib.AlignmentEnum.CenterAlignment);
items.set_CellBackColor(null, iInner, ToUInt32(Color.Black));
items.set_CellForeColor(null, iInner, ToUInt32(Color.White));
items.set_CellWidth(null, iInner, 48);
axG2antt1.EndUpdate();
```

property Items.ChildCount (Item as HITEM) as Long

Retrieves the number of children items.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
Long	A long value that indicates the number of child items.

Use the ChildCount property checks whether an item has child items. Use the ItemChild property to get the first child item, if there is one, 0 else. Use the ItemHasChildren property to specify whether the item should display a +/- sign even if it contains no child items.

method Items.ClearBars (Item as HITEM)

Clears the bars from the item.

Type	Description
Item as HITEM	A long expression that indicates the the handle of the item where the bars are removed. If the Item parameter is 0, the ClearBars method removes all bars from all items. In this case the DefaultItem property should be 0 (by default), else it refers a single item being indicated by the DefaultItem property.

Use the ClearBars method to remove all bars in the specified item. *If the Item parameter is not 0 (indicates a valid handle), the ClearBars removes only bars in the specified item. If the Item parameter is 0, the ClearBars method removes all bars from all items, in other words from the entire chart.* Use the BeginUpdate / EndUpdate methods to refresh the control's content after removing a bar or several bars.

Use the RemoveBar method to remove a bar from an item. Use the Remove method to remove a type of bar from the Bars collection. Use the Add method to add new types of bars to the Bars collection. Use the FirstVisibleDate property to specify the first visible date in the chart area. Use the Key parameter to identify a bar inside an item. Use the ItemBar property to access a bar inside the item. Use the PaneWidth property to specify the width of the chart. Use the NonworkingDays property to specify the non-working days.

method Items.ClearCellBackColor ([Item as Variant], [ColIndex as Variant])

Clears the cell's background color.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index or the cell's handle, a string expression that indicates the column's caption.

The ClearCellBackColor method clears the cell's background color when the CellBackColor property is used. Use the BackColor property to specify the control's background color.

method Items.ClearCellForeColor ([Item as Variant], [ColIndex as Variant])

Clears the cell's foreground color.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index or the cell's handle, a string expression that indicates the column's caption.

The ClearCellForeColor method clears the cell's foreground color when CellForeColor property was used.

method Items.ClearCellHAlignment ([Item as Variant], [ColIndex as Variant])

Clears the cell's alignment.

Type	Description
Item as Variant	A long expression that indicates the handle of the item.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's key or the column's caption.

Use the ClearCellHAlignment method to clear the alignment of the cell's caption previously set using the CellHAlignment property. If the CellHAlignment property is not called, the Alignment property of the Column object specifies the alignment of the cell's caption.

method Items.ClearItemBackColor (Item as HITEM)

Clears the item's background color.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.

The ClearItemBackColor method clears the item's background color when ItemBackColor property is used (columns/items part only). The ClearItemBackColor method clears the item's background color when ItemBackColor property is used (chart part only).

method Items.ClearItemForeColor (Item as HITEM)

Clears the item's foreground color.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.

The ClearItemForeColor method clears the item's foreground color when ItemForeColor property is used. Use the ForeColor property to change the control's foreground color.

method Items.ClearLinks ()

Clears all links in the chart.

Type	Description

Use the ClearLinks method to remove all links in the control. Use the ShowLinks property to hide all links in the control. Use the RemoveLink method to remove a specified link. Use the AddLink method to add a link between two bars. Use the RemoveAllItems method to remove all items in the control. Use the RemoveItem method to remove an item. The RemoveItem method removes all links related to the item. Use the RemoveLinksOf method to remove all links that start or end on the specified bar.

property Items.ComputeValue ([Expression as Variant], [Item as Variant], [ColIndex as Variant], [ValueFormatType as Variant]) as Variant

Computes the value of a specified formula.

Type	Description
Expression as Variant	A string expression that specifies the formula to compute
Item as Variant	A long expression that specifies the handle of the item.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's key or the column's caption.
ValueFormatType as Variant	A ValueFormatType expression that indicates the type of the formula being interpreted by the Expression parameter. For instance, if the ValueFormatType parameter is exTotalField, the Expression parameter should inidcate a total formula of type aggregate(list,direction,formula)
Variant	A string expression that indicates the result.

The ComputeValue property gets the result of a a computed or total field. The Item and ColIndex property refers the cells used as the source for the formula. Use the ComputeValue property to get the result of a total field. For instance, for a total field, the CellValue property indicates the formula, while the ComputeValue can be used to get the result of the formula at runtime.

The ComputeValue method returns the:

- value of the computed field, where the ValueFormatType is exComputedField, and the Expression indicates the formula for the computed field.
- value of the total field, where the ValueFormatType is exTotalField, and the Expression indicates a string as: aggregate(list,direction,formula)
- text with no HTML formatting, where the ValueFormatType is exHTML, and the Expression indicates the string including the HTML format.

For instance, based on the ValueFormatType and Expression parameters the result could be:

- exComputedField, dbl(%0) + dbl(%1), the sum between first two cells in the item referred by Item.
- exTotalField, sum(current,dir,dbl(%0) + dbl(%1)), the total of first two columns, for all direct child items of the item being referred by Item.
- exHTML, bold, returns bold (returns the result with no HTML formatting). In

this case, the Item and ColIndex have no effect.

property Items.DefaultItem as HITEM

Retrieves or sets the default item's handle.

Type	Description
HITEM	A long expression that indicates the handle of the item that's used by all properties of the Items object, that have a parameter Item.

The property is used in VFP implementation. The VFP fires "Invalid Subscript Range" error, while it tries to process a number grater than 65000. Since, the HITEM is a long value that most of the time exceeds 65000, the VFP users have to use this property, instead passing directly the handles to properties.

The following sample shows to change the cell's image:

```
.Items.DefaultItem = .Items.AddItem("Item 1")
.Items.CellImage(0,1) = 2
```

In VFP the following sample fires: "Invalid Subscript Range":

```
i = .Items.AddItem("Item 1")
.Items.CellImage(i,1) = 2
```

because the i variable is grater than 65000, and the VFP thinks that the CellImage is an array, but it is not. It is a property. Hope that future versions will correct this problem in VFP.

So, if you pass zero to a property that has a parameter titled Item, the control takes instead the DefaultItem value.

Let's say that your code looks like follows:

```
LOCAL h
SCAN
   _key="K_"+ALLTRIM(STR(projekte.ID))
   WITH THISFORM.myplan.Items
      h = .AddItem(ALLTRIM(projekte.project_name))
      .AddBar( h,"Project Summary" , DTOT(projekte.sdate),DTOT(projekte.edate), _key, "" )
      .ItemBar( h ,_key,3 ) = "my text"
   ENDWITH
ENDSCAN
```

The h variable indicates the handle of the newly created item. This value is always greater than 65000, so the VFP environment always fires an error when compiling the AddBar and ItemBar properties because it considers accessing an array, and its limit is 65000. Of course this problem is related to VFP ignoring the fact that it is calling a property! not an array, so our products provide a DefaultItem property that help VFP users to pass this error. So, in VFP the above code should look like follows:

```
SCAN
   _key="K_"+ALLTRIM(STR(projekte.ID))
   WITH THISFORM.myplan.Items
      .DefaultItem = .AddItem(ALLTRIM(projekte.project_name))
      .AddBar( 0,"Project Summary" , DTOT(projekte.sdate),DTOT(projekte.edate),_key, "" )
      THISFORM.myplan.Template = "Items.ItemBar( 0,`" + _key + "`,3 ) = `my text`"
   ENDWITH
ENDSCAN
```

The difference (marked in red) is that the first parameter for properties like AddBar and ItemBar is 0, and before calling them the Items.DefaultItem property indicates the handle of the item being accessed. How it works? The control uses the value of the Items.DefaultItem property, when the first parameter of the ItemBar, AddBar and so on is 0. The AddItem property saves before the handle of the newly created item to the DefaultItem property, and so the VFP error is gone, and the code works like you expect.

method Items.DefineSummaryBars (SummaryItem as HITEM, SummaryKey as Variant, ItemAdd as HITEM, KeyAdd as Variant)

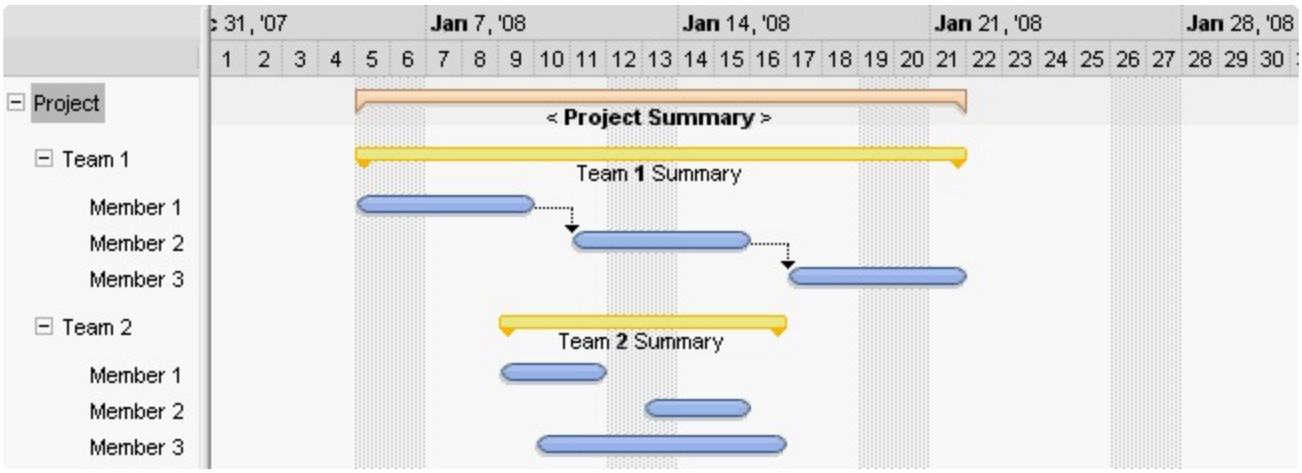
Defines the bars that belongs to a summary bar.

Type	Description
SummaryItem as HITEM	A long expression that specifies the handle of the item that displays the summary bar.
SummaryKey as Variant	A VARIANT expression that indicates the key of the summary bar
ItemAdd as HITEM	A long expression that specifies the item that holds the bar being included in the summary bar. The ItemAdd parameter could be • a valid handle, indicating the item itself • 0 indicates all items • -1 indicates the direct descendents/children items of the SummaryItem (child items of the SummaryItem) • -2 means leaf descendents/items of the SummaryItem, where a leaf or terminal item is an item with no child items • -3 means all descendents/children items of the SummaryItem (recursively) For instance, DefineSummaryBars(SummaryItem,SummaryKey,-1,"<K*>") defines the summary bar to include bars of SummaryItem descendents, whose key starts with character K, where the DefineSummaryBars(SummaryItem,SummaryKey,0,"K") defines the summary bar to include all bars with the key K from the entire chart. The 0, -1, -2 and -3 values are supported, starting from the version 12.0
	A VARIANT expression that indicates the key of the bar being included in the summary bar. The KeyAdd parameter supports pattern if specified such as "<pattern>", where the pattern may contain wild card characters such as '?' for any single character, '*' for zero or more occurrences of any character, '#' for any digit

KeyAdd as Variant	character. For instance, DefineSummaryBars(,,,"<K*>") defines the summary bar to include bars whose key starts with character K. The "<pattern>" syntax is supported, starting from the version 12.0

The DefineSummaryBars method defines bars being displayed under a summary bar. Once a bar that's included in a summary bar is moved or resized, its summary bar is automatically updated. Once a summary bar is moved all included bars are moved too. For instance, if your chart displays a "Summary" or "Project Summary" predefined bar, you can use the DefineSummaryBars method to define the bars included in the summary bar, so they automatically update the summary bars when moving or resizing. The DefineSummaryBars method defines a group of bars that belongs to another bar (called summary bar), so the margins of the summary bars are min and max of the margins of included bars. The margins of the bars are determined by ItemBar(exBarStart) and ItemBar(exBarEnd). The UndefineSummaryBars method does the reverse operation, as it removes a bar from a summary bar. Use the GroupBars method to group one or more bars. The ItemBar(exSummaryBarBackColor) property specifies the background color for the child bars in the summary bar portion.

For instance, in the following screen shot, the "< Project Summary >" is a summary bar for "Team 1 Summary" and "Team 2 Summary". The "Team 1 Summary" is a summary bar for all child bas being displayed under the Team 1 item. The "Team 2 Summary" is a summary bar for all child bas being displayed under the Team 2 item. Once a bar is moved, the owner summary bar is updated accordingly.



The following VB sample adds a "Summary" bar that includes 2 "Task" bars:

```
With G2antt1
  .BeginUpdate
```

```
   .Columns.Add "Tasks"
   With .Chart
      .FirstVisibleDate = #6/20/2005#
      .LevelCount = 2
   End With
   With .Items
      h = .AddItem("Project")
      .AddBar h,"Summary",#6/22/2005#,#6/23/2005 4:00:00 PM#
      h1 = .InsertItem(h,0,"Task 1")
      .AddBar h1,"Task",#6/21/2005 4:00:00 PM#,#6/23/2005#
      .ItemBar(h1,"",exBarHAlignCaption) = 18
      .DefineSummaryBars h,"",h1,""
      h2 = .InsertItem(h,0,"Task 2")
      .AddBar h2,"Task",#6/23/2005 8:00:00 AM#,#6/25/2005#
      .DefineSummaryBars h,"",h2,""
      .ExpandItem(h) = True
   End With
   .EndUpdate
End With
```

The following VB.NET sample adds a "Summary" bar that includes 2 "Task" bars:

```
Dim h,h1,h2
With AxG2antt1
   .BeginUpdate
   .Columns.Add "Tasks"
   With .Chart
      .FirstVisibleDate = #6/20/2005#
      .LevelCount = 2
   End With
   With .Items
      h = .AddItem("Project")
      .AddBar h,"Summary",#6/22/2005#,#6/23/2005 4:00:00 PM#
      h1 = .InsertItem(h,0,"Task 1")
      .AddBar h1,"Task",#6/21/2005 4:00:00 PM#,#6/23/2005#
      .ItemBar(h1,"",EXG2ANTTLib.ItemBarPropertyEnum.exBarHAlignCaption) = 18
      .DefineSummaryBars h,"",h1,""
```

```
      h2 = .InsertItem(h,0,"Task 2")
      .AddBar h2,"Task",#6/23/2005 8:00:00 AM#,#6/25/2005#
      .DefineSummaryBars h,"",h2,""
      .ExpandItem(h) = True
   End With
   .EndUpdate
End With
```

The following C# sample adds a "Summary" bar that includes 2 "Task" bars:

```
axG2antt1.BeginUpdate();
axG2antt1.Columns.Add("Tasks");
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.FirstVisibleDate = "6/20/2005";
   var_Chart.LevelCount = 2;
EXG2ANTTLib.Items var_Items = axG2antt1.Items;
   int h = var_Items.AddItem("Project");
   var_Items.AddBar(h,"Summary","6/22/2005","6/23/2005 4:00:00 PM",null,null);
   int h1 = var_Items.InsertItem(h,0,"Task 1");
   var_Items.AddBar(h1,"Task","6/21/2005 4:00:00 PM","6/23/2005",null,null);

var_Items.set_ItemBar(h1,"",EXG2ANTTLib.ItemBarPropertyEnum.exBarHAlignCaption,18);
   var_Items.DefineSummaryBars(h,"",h1,"");
   int h2 = var_Items.InsertItem(h,0,"Task 2");
   var_Items.AddBar(h2,"Task","6/23/2005 8:00:00 AM","6/25/2005",null,null);
   var_Items.DefineSummaryBars(h,"",h2,"");
   var_Items.set_ExpandItem(h,true);
axG2antt1.EndUpdate();
```

The following C++ sample adds a "Summary" bar that includes 2 "Task" bars:

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import "d:\\Exontrol\\ExG2antt\\project\\Demo\\ExG2antt.dll"
   using namespace EXG2ANTTLib;
*/
```

```
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
spG2antt1->GetColumns()->Add(L"Tasks");
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutFirstVisibleDate("6/20/2005");
   var_Chart->PutLevelCount(2);
EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems();
   long h = var_Items->AddItem("Project");
   var_Items->AddBar(h,"Summary","6/22/2005","6/23/2005 4:00:00
PM",vtMissing,vtMissing);
   long h1 = var_Items->InsertItem(h,long(0),"Task 1");
   var_Items->AddBar(h1,"Task","6/21/2005 4:00:00
PM","6/23/2005",vtMissing,vtMissing);
   var_Items->PutItemBar(h1,"",EXG2ANTTLib::exBarHAlignCaption,long(18));
   var_Items->DefineSummaryBars(h,"",h1,"");
   long h2 = var_Items->InsertItem(h,long(0),"Task 2");
   var_Items->AddBar(h2,"Task","6/23/2005 8:00:00
AM","6/25/2005",vtMissing,vtMissing);
   var_Items->DefineSummaryBars(h,"",h2,"");
   var_Items->PutExpandItem(h,VARIANT_TRUE);
spG2antt1->EndUpdate();
```

The following VFP sample adds a "Summary" bar that includes 2 "Task" bars:

```
with thisform.G2antt1
   .BeginUpdate
   .Columns.Add("Tasks")
   with .Chart
      .FirstVisibleDate = {^2005-6-20}
      .LevelCount = 2
   endwith
   with .Items
      h = .AddItem("Project")
      .AddBar(h,"Summary",{^2005-6-22},{^2005-6-23 16:00:00})
      h1 = .InsertItem(h,0,"Task 1")
      .AddBar(h1,"Task",{^2005-6-21 16:00:00},{^2005-6-23})
```

```
    .ItemBar(h1,"",4) = 18
    .DefineSummaryBars(h,"",h1,"")
    h2 = .InsertItem(h,0,"Task 2")
    .AddBar(h2,"Task",{^2005-6-23 8:00:00},{^2005-6-25})
    .DefineSummaryBars(h,"",h2,"")
    .ExpandItem(h) = .T.
  endwith
  .EndUpdate
endwith
```

property Items.DefSchedulePDM(Option as DefSchedulePDMEnum) as Variant

Retrieves or sets an option for SchedulePDM method.

Type	Description
Option as DefSchedulePDMEnum	A DefSchedulePDMEnum expression that indicates the option to be changed.
Variant	A Variant expression that indicates the value of the SchedulePDM's option to be changed.

The Def SchedulePDM property defines options to be used by the SchedulePDM method. If required any option to be used the DefSchedulePDM should be called before the SchedulePDM method else it will have no effect. For instance, use the Def SchedulePDM property to specify a start date for the project, so the SchedulePDM method will use it, to arrange all bars so no bars will start before the specified date. The same if you require to specify the end of the project.

For instance the following sample specifies the start of the project to be 1/8/2001,

```
With G2antt1
   With .Items
      .DefSchedulePDM(exPDMScheduleType) = 1
      .DefSchedulePDM(exPDMScheduleDate) = #1/8/2001#
     .SchedulePDM 0,"K1"
   End With
End With
```

and the following sample specifies the end of the project to be 1/8/2001

```
With G2antt1
   With .Items
      .DefSchedulePDM(exPDMScheduleType) = 2
      .DefSchedulePDM(exPDMScheduleDate) = #1/8/2001#
     .SchedulePDM 0,"K1"
   End With
End With
```

method Items.DeleteCellEditor ([Item as Variant], [ColIndex as Variant])

Deletes the cell's built-in editor created by CellEditor property.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index, or cell's handle, a string expression that indicates the column's caption or key.

Use the DeleteCellEditor method to delete the editor created using the CellEditor property. Use the CellEditorVisible property to hide or show the cell's editor. Use the HasCellEditor property to check whether the cell contains an editor (being created using the CellEditor property). The DeleteCellEditor method has no effect if the cell contains an editor assigned using the the Editor property of the Column object, or the cell has no editor.

property Items.EnableItem(Item as HITEM) as Boolean

Returns or sets a value that determines whether a item can respond to user-generated events.

Type	Description
Item as HITEM	A long expression that indicates the item's handle that is enabled or disabled.
Boolean	A boolean expression that indicates whether the item is enabled or disabled.

Use the EnableItem property to disable an item. A disabled item looks grayed and it is selectable. Use the SelectableItem property to specify the user can select an item. Once that an item is disabled all the cells of the item are disabled, so CellEnabled property has no effect. To disable a column you can use Enabled property of a Column object. The control prevents creating new bars inside disable items, so you can not create new bars in disabled items, if the AllowCreateBar property of the Chart is not 0. A disabled item looks grayed, and shows as unselected.

method Items.EndBlockUndoRedo ()

Ends recording the UI operations and adds the undo/redo operations as a block, so they all can be restored at once, if Undo method is performed.

Type	Description

The StartBlockUndoRedo method starts recording the UI operations as a block on undo/redo operations (equivalent of EndBlockUndoRedo method of the Chart object). The method has effect only if the AllowUndoRedo property is True. The EndBlockUndoRedo method collects all undo/redo operations since StartBlockUndoRedo method was called and add them to the undo/redo queue as a block. This way the next call on a Undo operation, the entire block is restored, so all UI operations are restored. For instance, if you have a procedure that moves several bars, and want all of them being grouped, you can use StartBlockUndoRedo to start recording the operations as a block, and call the EndBlockUndoRedo when procedure ends, so next call of an undo operation the bars are restored to their original position. The EndBlockUndoRedo method must be called the same number of times as the StartBlockUndoRedo method was called. For instance, if you have called the StartBlockUndoRedo twice the EndBlockUndoRedo method must be called twice too, and the collected operations are added to the chart's queue of undo/redo operations at the end.

method Items.EndUpdateBar (StartUpdateBar as Long)

Adds programmatically updated properties of the bar to undo/redo queue.

Type	Description
StartUpdateBar as Long	A long expression that specifies the handle being returned by the StartUpdateBar property.

Use the StartUpdateBar and EndUpdateBar methods to add new entries in the chart's undo/redo queue for properties of the bar being updated by code. The ItemBar property accesses the properties of the bar. For instance, if your application provides UI dialogs or forms that help users changing the properties of the selected bar such as color, text, tooltips and so on, you can provide undo/redo operations for them by using the StartUpdateBar and EndUpdateBar methods. Shortly, the StartUpdateBar method starts recording the properties being changed until the EndUpdateBar method is called. The EndUpdateBar method actually adds a new entry to the undo/redo queue based on the changed properties. If there were no changes of the bar during the Star/End session, no new entry is added. The EndUpdateBar method adds UpdateBar entries to the undo/redo queue.

The AllowUndoRedo property specifies whether the chart supports undo/redo operations for objects in the chart such as bars or links. The ChartStartChanging(exUndo/exRedo) / ChartEndChanging(exUndo/exRedo) event notifies your application whenever an Undo/Redo operation is performed. The UndoListAction property lists the Undo actions that can be performed in the chart. The RedoListAction property lists the Redo actions that can be performed in the chart.

The following VB sample adds a new entry "UpdateBar" in the chart's undo/redo queue for changing the text of the bar (/COM version):

```
With G2antt1.Items
  Dim hItem As Long
  hItem = .FocusItem
  Dim barKey As Variant
  barKey = .FirstItemBar(hItem)

  Dim iChangeBar As Long
  iChangeBar = .StartUpdateBar(hItem, barKey)
  .ItemBar(hItem, barKey, exBarCaption) = "new caption"
  .EndUpdateBar (iChangeBar)
End With
```

The following VB/NET sample adds a new entry "UpdateBar" in the chart's undo/redo queue for changing the text of the bar (/NET Assembly version):

```
With Exg2antt1.Items
    Dim hItem As Long = .FocusItem
    Dim barKey As Object = .get_FirstItemBar(hItem)

    Dim iChangeBar As Long = .get_StartUpdateBar(hItem, barKey)
    .set_ItemBar(hItem, barKey, exontrol.EXG2ANTTLib.ItemBarPropertyEnum.exBarCaption,
"new caption")
    .EndUpdateBar(iChangeBar)
End With
```

These samples add new entries to undo/redo queue as : "UpdateBar;94980832;B1;3;;new caption " which indicates , the handle of the items where the bar has been changed, the bar of the key as being B1, the 3 indicates the exBarCaption predefined value, and so on. Once the sample is called, the bar's caption is changed, and using the CTRL + Z, you can restore back the old value, or pressing the CTRL + Y you can change back after restoring.

method Items.EndUpdateLink (StartUpdateLink as Long)

Adds programmatically updated properties of the link to undo/redo queue.

Type	Description
StartUpdateLink as Long	A long expression that indicates the handle being returned by StartUpdateLink property.

Use the StartUpdateLink and EndUpdateLink methods to add new entries in the chart's undo/redo queue for properties of the link being updated by code. The Link property accesses the properties of the link. For instance, if your application provides UI dialogs or forms that help users changing the properties of the selected link such as color, text, tooltips and so on, you can provide undo/redo operations for them by using the StartUpdateLink and EndUpdateLink methods. Shortly, the StartUpdateLink method starts recording the properties being changed until the EndUpdateLink method is called. The EndUpdateLink method actually adds a new entry to the undo/redo queue based on the changed properties. If there were no changes of the link during the Star/End session, no new entry is added. The EndUpdateLink method adds UpdateLink entries to the undo/redo queue.

The AllowUndoRedo property specifies whether the chart supports undo/redo operations for objects in the chart such as bars or links. The ChartStartChanging(exUndo/exRedo) / ChartEndChanging(exUndo/exRedo) event notifies your application whenever an Undo/Redo operation is performed. The UndoListAction property lists the Undo actions that can be performed in the chart. The RedoListAction property lists the Redo actions that can be performed in the chart.

The following VB sample adds a new entry "UpdateLink" in the chart's undo/redo queue for changing the text being displayed on the link (/COM version):

```
With G2antt1.Items
   Dim linkKey As Variant
   linkKey = .FirstLink
   Dim iChangeLink As Long
   iChangeLink = .StartUpdateLink(linkKey)
   .Link(linkKey, exLinkText) = "new text"
   .EndUpdateLink (iChangeLink)
End With
```

The following VB/NET sample adds a new entry "UpdateLink" in the chart's undo/redo queue for changing the text being displayed on the link (/NET Assembly version):

```
With Exg2antt1.Items
    Dim linkKey As Object = .get_FirstLink
    Dim iChangeLink As Long = .get_StartUpdateLink(linkKey)
    .set_Link(linkKey, exontrol.EXG2ANTTLib.LinkPropertyEnum.exLinkText, "new text")
    .EndUpdateLink(iChangeLink)
End With
```

These samples add new entries to undo/redo queue as : "UpdateLink;L1;12;;new text " which indicates , the link as being L1, the 12 indicates the exLinkText predefined value, and so on. Once the sample is called, the link's text is changed, and using the CTRL + Z, you can restore back the old value, or pressing the CTRL + Y you can change back after restoring.

method Items.EnsureVisibleBar (Item as HITEM, [Key as Variant])

Ensures that the given item-bar fits the chart's visible area.

Type	Description
Item as HITEM	A long expression that indicates the the handle of the item that hosts the bar. If the Item parameter is 0, it indicates all bars. In this case the DefaultItem property should be zero (by default), else it refers the item being indicated by DefaultItem (/COM version only) property.
Key as Variant	A String expression that indicates the key of the bar being ensured. If missing, the method ensures that the item fits the control's visible area. The Key may include a pattern with wild characters as *,?,# or [], if the Key starts with "<" and ends on ">" aka "<K*>" which indicates all bars with the key K or starts on K. The pattern may include a space which divides multiple patterns for matching. For instance "<A* *K>" indicates all keys that start on A and all keys that end on K. The method ends once an item-bar is found.

The EnsureVisibleBar method ensures that the given item-bar fits the chart's visible area. The EnsureVisibleBar method expands the parent items if the bar is hosted by a collapsed item. The EnsureVisibleItem method ensures that the item fits the control's visible area. Use the EnsureVisibleColumn method to ensure that a specified column fits the control's client area.

The Item and Key parameters can be one of the following:

- EnsureVisibleBar(**item**), ensures that giving **item**
- EnsureVisibleBar(**item**, **key**), ensures that the bar of specified **key** of giving **item**
- EnsureVisibleBar(**item**, **<pattern>**), ensures that the first-bar of giving **item** whose key matches the **pattern** (can use wild characters such as wild characters as *,?,# or [])
- EnsureVisibleBar(**0**, **key**), ensures that the first bar of specified **key** (looks through all the items until one is found)
- EnsureVisibleBar(**0**, **<pattern>**), ensures that the first-bar whose key matches the **pattern** (can use wild characters such as wild characters as *,?,# or [], looks through all the items until one is found)

fits the chart's visible area

method Items.EnsureVisibleItem (Item as HITEM)

Ensures the given item is in the visible client area.

Type	Description
Item as HITEM	A long expression that indicates the item's handle that fits the client area.

The EnsureVisibleItem method ensures that the item fits the control's visible area. The EnsureVisibleItem method scrolls the control's content until the item fits the control's visible area. The EnsureVisibleItem method expands the parent items in case it is collapased. Use the IsItemVisible to check if an item fits the control's client area. Use the Scroll method to scroll programmatically the control. Use the EnsureVisibleColumn method to ensure that a specified column fits the control's client area. The EnsureVisibleBar method ensures that the given item-bar fits the chart's visible area.

The following VB sample ensures that first item is visible:

```
G2antt1.Items.EnsureVisibleItem G2antt1.Items(0)
```

The following C++ sample ensures that first item is visible:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.EnsureVisibleItem( items.GetItemByIndex( 0 ) );
```

The following C# sample ensures that first item is visible:

```
axG2antt1.Items.EnsureVisibleItem(axG2antt1.Items[0]);
```

The following VB.NET sample ensures that first item is visible:

```
AxG2antt1.Items.EnsureVisibleItem( AxG2antt1.Items.FocusItem );
```

The following VFP sample ensures that first item is visible:

```
with thisform.G2antt1.Items
   .EnsureVisibleItem( .ItemByIndex( 0 ) )
endwith
```

property Items.ExpandItem(Item as HITEM) as Boolean

Expands, or collapses, the child items of the specified item.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item being expanded or collapsed. If the Item is 0, setting the ExpandItem property expands or collapses all items. For instance, the ExpandItem(0) = False, collapses all items, while the ExpandItem(0) = True, expands all items.
Boolean	A boolean expression that indicates whether the item is expanded or collapsed.

Use ExpandItem property to programmatically expand or collapse an item. Use the ExpandItem property to check whether an items is expanded or collapsed. Before expanding/collapsing an item, the control fires the BeforeExpandItem event. Use the BeforeExpandIvent to cancel expanding/collapsing of an item. After item was expanded/collapsed the control fires the AfterExpandItem event. The following samples shows how to expand the selected item: G2antt1.Items.ExpandItem(G2antt1.Items.SelectedItem()) = True**.** The property has no effect if the item has no child items. To check if the item has child items you can use ChildCount property. Use the ItemHasChildren property to display a +/- expand sign to the item even if it doesn't contain child items. The ExpandOnSearch property specifies whether the control expands nodes when incremental searching is on (AutoSearch property is different than 0) and user types characters when the control has the focus. Use the ExpandOnKeys property to specify whether the user expands or collapses the focused items using arrow keys. Use the InsertItem property to add child items.

The following VB sample programmatically expands the item when the user selects it :

```
Private Sub G2antt1_SelectionChanged()
  G2antt1.Items.ExpandItem(G2antt1.Items.SelectedItem()) = True
End Sub
```

The following VB sample expands programmatically the focused item:

```
With G2antt1.Items
  .ExpandItem(.FocusItem) = True
End With
```

The following C++ sample expands programmatically the focused item:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetExpandItem( items.GetFocusItem(), TRUE );
```

The following VB.NET sample expands programmatically the focused item:

```
AxG2antt1.Items.ExpandItem( AxG2antt1.Items.FocusItem ) = True
```

The following C# sample expands programmatically the focused item:

```
axG2antt1.Items.set_ExpandItem( axG2antt1.Items.FocusItem, true );
```

The following VFP sample expands programmatically the focused item:

```
with thisform.G2antt1.Items
   .DefaultItem = .FocusItem
   .ExpandItem( 0 ) = .t.
endwith
```

property Items.FindBar (BarKey as Variant, [StartIndex as Variant]) as HITEM

Finds the item that hosts the specified bar.

Type	Description
BarKey as Variant	A Variant expression that holds the key of the bar to look for.
StartIndex as Variant	A Long expression that could be one of the following: • 0 or positive, specifies the index to start searching from. If missing, 0 is used instead. The FindBar method searches all unlocked items (not including the locked items). • -1, searches for specified task into locked-top, unlocked and locked-bottom items (all-items), and returns the handle of the item that holds the specified bar. • -2, searches for specified task into locked-top, and locked-bottom items (locked-items), and returns the handle of the item that holds the specified bar. • -3, searches for specified task into locked-top items, and returns the handle of the item that holds the specified bar. • -4 searches for specified task into locked-bottom items, and returns the handle of the item that holds the specified bar.
HITEM	A Long expression that specifies the handle of the item that hosts the specified bar. If 0, the FindBar found no item that hosts a bar with specified key.

The FindBar property looks for the item that hosts the specified bar. The FindBar property returns 0 if no item has been found. The EnsureVisibleItem method ensures that an item fits the control's client area, and so the chart is vertically scrolled until the specified item fits the control's area. The ScrollTo method scrolls horizontally the chart until the specified date fits the chart's client area. The ItemBar property accesses the properties of the specified bar.

The following VB sample ensures that specified bar fits the control's chart area (for the /COM version):

```
Private Sub ensureVisibleBar(BarKey)
  With G2antt1
```

```
      .BeginUpdate
      With .Items
         Dim h As HITEM
         h = .FindBar(BarKey)
         If (h <> 0) Then
            .EnsureVisibleItem h
            G2antt1.Chart.ScrollTo .ItemBar(h, BarKey, exBarStart),
AlignmentEnum.CenterAlignment
         End If
      End With
      .EndUpdate
   End With
End Sub
```

The following VB/NET sample ensures that the specified bar fits the control's chart area (the code is for the /NET Assembly version):

```
Private Sub ensureVisibleBar(ByVal BarKey)
   With Exg2antt1
      With .Items
         Dim h As Integer = .get_FindBar(BarKey)
         If (h <> 0) Then
            Exg2antt1.BeginUpdate()
            .EnsureVisibleItem(h)
            Exg2antt1.Chart.ScrollTo(.get_BarStart(h, BarKey),
exontrol.EXG2ANTTLib.AlignmentEnum.CenterAlignment)
            Exg2antt1.EndUpdate()
         End If
      End With
   End With
End Sub
```

The following C# sample ensures that the specified bar fits the control's chart area (the code is for the /NET Assembly version):

```
private void ensureVisibleBar(object barKey)
{
   int h = exg2antt1.Items.get_FindBar(barKey);
```

```
    if (h != 0)
    {
        exg2antt1.BeginUpdate();
        exg2antt1.Items.EnsureVisibleItem(h);
        exg2antt1.Chart.ScrollTo(exg2antt1.Items.get_BarStart(h, barKey),
exontrol.EXG2ANTTLib.AlignmentEnum.CenterAlignment);
        exg2antt1.EndUpdate();
    }
}
```

property Items.FindItem (Value as Variant, [ColIndex as Variant], [StartIndex as Variant]) as HITEM

Finds an item, looking for Caption in ColIndex colum. The searching starts at StartIndex item.

Type	Description
Value as Variant	A Variant expression that indicates the caption that is searched for.
ColIndex as Variant	A string expression that indicates the column's caption, or a long expression that indicates the column's index.
StartIndex as Variant	A long value that indicates the index of item from where the searching starts.
HITEM	A long expression that indicates the item's handle that matches the criteria.

Use the FindItem to search for an item. Finds a control's item that matches CellValue(Item, ColIndex) = Caption. The StartIndex parameter indicates the index from where the searching starts. If it is missing, the searching starts from the item with the 0 index. The searching is case sensitive only if the ASCIIUpper property is empty. Use the AutoSearch property to enable incremental search feature within the column. The FindBar method looks for the item that hosts a specified bar.

The following VB sample selects the first item that matches "DUMON" on the first column:

```
G2antt1.Items.SelectItem(G2antt1.Items.FindItem("DUMON", 0)) = True
```

The following C++ sample finds and selects an item:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
COleVariant vtMissing;
long hFind = items.GetFindItem( COleVariant("King"), COleVariant("LastName"), vtMissing
);
if ( hFind != NULL )
   items.SetSelectItem( hFind, TRUE );
```

The following C# sample finds and selects an item:

```
axG2antt1.Items.set_SelectItem(axG2antt1.Items.get_FindItem("Child 2", 0, 0), true);
```

The following VB.NET sample finds and selects an item:

```
With AxG2antt1.Items
    Dim iFind As Integer
    iFind = .FindItem("Child 2", 0)
    If Not (iFind = 0) Then
        .SelectItem(iFind) = True
    End If
End With
```

The following VFP sample finds and selects an item:

```
with thisform.G2antt1.Items
    .DefaultItem = .FindItem("Child 2",0)
    if ( .DefaultItem <> 0 )
        .SelectItem( 0 ) = .t.
    endif
endwith
```

property Items.FindItemData (UserData as Variant, [StartIndex as Variant]) as HITEM

Finds the item giving its data.

Type	Description
UserData as Variant	A Variant expression that indicates the value being searched.
StartIndex as Variant	A long expression that indicates the index of the item where the searching starts.
HITEM	A long expression that indicates the handle of the item found.

Use the FindItemData property to search for an item giving its extra-data. Use the ItemData property to associate an extra data to an item. Use the FindItem property to locate an item given its caption. Use the FindPath property to search for an item given its path.

property Items.FindPath (Path as String) as HITEM

Finds an item given its path.

Type	Description
Path as String	A string expression that indicates the item's path.
HITEM	A long expression that indicates the item's handle that matches the criteria.

The FindPath property searches the item on the column SearchColumnIndex. Use the FullPath property in order to get the item's path. Use the FindItem to search for an item.

The following VB sample selects the item based on its path:

```
G2antt1.Items.SelectItem(G2antt1.Items.FindPath("Files and Folders\Hidden Files and Folders\Do not show hidden files and folder")) = True
```

The following C++ sample selects the item based on its path:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
COleVariant vtMissing;
long hFind = items.GetFindPath( "Files and Folders\\Hidden Files and Folders\\Do not show hidden files and folder" );
if ( hFind != NULL )
   items.SetSelectItem( hFind, TRUE );
```

The following VB.NET sample selects the item based on its path:

```
With AxG2antt1.Items
   Dim iFind As Integer
   iFind = .FindPath("Files and Folders\Hidden Files and Folders\Do not show hidden files and folder")
   If Not (iFind = 0) Then
      .SelectItem(iFind) = True
   End If
End With
```

The following C# sample selects the item based on its path:

```
int iFind = axG2antt1.Items.get_FindPath("Files and Folders\\Hidden Files and
Folders\\Do not show hidden files and folder");
if ( iFind != 0 )
   axG2antt1.Items.set_SelectItem(iFind, true);
```

The following VFP sample selects the item based on its path:

```
with thisform.G2antt1.Items
   .DefaultItem = .FindPath("Files and Folders\Hidden Files and Folders\Do not show
hidden files and folder")
   if ( .DefaultItem <> 0 )
      .SelectItem( 0 ) = .t.
   endif
endwith
```

property Items.FirstItemBar (Item as HITEM) as Variant

Gets the key of the first bar in the item.

Type	Description
Item as HITEM	A HITEM expression that indicates the handle of the item where the bars are enumerated.
Variant	A String expression that indicates the key of the first bar in the item, or empty if the item contains no bar.

Use the FirstItemBar and NextItemBar methods to enumerate the bars inside the item. Use the ItemBar property to access properties of the specified bar. Use the AddBar method to add new bars to the item. Use the AddLink method to link a bar with another. Use the AllowCreateBar method to create new bars using the mouse. Use the RemoveBar method to remove a bar from an item. Use the ClearBars method to remove all bars in the item.

The following VB.NET sample enumerates all items and bars in the control (/NET or /WPF version):

```
With Exg2antt1
   Dim i, h As Integer, key As Object
   For i = 0 To .Items.ItemCount - 1
      h = .Items(i)
      key = .Items.get_FirstItemBar(h)
      While TypeOf key Is String
         Debug.Print("Key = " & key & ", Item " & .Items.get_CellCaption(h, 0))
         key = CStr(.Items.get_NextItemBar(h, key))
      End While
   Next
End With
```

The following C# sample enumerates all items and bars in the control (/NET or /WPF version):

```
for (int i = 0; i < exg2antt1.Items.ItemCount; i++)
{
   int h = exg2antt1.Items[i];
   object key = exg2antt1.Items.get_FirstItemBar(h);
   while (key != null)
   {
```

```
      System.Diagnostics.Debug.Print("Key = " + key + ", Item " +
exg2antt1.Items.get_CellCaption(h, 0));
      key = exg2antt1.Items.get_NextItemBar(h, key);
   }
}
```

The following VB sample enumerates the bars in the item (h indicates the handle of the item):

```
With G2antt1
   If Not (h = 0) Then
      Dim k As Variant
      k = .Items.FirstItemBar(h)
      While Not IsEmpty(k)
         Debug.Print "Key = " & k
         k = .Items.NextItemBar(h, k)
      Wend
   End If
End With
```

The following C++ sample enumerates the bars in the item (h indicates the handle of the item):

```
CItems items = m_g2antt.GetItems();
COleVariant vtBar = items.GetFirstItemBar(h) ;
while ( V_VT( &vtBar ) != VT_EMPTY )
{
   OutputDebugString( V2S( &vtBar ) );
   OutputDebugString( "\n" );
   vtBar = items.GetNextItemBar( h, vtBar );
}
```

where the V2S function converts a Variant expression to a string:

```
static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
   if ( pv )
   {
      if ( pv->vt == VT_ERROR )
```

```
        return szDefault;

    COleVariant vt;
    vt.ChangeType( VT_BSTR, pv );
    return V_BSTR( &vt );
  }
  return szDefault;
}
```

The following VB.NET sample enumerates the bars in the item (h indicates the handle of the item):

```
With AxG2antt1
  If Not (h = 0) Then
    Dim k As Object
    k = .Items.FirstItemBar(h)
    While TypeOf k Is String
      System.Diagnostics.Debug.Print(k.ToString)
      k = .Items.NextItemBar(h, k)
    End While
  End If
End With
```

The following C# sample enumerates the bars in the item (h indicates the handle of the item):

```
object k = axG2antt1.Items.get_FirstItemBar(h);
while ( k != null )
{
  System.Diagnostics.Debug.Print(k.ToString());
  k = axG2antt1.Items.get_NextItemBar(h, k);
}
```

The following VFP sample enumerates the bars in the item (h indicates the handle of the item):

```
With thisform.G2antt1
  If Not (h = 0) Then
    local k
```

```
      k = .Items.FirstItemBar(h)
      do While !empty(k)
         ?k
         k = .Items.NextItemBar(h, k)
      enddo
   Endif
EndWith
```

In VFP, please make sure that you are using non empty values for the keys. For instance, if you are omitting the Key parameter of the AddBar method, an empty key is missing. If you need to use the FirstItemBar and NextItemBar properties, you have to use non empty keys for the bars.

property Items.FirstLink as Variant

Gets the key of the first link.

Type	Description
Variant	A string expression that indicates the key of the first link, or empty, if there are no links.

Use the FirstLink and NextLink properties to enumerate the links in the control. The FirstLink property retrieves an empty value, if there are no links in the control. Use the AddLink property to link two bars. Use the ShowLinks property to show or hide the links. Use the Link property to access a property of the link.

The following VB sample enumerates the links:

```
With G2antt1.Items
   Dim k As Variant
   k = .FirstLink()
   While Not IsEmpty(k)
      Debug.Print "LinkKey = " & k
      k = .NextLink(k)
   Wend
End With
```

The following C++ sample enumerates the links:

```
CItems items = m_g2antt.GetItems();
COleVariant vtLinkKey = items.GetFirstLink() ;
while ( V_VT( &vtLinkKey ) != VT_EMPTY )
{
   OutputDebugString( V2S( &vtLinkKey ) );
   OutputDebugString( "\n" );
   vtLinkKey = items.GetNextLink( vtLinkKey );
}
```

where the V2S function converts a Variant expression to a string:

```
static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
   if ( pv )
```

```
    {
        if ( pv->vt == VT_ERROR )
            return szDefault;

        COleVariant vt;
        vt.ChangeType( VT_BSTR, pv );
        return V_BSTR( &vt );
    }
    return szDefault;
}
```

The following VB.NET sample enumerates the links:

```
With AxG2antt1.Items
    Dim k As Object
    k = .FirstLink
    While (TypeOf k Is String)
        System.Diagnostics.Debug.Print(k.ToString)
        k = .NextLink(k)
    End While
End With
```

The following C# sample enumerates the links:

```
object k = axG2antt1.Items.FirstLink;
while (k != null)
{
    System.Diagnostics.Debug.Print(k.ToString());
    k = axG2antt1.Items.get_NextLink(k);
}
```

The following VFP sample enumerates the links:

```
With thisform.G2antt1.Items
    local k
    k = .FirstLink
    do While !empty(k)
        ?k
        k = .NextLink(k)
```

```
    enddo
endwith
```

property Items.FirstVisibleItem as HITEM

Retrieves the handle of the first visible item into control.

Type	Description
HITEM	A long expression that indicates the handle of the first visible item.

Use the FirstVisibleItem, NextVisibleItem and IsItemVisible properties to get the items that fit the client area. Use the NextVisibleItem property to get the next visible item. Use the IsVisibleItem property to check whether an item fits the control's client area.

The following VB sample enumerates the items that fit the control's client area:

```
On Error Resume Next
Dim h As HITEM
Dim i As Long, j As Long, nCols As Long
nCols = G2antt1.Columns.Count
With G2antt1.Items
  h = .FirstVisibleItem
  While Not (h = 0) And .IsItemVisible(h)
    Dim s As String
    s = ""
    For j = 0 To nCols - 1
      s = s + .CellValue(h, j) + Chr(9)
    Next
    Debug.Print s
    h = .NextVisibleItem(h)
  Wend
End With
```

The FormatColumn event is fired before displaying a cell, so you can handle the FormatColumn to display anything on the cell at runtime. This way you can display the row position, you can display the value using the currency format, and so on. The FireFormatColumn property allows the control to fire the FormatColumn event for the column. The Position property specifies the position of the column.

- If your chart does *not* display a tree or a hierarchy this property is ok to be used with FormatColumn event to display the position

The following VB sample handles the FormatColumn event to display the row position:

```
Private Sub G2antt1_FormatColumn(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal
ColIndex As Long, Value As Variant)
   Value = G2antt1.Items.ItemPosition(Item)
End Sub
```

- If your chart displays a tree or a hierarchy the position of the item must be determined relative to the FirstVisibleItem as shown in the following VB sample:

```
Private Sub G2antt1_FormatColumn(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal
ColIndex As Long, Value As Variant)
   Value = G2antt1.ScrollPos(True) + RelPos(Item)
End Sub

Private Function RelPos(ByVal hVisible As Long) As Long
   With G2antt1.Items
      Dim h As Long, i As Long, n As Long
      i = 0
      n = .VisibleCount + 1
      h = .FirstVisibleItem
      While (i <= n) And h <> 0 And h <> hVisible
         i = i + 1
         h = .NextVisibleItem(h)
      Wend
      RelPos = i
   End With
End Function
```

The following C++ sample enumerates the items that fit the control's client area:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
long hItem = items.GetFirstVisibleItem();
while ( hItem && items.GetIsItemVisible( hItem ) )
{
   OutputDebugString( V2S( &items.GetCellValue( COleVariant( hItem ), COleVariant(
long(0) ) ) ) );
   hItem = items.GetNextVisibleItem( hItem );
}
```

The following VB.NET sample enumerates the items that fit the control's client area:

```
With AxG2antt1.Items
   Dim hItem As Integer
   hItem = .FirstVisibleItem
   While Not (hItem = 0)
      If (.IsItemVisible(hItem)) Then
         Debug.Print(.CellValue(hItem, 0))
         hItem = .NextVisibleItem(hItem)
      Else
         Exit While
      End If
   End While
End With
```

The following C# sample enumerates the items that fit the control's client area:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
int hItem = items.FirstVisibleItem;
while ( ( hItem != 0 ) && (items.get_IsItemVisible(hItem)) )
{
   object strCaption = items.get_CellValue(hItem, 0);
   System.Diagnostics.Debug.WriteLine( strCaption != null ? strCaption.ToString() : "" );
   hItem = items.get_NextVisibleItem(hItem);
}
```

The following VFP sample enumerates the items that fit the control's client area:

```
with thisform.G2antt1.Items
   .DefaultItem = .FirstVisibleItem
   do while ( ( .DefaultItem <> 0 ) and ( .IsItemVisible( 0 ) ) )
      wait window .CellValue( 0, 0 )
      .DefaultItem = .NextVisibleItem( 0 )
   enddo
endwith
```

property Items.FocusItem as HITEM

Retrieves the handle of item that has the focus.

Type	Description
HITEM	A long expression that indicates the handle of the focused item.

The FocusItem property specifies the handle of the focused item. If there is no focused item the FocusItem property retrieves 0. At one moment, only one item can be focused. When the selection is changed the focused item is changed too. Use the SelectCount property to get the number of selected items. Use the SelectedItem property to get the selected item. Use the SelectItem to select or unselect a specified item. If the control supports only single selection, you can use the FocusItem property to get the selected/focused item because they are always the same. Use the ShowFocusRect property to indicate whether the control draws a marking rectangle around the focused item. You can change the focused item, by selecting a new item using the SelectItem method. If the items is not selectable, it is not focusable as well. Use the SelectableItem property to specify whether an item is selectable/focusable.

property Items.FormatCell([Item as Variant], [ColIndex as Variant]) as String

Specifies the custom format to display the cell's content.

Type	Description
Item as Variant	A long expression that indicates the handle of the item.
ColIndex as Variant	A long expression that indicates the column's index, a string expression that indicates the column's key or the column's caption.
String	A string expression that indicates the format to be applied on the cell's value, including HTML formatting, if the cell supports it.

By default, the FormatCell property is empty. The format is being applied if valid (not empty, and syntactically correct). The expression may be a combination of variables, constants, strings, dates and operators, and value. The *value* operator gives the value to be formatted. A string is delimited by ", ` or ' characters, and inside they can have the starting character preceded by \ character, ie "\"This is a quote\"". A date is delimited by # character, ie #1/31/2001 10:00# means the January 31th, 2001, 10:00 AM. The FormatColumn property applies the predefined format for all cells in the columns. The CellValue property indicates the cell's value.

The CellValue property of the cell is being shown as:

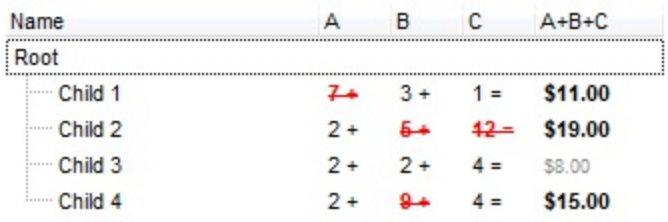
- formatted using the FormatCell property, if it is valid
- formatted using the FormatColumn property, if it is valid

In other words, all cells applies the format of the FormatColumn property, excepts the cells with the FormatCell property being set. If the cell belongs to a column with the FireFormatColumn property on True, the Value parameter of the FormatColumn event shows the newly caption for the cell to be shown.

For instance:

- the "*currency(value)*" displays the column using the current format for the currency ie, 1000 gets displayed as $1,000.00
- the "*longdate(date(value))*" converts the value to a date and gets the long format to display the date in the column, ie #1/1/2001# displays instead Monday, January 01, 2001
- the "*'' + ((0:=proper(value)) left 1) + '' + (=:0 mid 2)*" converts the name to proper, so the first letter is capitalized, bolds the first character, and let unchanged the rest, ie a "mihai filimon" gets displayed "**M**ihai Filimon".

- the "*len(value) ? ((0:=dbl(value)) < 10 ? '<fgcolor=808080><font ;7>' : '<b>') + currency(=:0)*" displays the cells that contains not empty daya, the value in currency format, with a different font and color for values less than 10, and bolded for those that are greater than 10, as can see in the following screen shot in the column (A+B+C):



The **value** keyword in the FormatColumn property indicates the value to be formatted.

The expression supports cell's identifiers as follows:

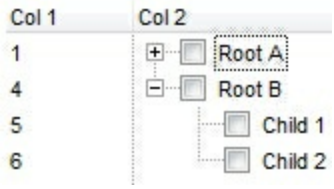
- ***%0**, **%1**, **%2**, ... specifies the value of the cell in the column with the index 0, 1 2, ... The CellValue property specifies the cell's value. For instance, "%0 format ``" formats the value on the cell with the index 0, using current regional setting, while "int(%1)" converts the value of the column with the index 1, to integer.*
- ***%C0**, **%C1**, **%C2**, ... specifies the caption of the cell, or the string the cell displays in the column with the index 0, 1 2, ... The CellCaption property specifies the cell's caption. The cell's value may be different than what the cell displays as a string. For instance, let's say a cell display HTML format. The %0 returns the html format including the HTML tags, while %C0 returns the cell's content as string without HTML tags. For instance, "upper(%C1)" converts the caption of the cell with the index 1, to upper case, while "%C0 left 2" returns the leftmost two characters on the cell with the index 0.*
- ***%CD0**, **%CD1**, **%CD2**, ... specifies the cell's extra data in the column with the index 0, 1 2, ... The CellData property associates any extra/user data to a cell. For instance, "%CD0 = `your user data`" specifies all cells whose CellData property is `your user data`, on the column with the index 0.*
- ***%CS0**, **%CS1**, **%CS2**, ... specifies the cell's state in the column with the index 0, 1 2, ... The CellState property specifies the cell's state, and so it indicates whether the cell is checked or un-checked. For instance, "%CS0" defines all checked items on the column with the index 0, or "not %CS1" defines all un-checked items in the column with the index 1.*

The expression supports auto-numbering predefined operators as follows:

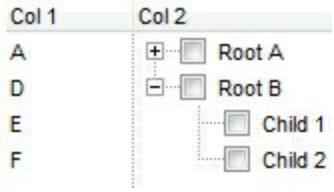
- number **index** 'format', indicates the index of the item. The first added item has the index 0, the second added item has the index 1, and so on. The index of the item remains the same even if the order of the items is changed by sorting. For instance, 1 index '' gets the index of the item starting from 1 while 100 index '' gets the index of the

item starting from 100. The number indicates the starting index, while the format is a set of characters to be used for specifying the index. If the format is missing, the index of the item is formatted as numbers. For instance: 1 index 'A-Z' gets the index as A, B, C... Z, BA, BB, ... BZ, CA, ... The 1 index 'abc' gives the index as: a,b,c,ba,bb,bc,ca,cb,cc,.... You can use other number formatting function to format the returned value. For instance "1 index " format '0||2|:'" gets the numbers grouped by 2 digits and separated by : character.

In the following screen shot the FormatColumn("Col 1") = "1 index ""

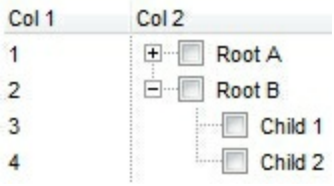


In the following screen shot the FormatColumn("Col 1") = "1 index 'A-Z'"

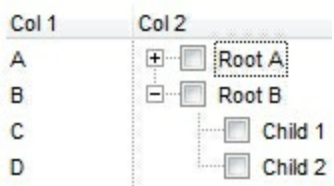


- number **apos** 'format' indicates the absolute position of the item. The first displayed item has the absolute position 0 (scrolling position on top), the next visible item is 1, and so on. The number indicates the starting position, while the format is a set of characters to be used for specifying the position. For instance, 1 apos " gets the absolute position of the item starting from 1, while 100 apos " gets the position of the item starting from 100. If the format is missing, the absolute position of the item is formatted as numbers.

  In the following screen shot the FormatColumn("Col 1") = "1 apos ""

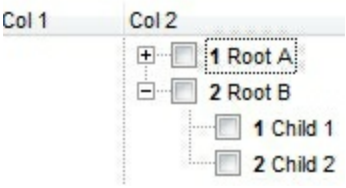


  In the following screen shot the FormatColumn("Col 1") = "1 apos 'A-Z'"

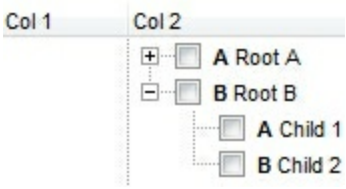


- number **pos** 'format' indicates the relative position of the item. The relative position is the position of the visible child item in the parent children collection. The number indicates the starting position, while the format is a set of characters to be used for specifying the position. For instance, 1 pos '' gets the relative position of the item starting from 1, while 100 pos '' gets the relative position of the item starting from 100. If the format is missing, the relative position of the item is formatted as numbers. *The difference between pos and opos can be seen while filtering the items in the control. For instance, if no filter is applied to the control, the pos and opos gets the same result. Instead, if the filter is applied, the opos gets the position of the item in the list of unfiltered items, while the pos gets the position of the item in the filtered list.*

  In the following screen shot the FormatColumn("Col 2") = "'' + 1 pos '' + ' ' + value"



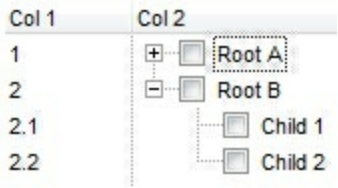
  In the following screen shot the FormatColumn("Col 2") = "'' + 1 pos 'A-Z' + ' ' + value"



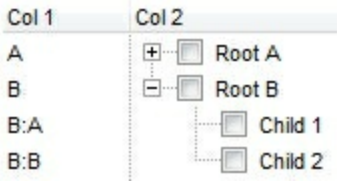
- number **opos** 'format' indicates the relative old position of the item. The relative old position is the position of the child item in the parent children collection. The number indicates the starting position, while the format is a set of characters to be used for specifying the position.For instance, 1 pos '' gets the relative position of the item starting from 1, while 100 pos '' gets the relative position of the item starting from 100. If the format is missing, the relative position of the item is formatted as numbers. *The difference between pos and opos can be seen while filtering the items in the control. For instance, if no filter is applied to the control, the pos and opos gets the same result. Instead, if the filter is applied, the opos gets the position of the item in the list of unfiltered items, while the pos gets the position of the item in the filtered list.*
- number **rpos** 'format' indicates the relative recursive position of the item. The recursive position indicates the position of the parent items too. The relative position is the position of the visible child item in the parent children collection. The number indicates the starting position, while the format is of the following type "delimiter|format|format|...". If the format is missing, the delimiter is . character, and the positions are formatted as numbers. The format is applied consecutively to each

parent item, from root to item itself.

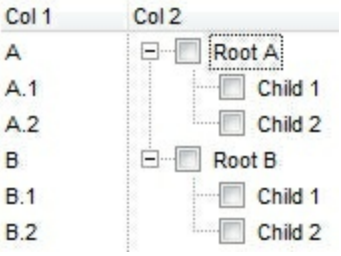
In the following screen shot the FormatColumn("Col 1") = "1 rpos ""



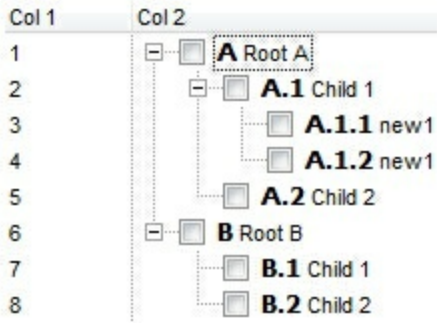
In the following screen shot the FormatColumn("Col 1") = "1 rpos ':|A-Z'"



In the following screen shot the FormatColumn("Col 1") = "1 rpos '.|A-Z|'"



In the following screen shot the FormatColumn("Col 1") = "1 apos "" and FormatColumn("Col 2") = "'' + 1 rpos '.|A-Z|' + ' ' + value"



- number **rindex** 'format', number **rapos** 'format' and number **ropos** 'format' are working similar with number **rpos** 'format', excepts that they gives the index, absolute position, or the old child position.

This property/method supports predefined constants and operators/functions as described here.

property Items.FullPath (Item as HITEM) as String

Returns the fully qualified path of the referenced item in the ExG2antt control.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item.
String	A string expression that indicates the fully qualified path.

Use the FullPath property in order to get the fully qualified path of the referenced item. Use PathSeparator to change the separator used by FullPath property. Use the FindPath property to get the item's selected based on its path. The fully qualified path is the concatenation of the text in the given cell's caption property on the column SearchColumnIndex with the CellValue property values of all its ancestors.

method Items.GroupBars (ItemA as HITEM, KeyA as Variant, StartA as Boolean, ItemB as HITEM, KeyB as Variant, StartB as Boolean, [GroupBarsOptions as Variant], [Options as Variant])

Groups two bars.

Type	Description
ItemA as HITEM	A long expression that indicates the handle of the item that contains the bar to group to.
KeyA as Variant	A long or string expression that specifies the key of the bar to group to. The Key parameter of the AddBar method specifies the key of the bar being added.
StartA as Boolean	A boolean expression that specifies whether the start or the end of the bar is grouped with other bar. True specifies that the start of the bar is grouped with other bar. False indicates that the end of the bar is grouped with other bar.
ItemB as HITEM	A long expression that indicates the handle of the item that contains the bar being grouped with.
KeyB as Variant	A long or string expression that specifies the key of the bar being grouped with.
StartB as Boolean	A boolean expression that specifies whether the start or the end of the bar is grouped with other bar. True specifies that the start of the bar is grouped with other bar. False indicates that the end of the bar is grouped with other bar.
GroupBarsOptions as Variant	(exGroupBarsOptionNone, by default or if missing) A GroupBarsOptionsEnum expression or a combination that specifies the way the bars gets grouped together. For instance, the exPreserveBarLength + exFlexibleInterval specifies that the bars preserves their lengths, and the bar B can be moved anywhere to the right of the bar A. *If the GroupBarsOptions is exGroupBarsNone the bars are ungrouped.*
Options as Variant	(empty, by default or if missing) A String expression that specifies a list of double values, separated by ; character, that specifies in this order: fixed interval ; maximum value when interval is increased, minimum value when the interval between bars is decreased. For instance, "2;4" value specifies that the interval between two bars should

	be 2 days and the interval can't be greater than 6 (2 + 4). Use the exLimitIntervalMin, exLimitIntervalMax, exLimitInterval or exFlexibleInterval for the GroupBarsOptions parameter when specifying the range of values that the interval between bars should be.

The GroupBars method groups two bars. In other words, you can associate a starting/ending point of one bar with any other starting/ending point of the bar. For instance, if you want to move both together you need to group the starting point of bar A with starting point of bar B, and ending point of the bar A with ending point of the bar B. Use the GroupBars method to handle or control the distance between 2 bars. For instance, if two bars or more bars are grouped, when a bar in the group is resize or moved, the other bars in the group are resized or moved accordingly. In the same manner, all other groups that are related with one of the group being resized or moved, are changed as well. You can group bars from different items. The GroupBars method may preserve the length of the bars, restrict the interval between bars, and so on when a change occurs in the group. Use the Link(exLinkGroupBars) to group two linked bars. For instance, the *.Link(LinkKey, exLinkGroupBars) = GroupBarsOptionsEnum.exPreserveBarLength + GroupBarsOptionsEnum.exFlexibleInterval + GroupBarsOptionsEnum.exIgnoreOriginalInterval* is equivalent with *.GroupBars .Link(LinkKey, exLinkStartItem), .Link(LinkKey, exLinkStartBar), False, .Link(LinkKey, exLinkEndItem), .Link(LinkKey, exLinkEndBar), True, GroupBarsOptionsEnum.exPreserveBarLength + GroupBarsOptionsEnum.exFlexibleInterval + GroupBarsOptionsEnum.exIgnoreOriginalInterval.*

The length of the bar and interval between two bars are defined as follows:

- The **length** of the bar is the same as its duration, in other words it is the difference between ending date of the bar and starting date of the bar.
- The **interval** between bars is the same as the distance between the starting and ending points of the grouping bars. For instance, if you have linked the end of the bar A with the start of the bar B, the interval is defined as difference between the starting date of the bar B and ending date of the bar A. In other sample, you may have linked the start of the bar A with start of the bar B, in this case the interval is defined as being the difference between the start of the bar A and starting date of the bar B, nothing else.

If using the NonworkingDays property, the ItemBar(exBarKeepWorkingCount) property indicates whether the working units of the bar is keep constant while moving/grouping. If the GroupBarsOptions parameter includes the exLimitIntervalTreatAsWorking the interval between bars is indicating the working days between days.

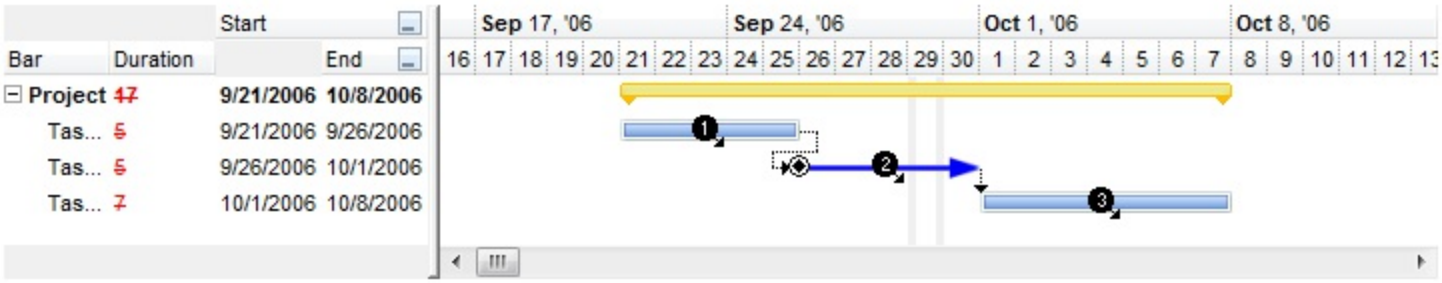
Samples:

- GroupBars(h1,"",False, h2, "", True, **7,"2"**) ' exPreserveBarLength + exIgnoreOriginalInterval, the distance between 2 bars is exactly 2 days.
- GroupBars(h1,"",False, h2, "", True, **95,"2"**) ' exPreserveBarLength + exIgnoreOriginalInterval + exLimitInterval + exLimitIntervalTreatAsWorking, the distance between 2 bars is exactly 2 working days.
- GroupBars(h1,"",False, h2, "", True, **15,"2"**) ' exPreserveBarLength + exIgnoreOriginalInterval + exLimitIntervalMin, the distance between 2 bars can be 2 days or more.
- GroupBars(h1,"",False, h2, "", True, **79,"2"**) ' exPreserveBarLength + exIgnoreOriginalInterval + exLimitIntervalMin + exLimitIntervalTreatAsWorking, the distance between 2 bars can be 2 working days or more.
- GroupBars(h1,"",False, h2, "", True, **31,"0;0;2"**) ' exPreserveBarLength + exIgnoreOriginalInterval + exLimitInterval, the distance between 2 bars can be 2 days or less.
- GroupBars(h1,"",False, h2, "", True, **95,"0;0;2"**) ' exPreserveBarLength + exIgnoreOriginalInterval + exLimitInterval + exLimitIntervalTreatAsWorking, the distance between 2 bars can be 2 working days or less.
- GroupBars(h1,"",False, h2, "", True, **31,"0;1;7"**) ' exPreserveBarLength + exIgnoreOriginalInterval + exLimitInterval, the distance between 2 bars can be between 1 and 7 days.
- GroupBars(h1,"",False, h2, "", True, **95,"0;1;5"**) ' exPreserveBarLength + exIgnoreOriginalInterval + exLimitInterval + exLimitIntervalTreatAsWorking, the distance between 2 bars can be between 1 and 7 working days.

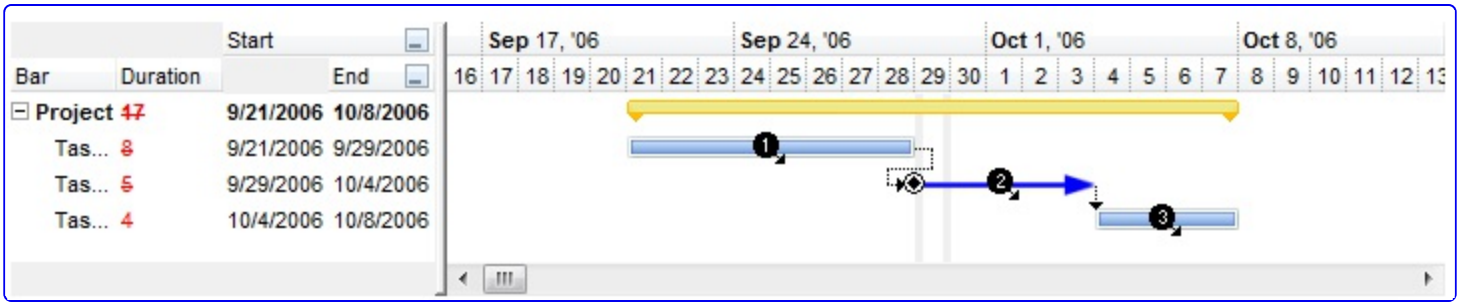
Use the AddLink method to add or draw a link between two bars. Use the DefineSummaryBars method to define bars in a summary bar, so it gets updated as soon as the child bars are moved or resized. Use the UngroupBars method to ungroup two bars or all bars. The Items.ItemBar(exBarsGroup) property retrieves a collection of item,key that defines the bars begin grouped with specified bar.

The following screen shots show the changes in a group of 3 bars (1, 2 and 3), when the bar 2 (the arrow) is moved from Sep 26 to Sep 29, using different options for GroupBarsOptions and Options parameters. Click on the picture and view the XML file that was used to generate the picture. The **Groups section** stores the groups of bars. You can use the LoadXML method to load the chart from these XML files. The XML file does not load EBN files!

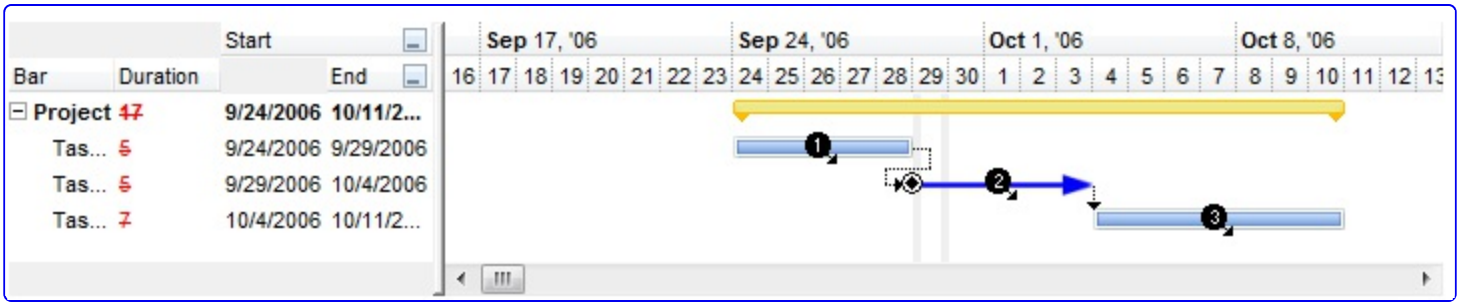
- *The following screen shot shows the chart before performing any change.*



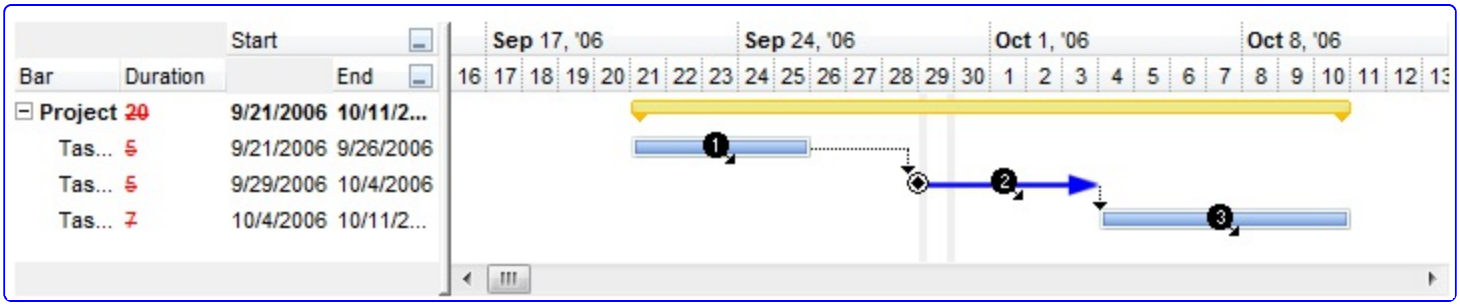
- *The following screen shot shows the chart after moving the bar 2, when the GroupBarsOptions and Options parameters are missing (by **default**). You can notice that the bar 1 and 3 are resized.*



- *The following screen shot shows the chart after moving the bar 2, when the GroupBarsOptions parameter is **exPreserveBarLength**. You can notice that all bars are moved to preserve their lengths.*

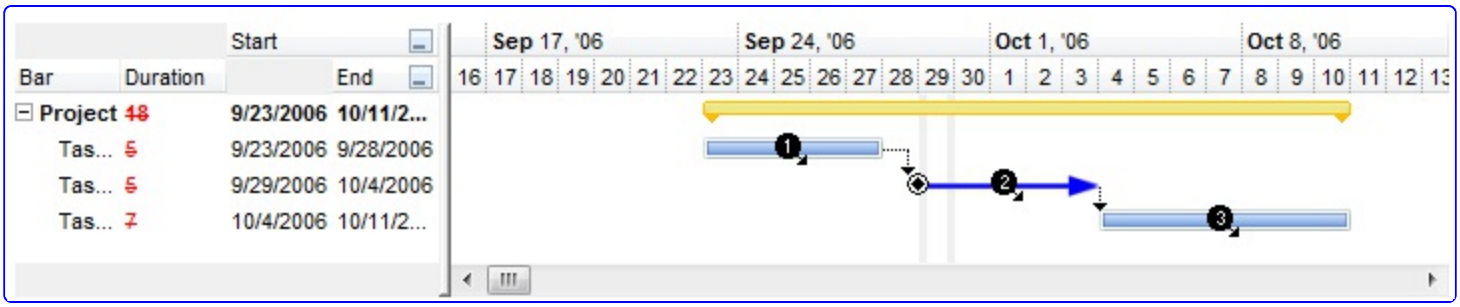


- *The following screen shot shows the chart after moving the bar 2, when the GroupBarsOptions parameter is **exPreserveBarLength + exFlexibleInterval**. All bars in the group preserves their lengths, and the bar 2 can be moved anywhere to the right of the bar 1*



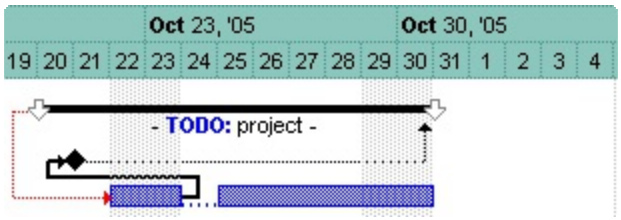
- The following screen shot shows the chart after moving the bar 2, when the GroupBarsOptions parameter is **exPreserveBarLength + exIgnoreOriginalInterval +**

exLimitInterval, and the Options parameter is **"0;1"**, which means the interval between two bars may be between 0 and 1, with a starting fixed interval being 0.

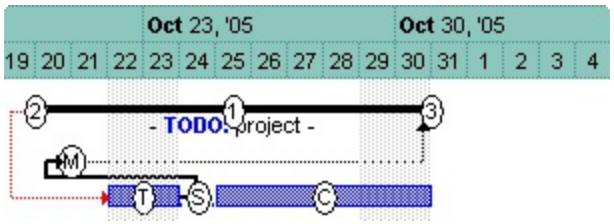


By default, when grouping, the distance between the margin of the bars being grouped is kept constant. For instance, if we group the end of the bar a with the start of the bar B, the distance between end of the bar a and the start of the bar B is the same when moving or resizing any of the bars A or B.

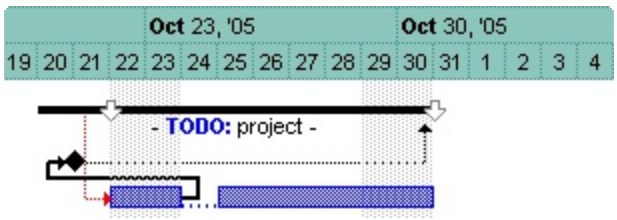
Let's stay that we have the following chart:



with the keys:



By default (no bars are grouped), if we move the **bar 2**, from Oct 19, to Oct 21 , we get the following (the bar 1, 3, and M are not moved or resized):

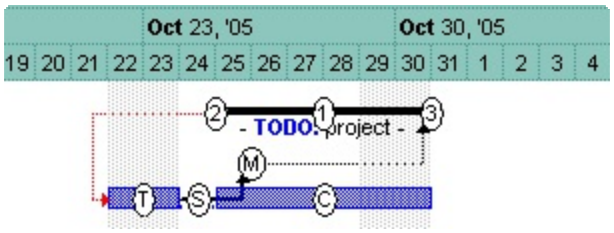


If we group the bars as follows:

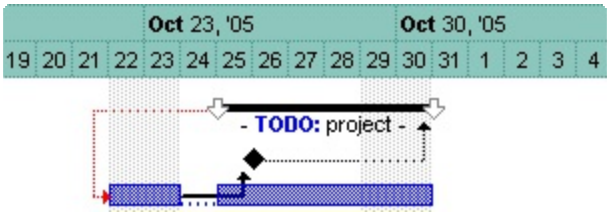
- the end of the bar 2 with the start of the bar 1
- the start of the bar 2 with the start of the bar 1

- the end of the bar 3 with the end of the bar 1
- the start of the bar 3 with the end of the bar 1
- the start of the bar M with the start of the bar 2
- the end of the bar M with the start of the bar 2

, and we move the **bar 2**, from Oct 19, to Oct 24 we get the following:



we notice that the bar 2 and M are moved, and bar 1 is resized.



property Items.GroupItem (Item as HITEM) as Long

Indicates a group item if positive, and the value specifies the index of the column that has been grouped.

Type	Description
Item as HITEM	A Long expression that specifies the handle of the item being queried
Long	A Long expression that specifies index of the column being grouped, or a negative value if the item is a regular item, not a grouping item.

The GroupItem method determines the index of the column that indicates the column being grouped. In other words, the CellCaption(Item,GroupItem(Item)) gets the default caption to be displayed for the grouping item. The Ungroup method removes all grouping items. For instance, when a column gets grouped by, the control sorts by that column, collects the unique values being found, and add a new item for each value found, by adding the items of the same value as children. The (AddGroupItem event is fired for each new item to be inserted in the Items collection during the grouping.

The following samples show how to display the grouping items with a solid background color, instead of a single line:

VBA

```
Private Sub G2antt1_AddGroupItem(ByVal Item As Long)
  With G2antt1
    With .Items
      .ItemDividerLine(Item) = 0
      .CellHAlignment(Item,.GroupItem(Item)) = 1
      .ItemBackColor(Item) = RGB(240,240,240)
    End With
  End With
End Sub
```

VB

```
Private Sub G2antt1_AddGroupItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
  With G2antt1
    With .Items
      .ItemDividerLine(Item) = EmptyLine
```

```
        .CellHAlignment(Item,.GroupItem(Item)) = CenterAlignment
        .ItemBackColor(Item) = RGB(240,240,240)
      End With
    End With
End Sub
```

VB.NET

```
Private Sub Exg2antt1_AddGroupItem(ByVal sender As System.Object,ByVal Item As
Integer) Handles Exg2antt1.AddGroupItem
    With Exg2antt1
      With .Items
        .set_ItemDividerLine(Item,exontrol.EXG2ANTTLib.DividerLineEnum.EmptyLine)

.set_CellHAlignment(Item,.get_GroupItem(Item),exontrol.EXG2ANTTLib.AlignmentEnum.Cer

        .set_ItemBackColor(Item,Color.FromArgb(240,240,240))
      End With
    End With
End Sub
```

C++

```
void OnAddGroupItemG2antt1(long Item)
{
    EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
    EXG2ANTTLib::IItemsPtr var_Items = spG2antt1->GetItems();
      var_Items->PutItemDividerLine(Item,EXG2ANTTLib::EmptyLine);
      var_Items->PutCellHAlignment(Item,var_Items-
>GetGroupItem(Item),EXG2ANTTLib::CenterAlignment);
      var_Items->PutItemBackColor(Item,RGB(240,240,240));
}
```

C++ Builder

```
void __fastcall TForm1::G2antt1AddGroupItem(TObject *Sender,Exg2anttlib_tlb::HITEM
Item)
```

```
{
   Exg2anttlib_tlb::IItemsPtr var_Items = G2antt1->Items;
      var_Items->set_ItemDividerLine(Item,Exg2anttlib_tlb::DividerLineEnum::EmptyLine);
      var_Items->set_CellHAlignment(TVariant(Item),TVariant(var_Items-
>get_GroupItem(Item)),Exg2anttlib_tlb::AlignmentEnum::CenterAlignment);
      var_Items->set_ItemBackColor(Item,RGB(240,240,240));
}
```

C#

```
private void exg2antt1_AddGroupItem(object sender,int Item)
{
   exontrol.EXG2ANTTLib.Items var_Items = exg2antt1.Items;

var_Items.set_ItemDividerLine(Item,exontrol.EXG2ANTTLib.DividerLineEnum.EmptyLine);

var_Items.set_CellHAlignment(Item,var_Items.get_GroupItem(Item),exontrol.EXG2ANTTLib.A

      var_Items.set_ItemBackColor(Item,Color.FromArgb(240,240,240));
}
```

JavaScript

```
<SCRIPT FOR="G2antt1" EVENT="AddGroupItem(Item)" LANGUAGE="JScript">
   var var_Items = G2antt1.Items;
      var_Items.ItemDividerLine(Item) = 0;
      var_Items.CellHAlignment(Item,var_Items.GroupItem(Item)) = 1;
      var_Items.ItemBackColor(Item) = 15790320;
</SCRIPT>
```

X++ (Dynamics Ax 2009)

```
void onEvent_AddGroupItem(int _Item)
{
   COM com_Items;
   anytype var_Items;
   ;
   var_Items = exg2antt1.Items(); com_Items = var_Items;
```

```
	com_Items.ItemDividerLine(_Item,0/*EmptyLine*/);

com_Items.CellHAlignment(_Item,com_Items.GroupItem(_Item),1/*CenterAlignment*/);
	com_Items.ItemBackColor(_Item,WinApi::RGB2int(240,240,240));
}
```

VFP

```
*** AddGroupItem event - Occurs after a new Group Item has been inserted to Items collection. ***
LPARAMETERS Item
   with thisform.G2antt1
      with .Items
         .ItemDividerLine(Item) = 0
         .CellHAlignment(Item,.GroupItem(Item)) = 1
         .ItemBackColor(Item) = RGB(240,240,240)
      endwith
   endwith

with thisform.G2antt1
   .BeginUpdate
   .HasLines = 0
   .ColumnAutoResize = .F.
   rs = CreateObject("ADOR.Recordset")
   with rs
      var_s = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Program Files\Exontrol\ExG2antt\Sample\SAMPLE.MDB"
      .Open("Orders",var_s,3,3)
   endwith
   .DataSource = rs
   .SingleSort = .F.
   .SortBarVisible = .T.
   .AllowGroupBy = .T.
   .Columns.Item(1).SortOrder = .T. && .T.
   .EndUpdate
endwith
```

Delphi (standard)

```
procedure TForm1.G2antt1AddGroupItem(ASender: TObject; Item : HITEM);
begin
   with G2antt1 do
   begin
      with Items do
      begin
         ItemDividerLine[Item] := EXG2ANTTLib_TLB.EmptyLine;
         CellHAlignment[OleVariant(Item),OleVariant(GroupItem[Item])] :=
EXG2ANTTLib_TLB.CenterAlignment;
         ItemBackColor[Item] := $f0f0f0;
      end;
   end
end;
```

Visual Objects

```
METHOD OCX_Exontrol1AddGroupItem(Item) CLASS MainDialog
   // AddGroupItem event - Occurs after a new Group Item has been inserted to
Items collection.
   local var_Items as IItems
   var_Items := oDCOCX_Exontrol1:Items
      var_Items:[ItemDividerLine,Item] := EmptyLine
      var_Items:[CellHAlignment,Item,var_Items:[GroupItem,Item]] := CenterAlignment
      var_Items:[ItemBackColor,Item] := RGB(240,240,240)
RETURN NIL
```

property Items.HasCellEditor ([Item as Variant], [ColIndex as Variant]) as Boolean

Specifies whether a cell has a built-in editor.

Type	Description
Item as Variant	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the cell's handle or the column's index, a string expression that indicates the column's caption or the column's key.
Boolean	A boolean expression that indicates whether the cell has a built-in editor created using the CellEditor method.

Use the HasCellEditor property to check whether the cell has an individual editor being added using the CellEditor method before. Use the HasCellEditor property to find if a cell has a particular editor. The HasCellEditor property gets true only if the cell has its own editor assigned, it never gets true, if the cell's column has an editor. Use the CellEditor method to assign different editors in the same column. Use the Editor property to assign the same editor for all cells in the column.

The following VB sample shows the drop down portion of the control when a cell is focused:

```
Private Sub G2antt1_FocusChanged()
  With G2antt1
    Dim i As Long
    i = .FocusColumnIndex
    With G2antt1.Items
      If (.CellEditorVisible(.FocusItem, i)) Then
        Dim e As EXG2ANTTLibCtl.Editor
        Set e = G2antt1.Columns(i).Editor
        If .HasCellEditor(.FocusItem, i) Then
          Set e = .CellEditor(.FocusItem, i)
        End If
        If Not e Is Nothing Then
          e.DropDown
        End If
      End If
    End With
  End With
End Sub
```

The following VB sample assigns a date type editor to the focused cell (the sample checks first if the cell doesn't have already an editor):

```
With G2antt1.Items
   Dim h As EXG2ANTTLibCtl.HITEM
   h = .FocusItem
   If Not .HasCellEditor(h, G2antt1.FocusColumnIndex) Then
      With .CellEditor(h, G2antt1.FocusColumnIndex)
         .EditType = DateType
      End With
   End If
End With
```

The following C++ sample assigns a date type editor to the focused cell (the sample checks first if the cell doesn't have already an editor):

```
#include "Items.h"
#include "Editor.h"
CItems items = m_g2antt.GetItems();
COleVariant vtItem( items.GetFocusItem() ), vtColumn(
long(m_g2antt.GetFocusColumnIndex() ) );
if ( !items.GetHasCellEditor( vtItem, vtColumn ) )
{
   CEditor editor = items.GetCellEditor( vtItem, vtColumn );
   editor.SetEditType( 7 /*DateType*/ );
}
```

The following VB.NET sample assigns a date type editor to the focused cell (the sample checks first if the cell doesn't have already an editor):

```
With AxG2antt1.Items
   Dim hItem As Integer = .FocusItem
   If Not .HasCellEditor(hItem, AxG2antt1.FocusColumnIndex) Then
      With .CellEditor(hItem, AxG2antt1.FocusColumnIndex)
         .EditType = EXG2ANTTLib.EditTypeEnum.DateType
      End With
   End If
End With
```

The following C# sample assigns a date type editor to the focused cell (the sample checks first if the cell doesn't have already an editor):

```
EXG2ANTTLib.Items items = axG2antt1.Items;
int hItem = items.FocusItem;
if (hItem != null)
   if (!items.get_HasCellEditor(hItem, axG2antt1.FocusColumnIndex))
   {
      EXG2ANTTLib.Editor editor = items.get_CellEditor(hItem,
axG2antt1.FocusColumnIndex);
      editor.EditType = EXG2ANTTLib.EditTypeEnum.DateType;
   }
```

The following VFP sample assigns a date type editor to the focused cell (the sample checks first if the cell doesn't have already an editor):

```
with thisform.G2antt1.Items
   .DefaultItem = .FocusItem
   if ( !.HasCellEditor(0, thisform.G2antt1.FocusColumnIndex ) )
      with .CellEditor( 0, thisform.G2antt1.FocusColumnIndex )
         .EditType = 7 && DateType
      endwith
   endif
endwith
```

property Items.InnerCell ([Item as Variant], [ColIndex as Variant], [Index as Variant]) as Variant

Retrieves the inner cell.

Type	Description
Item as Variant	A long expression that indicates the handle of the item where the cell is, or 0. If the Item parameter is 0, the ColIndex parameter must indicate the handle of the cell.
ColIndex as Variant	A long expression that indicates the index of the column where a cell is divided, or a long expression that indicates the handle of the cell being divided, if the Item parameter is missing or it is zero.
Index as Variant	A long expression that indicates the index of the inner being requested. If the Index parameter is missing or it is zero, the InnerCell property retrieves the master cell.
Variant	A long expression that indicates the handle of the inner cell.

Use the InnerCell property to get the inner cell. The InnerCell(, , 0) property always retrieves the same cell. The InnerCell(, , 1) retrieves the first inner cell, and so on. The InnerCells property always retrieves a non empty value. For instance, if a cell contains only two splited cells, the InnerCell(, , 3), or InnerCell(, , 4), and so on, always retrieves the last inner cell. The SplitCell method splits a cell in two cells (the newly created cell is called inner cell). Use the CellParent property to get the parent of the inner cell. Use the CellItem property to get the item that's the owner of the cell. Use the CellWidth property to specify the width of the inner cell. Use the CellParent property to determine whether the cell is a master cell or an inner cell. If the CellParent property gets 0, it means that the cell is master, else it is inner.

The following VB sample specifies whether a cell contains inner cells (the function checks whether a cell is splitted):

```
Private Function isSplit(ByVal g As EXG2ANTTLibCtl.G2antt, ByVal h As
EXG2ANTTLibCtl.HITEM, ByVal c As Long) As Boolean
  With g.Items
    isSplit = Iif(Not .InnerCell(h, c, 0) = .InnerCell(h, c, 1), True, False)
  End With
End Function
```

The following VB sample gets the master cell:

```
Private Function getMaster(ByVal g As EXG2ANTTLibCtl.G2antt, ByVal h As
EXG2ANTTLibCtl.HITEM, ByVal c As Long) As EXG2ANTTLibCtl.HCELL
   With g.Items
      Dim r As EXG2ANTTLibCtl.HCELL
      r = c
      If Not (h = 0) Then
         r = .ItemCell(h, c)
      End If
      While Not (.CellParent(, r) = 0)
         r = .CellParent(, r)
      Wend
      getMaster = r
   End With
End Function
```

The following VB sample counts the inner cells:

```
Private Function getInnerCount(ByVal g As EXG2ANTTLibCtl.G2antt, ByVal h As
EXG2ANTTLibCtl.HITEM, ByVal c As Long) As Long
   With g.Items
      Dim i As Long
      i = -1
      Do
         i = i + 1
      Loop While Not (.InnerCell(h, c, i) = .InnerCell(h, c, i + 1))
      getInnerCount = i
   End With
End Function
```

The following C++ sample specifies whether a cell contains inner cells (the function checks whether a cell is splitted):

```
long V2I( VARIANT* pvtValue )
{
   COleVariant vtResult;
   vtResult.ChangeType( VT_I4, pvtValue );
   return V_I4( &vtResult );
}
```

```
BOOL isSplit( CG2antt& g2antt, long h, long c )
{
   CItems items = g2antt.GetItems();
   return V2I( &items.GetInnerCell( COleVariant( h ), COleVariant( c ), COleVariant( (long)0 )
) ) != V2I( &items.GetInnerCell( COleVariant( h ), COleVariant( c ), COleVariant( (long)1 ) ) );
}
```

The following C++ sample gets the master cell:

```
long getMaster( CG2antt& g2antt, long h, long c )
{
   COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
   CItems items = g2antt.GetItems();
   long r = c;
   if ( h != 0 )
      r = items.GetItemCell( h, COleVariant( c ) );
   while ( V2I( &items.GetCellParent( vtMissing, COleVariant( r ) ) ) != 0 )
      r = V2I( &items.GetCellParent( vtMissing, COleVariant( r ) ) );
   return r;
}
```

The following C++ sample counts the inner cells:

```
long getInnerCount( CG2antt& g2antt, long h, long c )
{
   CItems items = g2antt.GetItems();
   COleVariant vtItem( h ), vtColumn( c );
   long i = -1;
   do
   {
      i++;
   }
   while ( V2I( &items.GetInnerCell( vtItem, vtColumn, COleVariant( i )  ) ) != V2I(
&items.GetInnerCell( vtItem, vtColumn, COleVariant( (long)(i + 1) )  ) ) );
   return i;
}
```

The following VB.NET sample splits the first visible cell in two cells:

```
With AxG2antt1.Items
    Dim i As Object
    i = .SplitCell(.FirstVisibleItem, 0)
    .CellValue(Nothing, i) = "inner cell"
End With
```

The following C# sample splits the first visible cell in two cells:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
object i = items.get_SplitCell(items.FirstVisibleItem, 0);
items.set_CellValue(null, i, "inner cell");
```

The following VFP sample splits the first visible cell in two cells:

```
with thisform.G2antt1.Items
    local i
    i = .SplitCell(.FirstVisibleItem,0)
    local s, crlf
    crlf = chr(13) + chr(10)
    s = "Items" + crlf
    s = s + "{" + crlf
    s = s + "CellValue(," + str(i) + ") = " + chr(34) + "inner cell" + chr(34) + crlf
    s = s + "}"
    thisform.G2antt1.Template = s
endwith
```

method Items.InsertControlItem (Parent as HITEM, ControlID as String, [License as Variant])

Inserts a new item of ActiveX type, and returns a handle to the newly created item.

Type	Description
Parent as HITEM	A long expression that indicates the handle of the parent item where the ActiveX will be inserted. If the argument is missing then the InsertControlItem property inserts the ActiveX control as a root item. If the Parent property is referring a locked item (ItemLocked property), the InsertControlItem property doesn't insert a new child ActiveX, instead insert the ActiveX control to the locked item that's specified by the Parent property.
ControlID as String	A string expression that can be formatted as follows: a prog ID, a CLSID, a URL, a reference to an Active document , a fragment of HTML.
License as Variant	A string expression that indicates the runtime license key, if it is required. An empty string, if the control doesn't require a runtime license key.

Return	Description
HITEM	A long expression that indicates the handle of the newly created item.

The InsertControlItem property creates the specified ActiveX control and hosts to a new child item of the control, while the InsertObjectItem property hosts the already created object to a new child item of the control. An inner control sends notifications/events to parent control through the ItemOleEvent event. Use the AddBar method to add bars to the item. The bars are always shown in the chart area. Use the PaneWidth property to specify the width of the chart.

The ControlID must be formatted in one of the following ways:

- A ProgID such as "Exontrol.G2antt"
- A CLSID such as "{8E27C92B-1264-101C-8A2F-040224009C02}"
- A URL such as "https://www.exontrol.com"
- A reference to an Active document such as "c:\temp\myfile.doc", or "c:\temp\picture.gif"
- A fragment of HTML such as "MSHTML:<HTML><BODY>This is a line of text</BODY></HTML>"
- A fragment of XML

In case the control you want to insert fails, you can add the "A2X:" prefix to the ControlID such as:

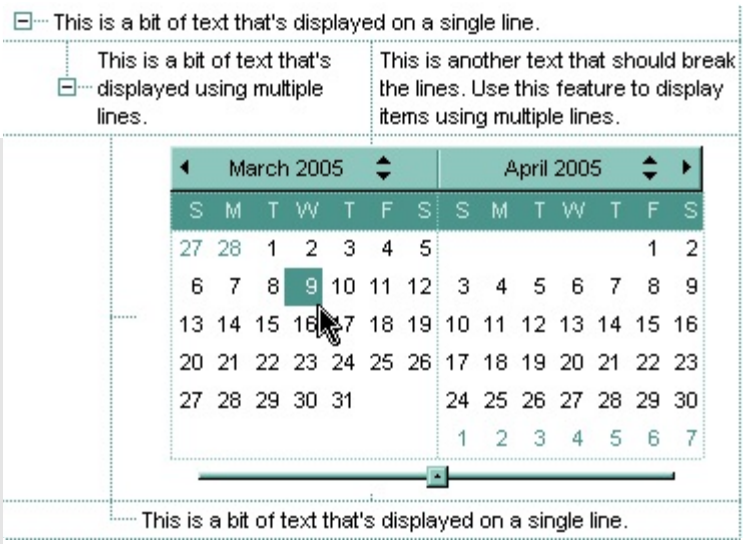
- A ProgID such as "A2X:Exontrol.Grid"
- A CLSID such as "A2X:{8E27C92B-1264-101C-8A2F-040224009C02}"
- A URL such as "A2X:https://www.exontrol.com"
- A reference to an Active document such as "A2X:c:\temp\myfile.doc", or "c:\temp\picture.gif"
- A fragment of HTML such as "A2X:MSHTML:<HTML><BODY>This is a line of text</BODY></HTML>"

The InsertControlItem property creates an ActiveX control that's hosted by the exGrid control. **The look and feel of the inner ActiveX control depends on the identifier you are using, and the version of the library that implements the ActiveX control, so you need to consult the documentation of the inner ActiveX control you are inserting inside the exG2antt control.**

Use the ItemHeight property to specify the height of the item when it contains an ActiveX control. Use the ItemWidth property to specify the width of the ActiveX control, or the position in the item where the ActiveX is displayed. Once that an item of ActiveX type has been added you can get the OLE control created using the ItemObject property. To check if an item contains an ActiveX control you can use ItemControlID property. To change the height of an ActiveX item you have to use ItemHeight property. When the control contains at least an item of ActiveX type, it is recommended to set ScrollBySingleLine property of control to true. Events from contained components are fired through to your program using the exact same model used in VB6 for components added at run time (See ItemOleEvent event, OleEvent and OleEventParam). For instance, when an ActiveX control fires an event, the control forwards that event to your container using ItemOleEvent event of the exG2antt control. Use the BeginUpdate and EndUpdate methods to update the control's content when adding ActiveX controls on the fly. Use the ItemControlID property to retrieve the control's identifier.

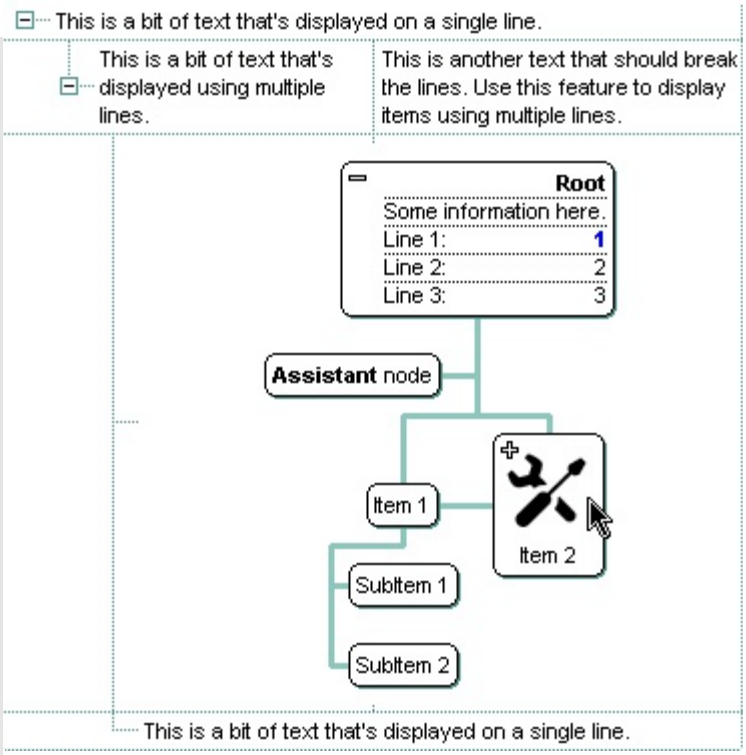
The following VB sample adds the Exontrol's ExCalendar Component:

```
With G2antt1
   .BeginUpdate
   .ScrollBySingleLine = True
   With G2antt1.Items
      Dim h As HITEM
      h = .InsertControlItem(,
"Exontrol.Calendar")
```



```
      .ItemHeight(h) = 182
      With .ItemObject(h)
         .Appearance = 0
         .BackColor = vbWhite
         .ForeColor = vbBlack
         .ShowTodayButton = False
      End With
   End With
   .EndUpdate
End With
```

The following C++ sample adds the Exontrol's ExOrgChart Component:




```
#include "Items.h"

#pragma warning( disable : 4146 )
#import <ExOrgChart.dll>

CItems items = m_g2antt.GetItems();
m_g2antt.BeginUpdate();
m_g2antt.SetScrollBySingleLine( TRUE );
COleVariant vtMissing; V_VT( &vtMissing ) =
VT_ERROR;
long h = items.InsertControlItem( 0,
"Exontrol.ChartView", vtMissing );
items.SetItemHeight( h, 182 );
EXORGCHARTLib::IChartViewPtr spChart(
items.GetItemObject(h) );
if ( spChart != NULL )
{
   spChart->BeginUpdate();
   spChart->BackColor = RGB(255,255,255);
   spChart->ForeColor = RGB(0,0,0);
   EXORGCHARTLib::INodesPtr spNodes =
spChart->Nodes;
   spNodes->Add( "Child 1", "Root", "1",
vtMissing, vtMissing );
```

```
   spNodes->Add( "SubChild 1", "1", vtMissing,
vtMissing, vtMissing );
   spNodes->Add( "SubChild 2", "1", vtMissing,
vtMissing, vtMissing );
   spNodes->Add( "Child 2", "Root", vtMissing,
vtMissing, vtMissing );
   spChart->EndUpdate();
}
m_g2antt.EndUpdate();
```

The sample uses the #import statement to include the ExOrgChart's Type Library. In this sample, the ItemObject property retrieves an IChartView object. The path to the library should be provided in case it is not located in your system folder.

The following C# sample adds the Exontrol's ExG2antt Component:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Items items = axG2antt1.Items;
axG2antt1.ScrollBySingleLine = true;
int h = items.InsertControlItem(0, "Exontrol.G2antt","");
items.set_ItemHeight(h, 182);
object g2anttInside = items.get_ItemObject(h);
if ( g2anttInside != null )
{
   EXG2ANTTLib.G2antt g2antt = g2anttInside as EXG2ANTTLib.G2antt;
   if (g2antt != null)
   {
      g2antt.BeginUpdate();
      g2antt.LinesAtRoot = EXG2ANTTLib.LinesAtRootEnum.exLinesAtRoot;
      g2antt.Columns.Add("Column 1");
      g2antt.Columns.Add("Column 2");
      g2antt.Columns.Add("Column 3");
      EXG2ANTTLib.Items itemsInside = g2antt.Items;
      int hInside = itemsInside.AddItem("Item 1");
      itemsInside.set_CellValue(hInside, 1, "SubItem 1");
      itemsInside.set_CellValue(hInside, 2, "SubItem 2");
      hInside = itemsInside.InsertItem(hInside, null, "Item 2");
      itemsInside.set_CellValue(hInside, 1, "SubItem 1");
```

```
      itemsInside.set_CellValue(hInside, 2, "SubItem 2");
      g2antt.EndUpdate();
   }
}
axG2antt1.EndUpdate();
```

The following VB.NET sample adds the Exontrol's ExOrgChart Component:

```
With AxG2antt1
   .BeginUpdate()
   .ScrollBySingleLine = True
   With .Items
      Dim hItem As Integer
      hItem = .InsertControlItem(, "Exontrol.ChartView")
      .ItemHeight(hItem) = 182
      With .ItemObject(hItem)
         .BackColor = ToUInt32(Color.White)
         .ForeColor = ToUInt32(Color.Black)
         With .Nodes
            .Add("Child 1", , "1")
            .Add("SubChild 1", "1")
            .Add("SubChild 2", "1")
            .Add("Child 2")
         End With
      End With
   End With
   .EndUpdate()
End With
```

The following VFP sample adds the Exontrol's ExGrid Component:

```
with thisform.G2antt1
   .BeginUpdate()
   .ScrollBySingleLine = .t.
   with .Items
      .DefaultItem = .InsertControlItem(0, "Exontrol.Grid")
      .ItemHeight( 0 ) = 182
      with .ItemObject( 0 )
```

```
        .BeginUpdate()
        with .Columns
           with .Add("Column 1").Editor()
              .EditType = 1 && EditType editor
           endwith
        endwith
        with .Items
           .AddItem("Text 1")
           .AddItem("Text 2")
           .AddItem("Text 3")
        endwith
        .EndUpdate()
     endwith
  endwith
  .EndUpdate()
endwith
```

The following VB sample adds dynamically an ExG2antt ActiveX Control and a Microsoft Calendar Control:

```
' Inserts a new ActiveX control of Exontrol.G2antt type
Dim hG2antt As HITEM
hG2antt = G2antt1.Items.InsertControlItem(G2antt1.Items(0), "Exontrol.G2antt",
runtimelicensekey )
' Sets the ActiveX control height
G2antt1.Items.ItemHeight(hG2antt) = 212
' Gets the ExG2antt control created. Since the ProgID used to create the item is
"Exontrol.G2antt"
' the object will be of EXG2ANTTLibCtl.G2antt type
Dim objG2antt As Object
Set objG2antt = G2antt1.Items.ItemObject(hG2antt)
objG2antt.Columns.Add "Column"
objG2antt.Items.AddItem "One"
objG2antt.Items.AddItem "Two"
objG2antt.Items.AddItem "Three"

' Inserts a new ActiveX control of MSCAL.Calendar type
Dim hCalc As HITEM
```

```
hCalc = objG2antt.Items.InsertControlItem(, "MSCal.Calendar")
Set objCalc = G2antt1.Items.ItemObject(hCalc)
objCalc.ShowTitle = False
objCalc.ShowDateSelectors = False
```

where the runtimelicensekey is the exG2antt's runtime license key. Please contact us to get the exG2antt's runtime license key. Please notice that your development license key **is not equivalent** with the generated runtime license key. **Your order number is required**, when requesting the control's runtime license key. If you are using the DEMO version for testing purpose, you don't need a runtime license key.

The following VB sample handles any event that a contained ActiveX fires:

```
Private Sub G2antt1_ItemOleEvent(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal Ev As
EXG2ANTTLibCtl.IOleEvent)
    On Error Resume Next
    Dim i As Long
    Debug.Print "The " & Ev.Name & " was fired. "
    If Not (Ev.CountParam = 0) Then
        Debug.Print "The event has the following parameters: "
        For i = 0 To Ev.CountParam - 1
            Debug.Print " - " & Ev(i).Name & " = " & Ev(i).Value
        Next
    End If
End Sub
```

Some of ActiveX controls requires additional window styles to be added to the conatiner window. For instance, the Web Brower added by the G2antt1.Items.InsertControlItem(, "https://www.exontrol.com") won't add scroll bars, so you have to do the following:

First thing is to declare the WS_HSCROLL and WS_VSCROLL constants at the top of your module:

```
Private Const WS_VSCROLL = &H200000
Private Const WS_HSCROLL = &H100000
```

Then you need to to insert a Web control use the following lines:

```
Dim hWeb As HITEM
hWeb = G2antt1.Items.InsertControlItem(, "https://www.exontrol.com")
```

```
G2antt1.Items.ItemHeight(hWeb) = 196
```

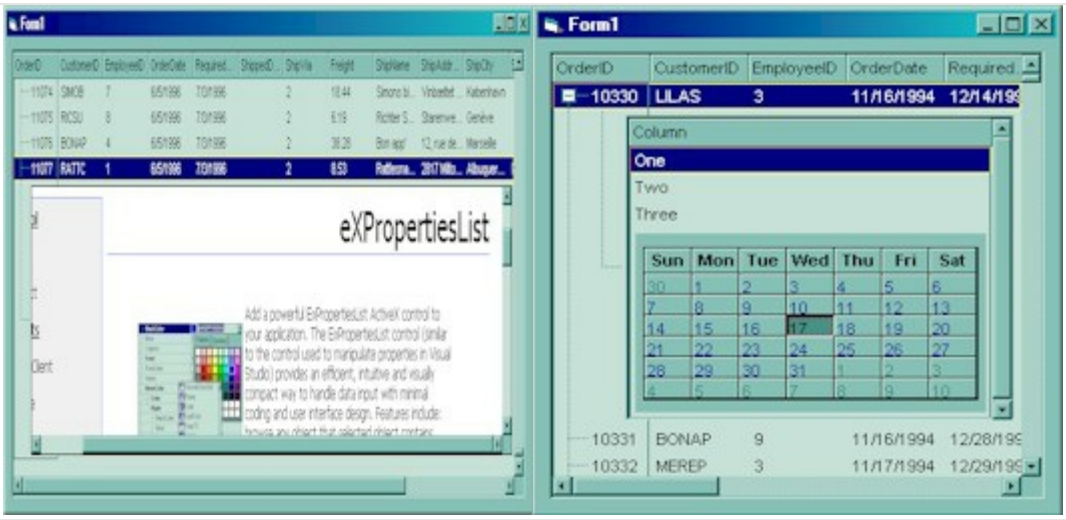
Next step is adding the AddItem event handler:

```
Private Sub G2antt1_AddItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
    If (G2antt1.Items.ItemControlID(Item) = "https://www.exontrol.com") Then
        ' Some of controls like the WEB control, requires some additional window styles ( like WS_HSCROLL and WS_VSCROLL window styles )
        ' for the window that host that WEB control, to allow scrolling the web page
        G2antt1.Items.ItemWindowHostCreateStyle(Item) = G2antt1.Items.ItemWindowHostCreateStyle(Item) + WS_HSCROLL + WS_VSCROLL
    End If
End Sub
```

If somehow the InsertItemControl wasn't able to create your ActiveX on some Windows platforms, and you don't know why, you can use the following

code to make sure that ActiveX control can be created properly by using (the sample is trying to add a new Microsoft RichText ActivX control into your form):

```
Controls.Add "RICHTEXT.RichtextCtrl", "rich"
```



method Items.InsertItem ([Parent as HITEM], [UserData as Variant], [Value as Variant])

Inserts a new item, and returns a handle to the newly created item.

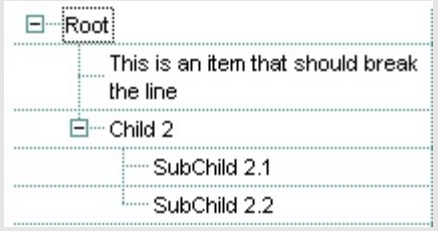
Type	Description
Parent as HITEM	A long expression that indicates the item's handle that indicates the parent item where the newly item is inserted
UserData as Variant	A Variant expression that indicates the item's extra data. Use the ItemData property to retrieve later this value.
Value as Variant	A Variant expression that indicates the cell's value on the first column, or a safe array that holds values for each column. The control displays the cell's value based on the CellValueFormat specification.

Return	Description
HITEM	Retrieves the handle of the newly created item.

Use the InsertItem property to add a new child to an item. Use the LoadXML/SaveXML methods to load/save the control's data from/to XML files. The InsertItem property fires the AddItem event. You can use the InsertItem(,,"Root") or AddItem("Root") to add a root item. An item that has no parent is a root item. To insert an ActiveX control, use the InsertControlItem property of the Items property. Use the CellValue property to specify the values for cells in the second, third columns, and so on. Use the CellValueFormat property to specify whether the value contains HTML format. Use the LockedItemCount property to lock or unlock items to the top or bottom side of the control. Use the MergeCells method to combine one or more cells in a single cell. Use the SplitCell property to split a cell. If the CauseValidateValue property is True, the control fires the ValidateValue property when the user adds a new item. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula.

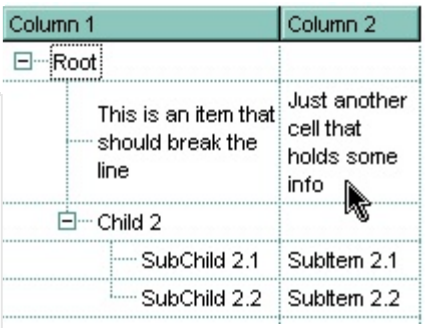
The following VB sample shows how to create a simple hierarchy (few items and one column):

```
With G2antt1
   .BeginUpdate
   .ColumnAutoResize = True
   .LinesAtRoot = exLinesAtRoot
   .FullRowSelect = False
   .MarkSearchColumn = False
```



```
    .Columns.Add "Default"
    With .Items
        Dim h As HITEM, hx As HITEM
        h = .InsertItem(, , "Root")
        hx = .InsertItem(h, , "This is an item that should break the line")
        .CellSingleLine(hx, 0) = False
        h = .InsertItem(h, , "Child 2")
        .InsertItem h, , "SubChild 2.1"
        h = .InsertItem(h, , "SubChild 2.2")
    End With
    .EndUpdate
End With
```

The following VB sample insert items and multiple columns as well:




```
With G2antt1
    .BeginUpdate
    .HeaderVisible = True
    .ColumnAutoResize = True
    .LinesAtRoot = exLinesAtRoot
    .FullRowSelect = False
    .MarkSearchColumn = False
    .Columns.Add "Column 1"
    .Columns.Add "Column 2"
    With .Items
        Dim h As HITEM, hx As HITEM
        h = .InsertItem(, , "Root")
        hx = .InsertItem(h, , Array("This is an item that should break
the line", "Just another cell that holds some info"))
        .CellSingleLine(hx, 0) = False
        .CellSingleLine(hx, 1) = False
        h = .InsertItem(h, , "Child 2")
        .InsertItem h, , Array("SubChild 2.1", "SubItem 2.1")
        h = .InsertItem(h, , Array("SubChild 2.2", "SubItem 2.2"))
    End With
    .EndUpdate
End With
```

The following VB sample inserts a child item and expands the focused item:

```
With G2antt1.Items
   .InsertItem .FocusItem, , "new child"
   .ExpandItem(.FocusItem) = True
End With
```

The following C++ sample inserts a child item and expands the focused item:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
long h = items.InsertItem( items.GetFocusItem(), vtMissing, COleVariant( "new child" ) );
items.SetExpandItem( items.GetFocusItem(), TRUE );
```

The following VB.NET sample inserts a child item and expands the focused item:

```
With AxG2antt1.Items
   Dim hItem As Integer = .InsertItem(.FocusItem, , "new child")
   .ExpandItem(.FocusItem) = True
End With
```

The following C# sample inserts a child item and expands the focused item:

```
int hItem = axG2antt1.Items.InsertItem(axG2antt1.Items.FocusItem, null, "new child");
axG2antt1.Items.set_ExpandItem(axG2antt1.Items.FocusItem, true);
```

The following VFP sample inserts a child item and expands the focused item:

```
with thisform.G2antt1.Items
   .DefaultItem = .InsertItem( .FocusItem, "", "new child" )
   .DefaultItem = .FocusItem
   .ExpandItem(0) = .t.
endwith
```

property Items.IntersectBars (ItemA as HITEM, KeyA as Variant, ItemB as HITEM, KeyB as Variant) as Long

Specifies whether two bars intersect if returns 0, if 1 A is before B and -1 if A is after bar B.

Type	Description
ItemA as HITEM	A Long expression that indicates the handle that hosts the bar A.
KeyA as Variant	A Variant expression that indicates the key of the bar A.
ItemB as HITEM	A Long expression that indicates the handle that hosts the bar B.
KeyB as Variant	A Variant expression that indicates the key of the bar B.
Long	A long expression that specifies whether the bar A intersects bar B, if 0, -1 if the bar A is before bar B, and 1 if the bar A is after bar B. Any other value being returned indicates that the ItemA, KeyA, ItemB or KeyB are not indicating a valid bar. For instance, if 2, the bar B does not exists so even the ItemB is not valid, or the item does not contain any bar with the key B

The IntersectBars property determines if two bars intersects as follows:

- if returns 0, if the two bars intersects.
- if returns -1, the bar being indicated by ItemA/KeyA is before the bar being indicated by ItemB/KeyB
- if returns 1, the bar being indicated by ItemA/KeyA is after the bar being indicated by ItemB/KeyB

The ItemBar(exBarStart) and ItemBar(exBarEnd) properties indicates the starting and ending point of the bar. The OverlaidType property indicates the way two bars get shown when they cover each other, or get intersected. For instance, you can get the bars in the item being stacked once they intersect, so the height of the item is automatically adjusted to fit the stack, if the OverlaidType property is exOverlaidBarsOffset + exOverlaidBarsStackAutoArrange. You can use the ItemBar(exBarIntersectWith), ItemBar(exBarIntersectWithAsString) or ItemBar(exBarIntersectWithCount) property to determine the bars that intersects with the current bar.

property Items.IsItemLocked (Item as HITEM) as Boolean

Returns a value that indicates whether the item is locked or unlocked.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item.
Boolean	A boolean expression that indicates whether the item is locked or unlocked.

Use the IsItemLocked property to check whether an item is locked or unlocked. A locked item is always displayed on the top or bottom side of the control no matter if the control's list is scrolled up or down. Use the LockedItemCount property to add or remove items fixed/locked to the top or bottom side of the control. Use the LockedItem property to access a locked item by its position. Use the ShowLockedItems property to show or hide the locked items.

The following VB sample prints the locked item from the cursor:

```
Private Sub G2antt1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
  On Error Resume Next
  ' Converts the container coordinates to client coordinates
  X = X / Screen.TwipsPerPixelX
  Y = Y / Screen.TwipsPerPixelY
  Dim h As HITEM
  Dim c As Long
  Dim hit As EXG2ANTTLibCtl.HitTestInfoEnum
  ' Gets the item from (X,Y)
  With G2antt1
    h = .ItemFromPoint(X, Y, c, hit)
    If Not (h = 0) Then
      If (.Items.IsItemLocked(h)) Then
        Debug.Print .Items.CellValue(h, c)
      End If
    End If
  End With
End Sub
```

The following C++ sample prints the locked item from the cursor:

```
#include "Items.h"
void OnMouseMoveG2antt1(short Button, short Shift, long X, long Y)
{
   long c = 0, hit = 0, hItem = m_g2antt.GetItemFromPoint( X, Y, &c, &hit );
   if ( hItem != 0 )
   {
      CItems items = m_g2antt.GetItems();
      if ( items.GetIsItemLocked( hItem ) )
      {
         COleVariant vtItem( hItem ), vtColumn( c );
         CString strCaption = V2S( &items.GetCellValue( vtItem, vtColumn ) ), strOutput;
         strOutput.Format( "Cell: '%s', Hit = %08X\n", strCaption, hit );
         OutputDebugString( strOutput );
      }
   }
}
```

The following VB.NET sample prints the locked item from the cursor:

```
Private Sub AxG2antt1_MouseMoveEvent(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_MouseMoveEvent) Handles AxG2antt1.MouseMoveEvent
   With AxG2antt1
      Dim i As Integer, c As Integer, hit As EXG2ANTTLib.HitTestInfoEnum
      i = .get_ItemFromPoint(e.x, e.y, c, hit)
      If Not (i = 0) Then
         With .Items
            If (.IsItemLocked(i)) Then
               Debug.WriteLine("Cell: " & .CellValue(i, c) & " Hit: " & hit.ToString())
            End If
         End With
      End If
   End With
End Sub
```

The following C# sample prints the locked item from the cursor:

```
private void axG2antt1_MouseMoveEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_MouseMoveEvent e)
```

```
{
   int c = 0;
   EXG2ANTTLib.HitTestInfoEnum hit;
   int i = axG2antt1.get_ItemFromPoint(e.x, e.y, out c, out hit);
   if (i != 0)
      if ( axG2antt1.Items.get_IsItemLocked( i ) )
      {
         object cap = axG2antt1.Items.get_CellValue(i, c);
         string s = cap != null ? cap.ToString() : "";
         s = "Cell: " + s + ", Hit: " + hit.ToString();
         System.Diagnostics.Debug.WriteLine(s);
      }
}
```

The following VFP sample prints the locked item from the cursor:

```
*** ActiveX Control Event ***
LPARAMETERS button, shift, x, y

local c, hit
c = 0
hit = 0
with thisform.G2antt1
   .Items.DefaultItem = .ItemFromPoint( x, y, @c, @hit )
   with .Items
      if ( .DefaultItem <> 0 )
         if ( .IsItemLocked( 0 ) )
            wait window nowait .CellValue( 0, c ) + " " + Str( hit )
         endif
      endif
   endwith
endwith
```

property Items.IsItemVisible (Item as HITEM) as Boolean

Checks if the specific item fits the control's client area.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item that fits the client area.
Boolean	A boolean expression that indicates whether the item fits the client area.

To make sure that an item fits the client area call EnsureVisibleItem method. Use the FirstVisibleItem, NextVisibleItem and IsItemVisible properties to get the items that fit the client area. Use the NextVisibleItem property to get the next visible item. Use the IsVisibleItem property to check whether an item fits the control's client area.

The following VB sample enumerates the items that fit the control's client area:

```
On Error Resume Next
Dim h As HITEM
Dim i As Long, j As Long, nCols As Long
nCols = G2antt1.Columns.Count
With G2antt1.Items
   h = .FirstVisibleItem
   While Not (h = 0) And .IsItemVisible(h)
      Dim s As String
      s = ""
      For j = 0 To nCols - 1
         s = s + .CellValue(h, j) + Chr(9)
      Next
      Debug.Print s
      h = .NextVisibleItem(h)
   Wend
End With
```

The following C++ sample enumerates the items that fit the control's client area:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
long hItem = items.GetFirstVisibleItem();
```

```
while ( hItem && items.GetIsItemVisible( hItem ) )
{
   OutputDebugString( V2S( &items.GetCellValue( COleVariant( hItem ), COleVariant(
long(0) ) ) ) );
   hItem = items.GetNextVisibleItem( hItem );
}
```

The following VB.NET sample enumerates the items that fit the control's client area:

```
With AxG2antt1.Items
   Dim hItem As Integer
   hItem = .FirstVisibleItem
   While Not (hItem = 0)
      If (.IsItemVisible(hItem)) Then
         Debug.Print(.CellValue(hItem, 0))
         hItem = .NextVisibleItem(hItem)
      Else
         Exit While
      End If
   End While
End With
```

The following C# sample enumerates the items that fit the control's client area:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
int hItem = items.FirstVisibleItem;
while ( ( hItem != 0 ) && (items.get_IsItemVisible(hItem)) )
{
   object strCaption = items.get_CellValue(hItem, 0);
   System.Diagnostics.Debug.WriteLine( strCaption != null ? strCaption.ToString() : "" );
   hItem = items.get_NextVisibleItem(hItem);
}
```

The following VFP sample enumerates the items that fit the control's client area:

```
with thisform.G2antt1.Items
   .DefaultItem = .FirstVisibleItem
   do while ( ( .DefaultItem <> 0 ) and ( .IsItemVisible( 0 ) ) )
      wait window .CellValue( 0, 0 )
```

```
      .DefaultItem = .NextVisibleItem( 0 )
   enddo
endwith
```

property Items.ItemAllowSizing(Item as HITEM) as Boolean

Retrieves or sets a value that indicates whether a user can resize the item at run-time.

Type	Description
Item as HITEM	A HITEM expression that indicates the handle of the item that can be resized.
Boolean	A Boolean expression that specifies whether the user can resize the item at run-time.

By default, the user can resize the item at run-time using mouse movements. Use the ItemAllowSizing property to specify whether a user can resize the item at run-time. Use the ItemsAllowSizing property to specify whether all items are resizable or not. Use the ItemHeight property to specify the height of the item. An item is resizable if the ItemAllowSizing property is True, or if the ItemsAllowSizing property is True (that means all items are resizable), and the ItemAllowSizing property is not False. For instance, if your application requires all items being resizable but only few of them being not resizable, you can have the ItemsAllowSizing property on True, and for those items that are not resizable, you can call the ItemAllowSizing property on False. The user can resize an item by moving the mouse between two items, so the vertical split cursor shows up, click and drag the mouse to the new position. Use the CellSingleLine property to specify whether the cell displays its caption using multiple lines. The ScrollBySingleLine property is automatically set on True, as soon as the user resizes an item.

property Items.ItemAppearance(Item as HITEM) as AppearanceEnum

Specifies the item's appearance when the item hosts an ActiveX control.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item that was previously created by InsertControlItem property.
AppearanceEnum	An AppearanceEnum expression that indicates the item's appearance.

Use the ItemAppearance property to specify the item's appearance if the item is of ActiveX type. Use the InsertControlItem property to insert an ActiveX control inside. Use the ItemObject property to access the object being created by the InsertControlItem property. Use the ItemHeight property to specify the height of the item when containing an ActiveX control.

property Items.ItemBackColor(Item as HITEM) as Color

Retrieves or sets a background color for a specific item.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item.
Color	A color expression that indicates the item's background color. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the

The ItemBackColor property specifies the background or the visual appearance for the item's background on the columns/item section. Use the CellBackColor property to change the cell's background color. To change the background color of the entire control you can call BackColor property of the control. Use the ClearItemBackColor property to clear the item's background color, after setting using the ItemBackColor property. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula. The ItemBackColor property of the Chart object specifies the item's background or visual appearance for the chart area.

In VB.NET or C# you require the following functions until the .NET framework will provide:

You can use the following VB.NET function:

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
   Dim i As Long
i = c.R
i = i + 256 * c.G
i = i + 256 * 256 * c.B
ToUInt32 = Convert.ToUInt32(i)
End Function
```

You can use the following C# function:

```
private UInt32 ToUInt32(Color c)
{
   long i;
```

```
    i = c.R;
    i = i + 256 * c.G;
    i = i + 256 * 256 * c.B;
    return Convert.ToUInt32(i);
}
```

The following C# sample changes the background color for the focused item:

```
axG2antt1.Items.set_ItemBackColor(axG2antt1.Items.FocusItem, ToUInt32(Color.Red) );
```

The following VB.NET sample changes the background color for the focused item:

```
With AxG2antt1.Items
    .ItemBackColor(.FocusItem) = ToUInt32(Color.Red)
End With
```

The following C++ sample changes the background color for the focused item:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetItemBackColor( items.GetFocusItem(), RGB(255,0,0) );
```

The following VFP sample changes the background color for the focused item:

```
with thisform.G2antt1.Items
    .DefaultItem = .FocusItem
    .ItemBackColor( 0 ) = RGB(255,0,0)
endwith
```

Use the following VB sample changes the background color for the cells in the first column, when adding new items:

```
Private Sub G2antt1_AddItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
    G2antt1.Items.CellBackColor(Item, o) = vbBlue
End Sub
```

property Items.ItemBar(Item as HITEM, Key as Variant, Property as ItemBarPropertyEnum) as Variant

Gets or sets a bar property.

Type	Description
Item as HITEM	A long expression that indicates the the handle of the item that hosts the bar. If the Item parameter is 0, it indicates all bars. In this case the DefaultItem property should be zero (by default), else it refers the item being indicated by DefaultItem (/COM version only) property.
Key as Variant	A String expression that indicates the key of the bar being accessed. If missing, the Key parameter is empty. If the Item has only a single Bar you may not use the Key parameter, else an unique key should be used to allow multiple bars inside the item. The Key may include a pattern with wild characters as *,?,# or [], if the Key starts with "<" and ends on ">" aka "<K*>" which indicates all bars with the key K or starts on K. The pattern may include a space which divides multiple patterns for matching. For instance "<A* *K>" indicates all keys that start on A and all keys that end on K.
Property as ItemBarPropertyEnum	An ItemBarPropertyEnum expression that indicates the property being accessed
Variant	A Variant expression that indicates the property's value.

Use the ItemBar property to access properties related to the bars being shown in the item. Use the AddBar property to add new bars to the item. Use the FirstVisibleDate property to specify the first visible date in the chart area. Use the RemoveBar method to remove a bar from an item. Use the ClearBars method to remove all bars in the item. Use the Refresh method to refresh the chart. For instance, you can use the ItemBar(,,exBarToolTip) property to specify a tooltip for a bar, or ItemBar(,,exBarItemParent) property to move the bar from an item to another item. The AllowCellValueToItemBar property allows the cells to display properties of the bars. Use the SelectOnClick property to disable selecting new items when the user clicks the chart area. The FindBar method looks for the item that hosts a specified bar. You can use the Def property to define the default values for bar's ItemBar properties. The ItemBarEx property gets or sets the property's bar that matches the criteria.

Based on the values of the Item and Key parameters the ItemBar property changes a property for none, one or multiple bars as follows:

- **ItemBar(0,"<*>",Property) = Value** changes the Property of all bars in the chart.

- **ItemBar(0,"<pattern>",Property) = Value** changes the Property of all bars in the chart that match a specified pattern using wild characters as *,?,# or [].
- **ItemBar(Item,"<*>",Property) = Value** changes the Property of all bars in the item.
- **ItemBar(Item,"<pattern>",Property) = Value** changes the Property of all bars in the item that match a specified pattern using wild characters as *,?,# or []

The pattern may include the space character which indicates multiple patterns to be used when matching. For instance "A* *K" indicates all keys that starts on A and all keys that ends on K. If not using a pattern, the ItemBar changes the property for specified key in all items if 0 is used for Item, or single Item if a valid handle is used on the Item parameter.

Here's few samples of using the set ItemBar property:

- *ItemBar(Item,"K1",Property) = Value changes the Property of the bar K1 from the specified Item.*
- *ItemBar(0,"K1",Property) = Value changes the Property of the bar K1 from the entire chart.*
- *ItemBar(0,"<A* K*>",Property) = Value changes the Property of all bars from the chart with the Key A or K or starts with A or K.*
- *ItemBar(0,"<*K>",Property) = Value changes the Property of all bars from the chart with the Key K or ends on K.*
- *ItemBar(Item,"<K*>",Property) = Value changes the Property of all bars from the specified Item with the Key K or starts on K.*
- *ItemBar(Item,"<K??>",Property) = Value changes the Property of all bars from the specified Item with the Key of 3 characters and starts with K.*

Currently, the single read-only property that supports pattern for the Key parameter is **exBarsCount**, which counts the bars as follows:

- **ItemBar(0,"<*>",exBarsCount)** counts all bars in the chart.
- **ItemBar(0,"<pattern>",exBarsCount)** counts all bars in the chart that match a specified pattern using wild characters as *,?,# or [].
- **ItemBar(Item,"<*>",exBarsCount) counts** all bars in the giving Item.
- **ItemBar(Item,"<pattern>",exBarsCount)** counts all bars in the item that match a specified pattern using wild characters as *,?,# or [].

The pattern may include the space character which indicates multiple patterns to be used when matching. For instance "A* *K" indicates all keys that start on A and all keys that end on K. For any other property, the ItemBar property returns the bar's property for the first matching bar.

Here's few samples of using the get ItemBar(exBarsCount) property:

- *ItemBar(Item,"K1",exBarsCount) gets the count of the bar K1 from the specified Item. This could be 0, if K1 is not found or 1, if the K1 is found on the Item, as an item could hold a single bar with the same Key.*
- *ItemBar(0,"K1",exBarsCount) counts all bars K1 from the entire chart.*
- *ItemBar(Item,"<*>",exBarsCount) counts all bars in the specified item.*
- *ItemBar(Item,"",exBarsCount) is equivalent with ItemBar(Item,"<*>",exBarsCount).*
- *ItemBar(0,"<*>",exBarsCount) counts all bars from the entire chart.*
- *ItemBar(0,"",exBarsCount) is equivalent with ItemBar(0,"<*>",exBarsCount).*
- *ItemBar(0,"<A* K*>",exBarsCount) gets the count of all bars from the chart with the Key A or K or starts with A or K.*
- *ItemBar(0,"<*K>",exBarsCount) gets the number of bars from the chart with the Key K or ends on K.*
- *ItemBar(Item,"<K*>",exBarsCount) counts all bars from the specified Item with the Key K or starts on K.*
- *ItemBar(Item,"<K??>",exBarsCount) counts all bars from the specified Item with the Key of 3 characters and starts with K.*

The /NET Assembly version defines get/set shortcut properties as follow (they start with get_ or set_ keywords):

- **BarName** : String, retrieves or sets a value that indicates the name of the bar
- **BarStart** : DateTime, retrieves or sets a value that indicates the start of the bar
- **BarEnd** : DateTime, retrieves or sets a value that indicates the end of the bar
- **BarCaption** : String Retrieves or sets a value that indicates the caption being assigned to the bar
- **BarHAlignCaption** : AlignmentEnum, retrieves or sets a value that indicates the horizontal alignment of the caption inside the bar
- **BarVAlignCaption** : VAlignmentEnum, retrieves or sets a value that indicates the vertical alignment of the caption inside the bar
- **BarToolTip** : String, retrieves or sets a value that indicates the tooltip being shown when the cursor hovers the bar
- **BarBackColor** : Color, retrieves or sets a value that indicates the background color for the area being occupied by the bar
- **BarForeColor** : Color, retrieves or sets a value that indicates the foreground color for the caption of the bar
- **BarKey** : Object, specifies key of the bar
- **BarCanResize** : Boolean, specifies whether the user can resize the bar
- **BarCanMove** : Boolean, specifies whether the user can move the bar
- **BarPercent** : Double, specifies the percent to display the progress on the bar
- **BarPercentCaptionFormat** : String, specifies the HTML format to be displayed as percent
- **BarShowPercentCaption** : Boolean, specifies whether the percent is displayed as

caption on the bar
- **BarAlignPercentCaption** : AlignmentEnum, specifies the alignment of the percent caption on the bar
- **BarCanResizePercent** : Boolean, specifies whether the user can resize the percent at runtime
- **BarData** : Object, associates an extra data to a bar
- **BarOffset** : Integer, specifies the vertical offset where the bar is shown
- **BarTransparent** : Integer, specifies the percent of the transparency to display the bar
- **BarKeepWorkingCount** : Boolean, specifies a value that indicates whether the bar keeps constant the working units while the user moves the bar to a new position
- **BarEffort** : Double, Specifies the effort to execute an unit in the task
- **BarMinStart** : Object/DateTime, specifies the minimum value for the starting date of the bar
- **BarMaxStart** : Object/DateTime, specifies the maximum value for the starting date of the bar
- **BarMinEnd** : Object/DateTime, specifies the minimum value for the ending date of the bar
- **BarMaxEnd** : Object/DateTime, specifies the maximum value for the ending date of the bar
- **BarShowRange** : PatternEnum, indicates whether the bar shows its range where it can be moved or resized
- **BarShowRangeTransparent** : Integer, specifies the percent of the transparency to display the range of the bar
- **BarCanMoveToAnother** : Boolean, specifies whether the bar can be moved to another item
- **BarSelectable** : Boolean, specifies whether the bar can be selected
- **BarsCount** : Integer, retrieves a value that indicates the number of bars in the item
- **BarSelected** : Boolean, specifies whether the bar is selected or unselected
- **BarCanBeLinked** : Boolean, specifies whether the bar can participate to a link
- **BarCanStartLink** : Boolean, specifies whether a link can start from specified bar
- **BarCanEndLink** : Boolean, specifies whether a link can end to specified bar
- **BarWorkingCount** : Integer, specifies the count of working units in the bar
- **BarNonWorkingCount** : Integer, retrieves the count of non-working units in the bar
- **BarParent** : HITEM, specifies the handle of the parent item that displays the bar
- **BarColor** : Color, specifies the color for the bar. If used it replaces the bar's type color, for current bar only.
- **BarDuration** : Double, specifies the duration of the bar in days
- **BarMove** : Double Moves the bar by specified amount of time
- **BarStartPrev** : DateTime, retrieves the starting date of the bar before changing it
- **BarEndPrev** : DateTime, retrieves the ending date of the bar before changing it
- **BarDurationPrev** : Double, retrieves the duration or length of the bar before

For instance, You can use the get_BarColor property instead the get_ItemBar(exBarColor) property.

For instance, the following VB/NET sample changes the bar's color:

```
With Exg2antt1.Items
   .set_BarColor(.FocusItem, .get_FirstItemBar(.FocusItem), Color.Red)
End With
```

The following VB sample changes the end date for the bar in the first visible item (in this sample we consider that AddBar method was used with the Key parameter as being empty) :

```
With G2antt1.Items
   .ItemBar(.FirstVisibleItem, "", exBarEnd) = "6/19/2005"
End With
```

The following C++ sample changes the end date for the bar in the first visible item:

```
CItems items = m_g2antt.GetItems();
items.SetItemBar( items.GetFirstVisibleItem(), COleVariant(""), 2 /*exBarEnd*/,
COleVariant("6/19/2005") );
```

The following VB.NET sample changes the end date for the bar in the first visible item:

```
With AxG2antt1.Items
   .ItemBar(.FirstVisibleItem, "", EXG2ANTTLib.ItemBarPropertyEnum.exBarEnd) =
"6/19/2005"
End With
```

The following C# sample changes the end date for the bar in the first visible item:

```
axG2antt1.Items.set_ItemBar(axG2antt1.Items.FirstVisibleItem, "",
EXG2ANTTLib.ItemBarPropertyEnum.exBarEnd, "6/19/2005");
```

The following VFP sample changes the end date for the bar in the first visible item:

```
with thisform.G2antt1.Items
   .DefaultItem = .FirstVisibleItem
   thisform.G2antt1.Template = "Items.ItemBar(0,`" + _key + "`,2 ) = `20/07/2005`"
```

```
endwith
```

where the _key is the key of the bar being resized.

The VFP sample uses the Template property in order to execute the ItemBar property, else some version of VFP could fire "Function argument, value, type, or count is invalid". The sample builds the script:

Items.ItemBar(0,_key,2) = `20/07/2005`

This way the ItemBar property for the default item is invoked.

property Items.ItemBarEx(Criteria as Variant, Property as ItemBarPropertyEnum) as Variant

Gets or sets the property's bar that matches the criteria.

Type	Description
Criteria as Variant	A String expression that defines the criteria / formula to select/query the bars in the chart, or a Boolean expression that specifies that all or none bars are selected to be queried. For instance, "cellstate(0) = 1" queries all bars hosted by items whose check-box in the column with the index 0, are checked, or "itemisselected and itembar(0) = `Task`" queries all Task bars from the selected items. The Criteria parameter of the ItemBarEx property supports predefined functions and keywords as defined bellow.
Property as ItemBarPropertyEnum	A ItemBarPropertyEnum expression that defines the bar's property to be queried. For instance, you can use exBarsCount to query the number of bars that matches the criteria.
Variant	For the get_ItemBarEx property, it indicates the value of the bar's property being found (for instance, if the Property is exBarsCount, the ItemBarEx property returns a numeric value that specifies the number of bars that matches the criteria). For set_ItemBarEx property it could be a string expression that defines the formula to change the bar's property, or any other value to assign to the bar's property for all bars that matches the criteria. For instance, "value + 1", indicates the previously value plus one. The Value parameter of the ItemBarEx property supports predefined functions and keywords as defined bellow.

The ItemBarEx property is an extension of ItemBar property, that allows changing the properties for a set of bars, using expressions. For instance, you want to select all bars whose value on the Country column is France, or move all bars of type "Task", change the color for all checked items, change the percent for selected bars, and so on.

Compared with ItemBar property that can access bars based on bar's Key only, the ItemBarEx property can access bars based on any:

- bar's property specified by the ItemBar property. For instance, "itembar(0) = `Task`" specifies all Task bars (0 indicates the value of exBarName property), or "itembar(257)" specifies all selected bars (257 defines the value of exBarSelected

property), or "itembar(0) = `Task`and itembar(257)" indicates all Task bars being selected
- caption, value or user-data associated with any cell in the Items section of the control. For instance, "cellcaption(12) = `France`" queries all bars hosted by items that have France on the column with the index 12.
- check-box / radio-button state of any cell on any column. For instance, "cellstate(0) = 1" queries all bars hosted by checked-items in the column with the index 0.
- item's extra user-data being defined by the ItemData property. For instance, "itemdata = `the item data`" queries all bars hosted by items with the user data set on "the item data".
- item's selection state. For instance, "itemisselected" queries all bars hosted by selected items
- item's focusing state. For instance, "itemisfocused and itembar(0) = `Task`" queries all Task bars hosted by focused item.
- item's level, that defines how many parent items the item has. For instance ""not(itemlevel=0)"" queries all bars hosted on child items. A root item's level is 0, while a child item has the level of its parent item plus one.

For instance:

- ItemBarEx("itembar(0) = `Task`",exBarMove) = 1, moves one-day forward all Task bars.
- ItemBarEx("itembar(0) = `Task`",exBarMove) = -2, moves two-day backward all Task bars.
- ItemBarEx("itembar(0) = `Task`",exBarEnd) = "value + 2", resizes all Task bars (adds a 2 days to ending-margin of each Task bar)
- ItemBarEx(True,exBarTransparent) = "cellstate(0) = 1 ? 0 : 100" hides shows bars being checked, and hides those are un-checked.
- ItemBarEx("(itembar(0) = `Task`)",exBarMove) = "#8/3/2017# - itembar(1)" moves all Task bars, so they all start at the same date-time #8/3/2017#
- ItemBarEx("(itembar(0) = `Task`)",exBarMove) = "#8/14/2017# - itembar(2)" moves all Task bars, so they all end at the same date-time #8/14/2017#
- ItemBarEx("itemisselected and itembar(exBarName) like `Task*`",exBarPercent100) = "value + 1", adds 1% (percent) to each "Task" bar found in the selected items.

The Criteria and Value parameters of the ItemBarEx property support the following pre-defined functions:

- **cellcaption** (unary operator) retrieves the value of the CellCaption property. The single-parameter of the cellcaption operator must be of numeric type, specifying the index of the column. For instance, "cellcaption(12) = `France`" defines all bars hosted by items where France is found on the column with the index 12
- **celldata** (unary operator) retrieves the value of the CellData property. The single-parameter of the celldata operator must be of numeric type, specifying the index of the

column. For instance, "celldata(0) = `the cell data`" defines all bars hosted by items whose cell's data is "the cell data" on the column with the index 0.

- **cellstate** (unary operator) retrieves the value of the CellState property. The single-parameter of the cellvalue operator must be of numeric type, specifying the index of the column. For instance, "cellstate(0) = 1" queries all bars hosted by checked-items in the column with the index 0.
- **cellvalue** (unary operator) retrieves the value of the CellValue property. The single-parameter of the cellvalue operator must be of numeric type, specifying the index of the column. For instance, "cellvalue(1) = cellvalue(2)" defines all bars hosted by items who have the same value on columns with the index 1 and 2.
- **itembar** (unary operator) retrieves the value of the ItemBar property. The single-parameter of the itembar operator must be of numeric type, specifying any value listed on the ItemBarPropertyEnum type. For instance, "itembar(0) = `Task`" defines all Task bars.

The Criteria and Value parameters of the ItemBarEx property support the following pre-defined keywords:

- **itemdata** keyword returns the item's user data. The ItemData property retrieves or sets the extra data for a specific item. For instance, "itemdata = `the item data`" queries all bars hosted by items with the user data set on "the item data".
- **itemisfocused** keyword returns a boolean value that indicates whether the item is focused. The FocusItem property returns the handle of the item that has the focus. At any time, the control can have a single item with the focus, instead can have more selected items. For instance, "itemisfocused and itembar(0) = `Task`" queries all Task bars from the focused item.
- **itemisselected** keyword returns a boolean value that specifies whether the item is selected. The SelectItem property specifies whether the giving item is selected or unselected. For instance, "itemisselected and itembar(0) = `Task`" queries all Task bars from the selected items .
- **itemlevel** keyword returns the item's level. This value is 0-based, which indicates the root-items. A root item's level is 0, while a child item has the level of its parent item plus one. For instance ""not(itemlevel=0)"" queries all bars hosted on child items.

Additionally, the Value parameter of the ItemBarEx property supports the following pre-defined keywords:

- **value** keyword returns the previously value of the ItemBar(Property) property. The Property parameter defines the bar's property to be queried / changed. For instance, "value + 1" increases the previously value by one-unit. Based on the type of the value, the value could be added or concatenated. For instance, if the Property is exBarCaption, the value + 1, actually appends 1 to the bar's caption.

For instance, the Criteria parameter could be:

- True, queries all bars within the chart, or False which specifies no bars will be queried.
- "itembar(0) = `Task`", queries all Task bars in the chart
- "itembar(0) = `Task`and itembar(257)" indicates all Task bars being selected
- "itembar(0) = `Task`and itemisselected" indicates all Task bars from selected items
- "itemisselected" queries all bars hosted by selected items
- "not(itemisselected) and itembar(0) = `Task`" queries all Task bars hosted by not-selected item.
- "cellcaption(12) = `France`" queries all bars hosted by items that have France on the column with the index 12.
- "cellstate(0) = 1" queries all bars hosted by checked-items in the column with the index 0.

The expression may be a combination of variables, constants, strings, dates and operators. A string is delimited by ", ` or ' characters, and inside they can have the starting character preceded by \ character, ie "\"This is a quote\"". A date is delimited by two # characters, ie #1/31/2001 10:00# means the January 31th, 2001, 10:00 AM.

This property/method supports predefined constants and operators/functions as described here.

property Items.ItemBold(Item as HITEM) as Boolean

Retrieves or sets a value that indicates whether the item should appear in bold.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item.
Boolean	A boolean expression that indicates whether the item should appear in bold.

Use ItemBold, ItemItalic, ItemUnderline or ItemStrikeOut property to apply different font attributes to the item. Use the CellItalic, CellUnderline, CellBold or CellStrikeOut property to apply different font attributes to the cell. Use the CellValueFormat property to specify an HTML caption. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula.

The following VB sample bolds the selected item:

```
Dim hOldBold As HITEM
```

```
Private Sub G2antt1_SelectionChanged()
   If Not (hOldBold = 0) Then
      G2antt1.Items.ItemBold(hOldBold) = False
   End If
   hOldBold = G2antt1.Items.SelectedItem()
   G2antt1.Items.ItemBold(hOldBold) = True
End Sub
```

The following VB sample bolds the focused item:

```
With G2antt1.Items
   .ItemBold(.FocusItem) = True
End With
```

The following C++ sample bolds the focused item:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetItemBold( items.GetFocusItem() , TRUE );
```

The following C# sample bolds the focused item:

```
axG2antt1.Items.set_ItemBold(axG2antt1.Items.FocusItem, true);
```

The following VB.NET sample bolds the focused item:

```
With AxG2antt1.Items
   .ItemBold(.FocusItem) = True
End With
```

The following VFP sample bolds the focused item:

```
with thisform.G2antt1.Items
   .DefaultItem = .FocusItem
   .ItemBold( 0 ) = .t.
endwith
```

property Items.ItemByIndex (Index as Long) as HITEM

Retrieves the handle of the item given its index in Items collection..

Type	Description
Index as Long	A long expression that indicates the index of the item.
HITEM	A long expression that indicates the item's handle.

Use the ItemByIndex to get the index of an item. Use the ItemCount property to count the items in the control. the Use the ItemPosition property to get the item's position. Use the ItemToIndex property to get the index of giving item. For instance, The ItemByIndex property is the default property for Items object, so the following statements are equivalents: G2antt1.Items(0), G2antt1.Items.ItemByIndex(0).

The following VB sample enumerates all items in the control:

```
Dim i As Long, n As Long
With G2antt1.Items
   n = .ItemCount
   For i = 0 To n - 1
      Debug.Print .ItemByIndex(i)
   Next
End With
```

The following C++ sample enumerates all items in the control:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
for ( long i = 0; i < items.GetItemCount(); i++ )
{
   COleVariant vtItem( items.GetItemByIndex( i ) ), vtColumn( long(0) );
   CString strCaption = V2S( &items.GetCellValue( vtItem, vtColumn ) ), strOutput;
   strOutput.Format( "Cell: '%s'\n", strCaption );
   OutputDebugString( strOutput );
}
```

The following VB.NET sample enumerates all items in the control:

```
With AxG2antt1
```

```
    Dim i As Integer
    For i = 0 To .Items.ItemCount - 1
        Debug.Print(.Items.CellValue(.Items(i), 0))
    Next
End With
```

The following C# sample enumerates all items in the control:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
for (int i = 0; i < items.ItemCount; i++)
{
    object caption = items.get_CellValue(items[i], 0);
    string strCaption = caption != null ? caption.ToString() : "";
    System.Diagnostics.Debug.WriteLine(strCaption);
}
```

The following VFP sample enumerates all items in the control:

```
with thisform.G2antt1.Items
    local i
    for i = 0 to .ItemCount - 1
        .DefaultItem = .ItemByIndex( i )
        wait window nowait .CellValue(0,0)
    next
endwith
```

property Items.ItemCell (Item as HITEM, ColIndex as Variant) as HCELL

Retrieves the cell's handle based on a specific column.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
ColIndex as Variant	A long expression that indicates the column's index or the cell's handle, a string expression that indicates the column's caption.
HCELL	A long expression that indicates the handle of the cell.

A cell is the intersection of an item with a column. All properties that has an Item and a ColIndex parameters are referring to a cell. The Item parameter represents the handle of an item, and the ColIndex parameter indicates an index (a numerical value, see Column.Index property) of a column , the column's caption (a string value, see Column.Caption property), or a handle to a cell. Here's few hints how to use properties with Item and ColIndex parameters:

```
G2antt1.Items.CellBold(, G2antt1.Items.ItemCell(G2antt1.Items(0), 0)) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), 0) = True
```

```
G2antt1.Items.CellBold(G2antt1.Items(0), "ColumnName") = True
```

property Items.ItemChild (Item as HITEM) as HITEM

Retrieves the first child item of a specified item.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
HITEM	A long expression that indicates the handle of the first child item.

If the ItemChild property gets 0, the item has no child items. Use this property to get the first child of an item. NextVisibleItem or NextSiblingItem to get the next visible, sibling item. The control displays a +/- sign to parent items, if the HasButtons property is not zero, the ItemChild property is not empty, or the ItemHasChildren property is True.

The following VB function recursively enumerates the item and all its child items:

```
Sub RecItem(ByVal c As EXG2ANTTLibCtl.G2antt, ByVal h As HITEM)
  If Not (h = 0) Then
    Dim hChild As HITEM
    With c.Items
      Debug.Print .CellValue(h, 0)
      hChild = .ItemChild(h)
      While Not (hChild = 0)
        RecItem c, hChild
        hChild = .NextSiblingItem(hChild)
      Wend
    End With
  End If
End Sub
```

The following C++ function recursively enumerates the item and all its child items:

```
void RecItem( CG2antt* pG2antt, long hItem )
{
  COleVariant vtColumn( (long)0 );
  if ( hItem )
  {
    CItems items = pG2antt->GetItems();
```

```
        CString strCaption = V2S( &items.GetCellValue( COleVariant( hItem ), vtColumn ) ),
strOutput;
        strOutput.Format( "Cell: '%s'\n", strCaption );
        OutputDebugString( strOutput );

        long hChild = items.GetItemChild( hItem );
        while ( hChild )
        {
            RecItem( pG2antt, hChild );
            hChild = items.GetNextSiblingItem( hChild );
        }
    }
}
```

The following VB.NET function recursively enumerates the item and all its child items:

```
Shared Sub RecItem(ByVal c As AxEXG2ANTTLib.AxG2antt, ByVal h As Integer)
    If Not (h = 0) Then
        Dim hChild As Integer
        With c.Items
            Debug.WriteLine(.CellValue(h, 0))
            hChild = .ItemChild(h)
            While Not (hChild = 0)
                RecItem(c, hChild)
                hChild = .NextSiblingItem(hChild)
            End While
        End With
    End If
End Sub
```

The following C# function recursively enumerates the item and all its child items:

```
internal void RecItem(AxEXG2ANTTLib.AxG2antt g2antt, int hItem)
{
    if (hItem != 0)
    {
        EXG2ANTTLib.Items items = g2antt.Items;
        object caption = items.get_CellValue( hItem, 0 );
```

```
        System.Diagnostics.Debug.WriteLine(caption != null ? caption.ToString() : "");

        int hChild = items.get_ItemChild(hItem);
        while (hChild != 0)
        {
            RecItem(g2antt, hChild);
            hChild = items.get_NextSiblingItem(hChild);
        }
    }
}
```

The following VFP function recursively enumerates the item and all its child items (recitem method):

```
LPARAMETERS h

with thisform.G2antt1
    If ( h != 0 ) Then
        local hChild
        With .Items
            .DefaultItem = h
            wait window .CellValue(0, 0)
            hChild = .ItemChild(h)
            do While (hChild != 0)
                thisform.recitem(hChild)
                hChild = .NextSiblingItem(hChild)
            enddo
        EndWith
    EndIf
endwith
```

property Items.ItemControlID (Item as HITEM) as String

Retrieves the item's control identifier that was used by InsertControlItem property.

Type	Description
Item as HITEM	A long expression that indicates the item's handle that was previously created by the InsertControlItem property.
String	A string expression that indicates the control identifier used by InsertControlItem method to create an item that hosts an ActiveX control.

The ItemControlID property retrieves the control identifier used by the InsertControlItem property. If the item was created using AddItem or InsertItem properties the ItemControlID property retrieves an empty string. For instance, the ItemControlID property can be used to check if an item contains an ActiveX control or not.

property Items.ItemCount as Long

Retrieves the number of items.

Type	Description
Long	A long value that indicates the number of items into the Items collection.

The ItemCount property counts the items in the control. Use the ItemByIndex property to access an item giving its index. Use the VisibleItemCount property to specify the number of visible items in the list. Use the ItemByIndex property to get the handle of the item giving its index. Use ChildCount to get the number of child items giving an item. Use the ItemChild property to get the first child item. Use the FirstVisibleItem property to get the first visible item. Use the NextVisibleItem property to get the next visible item. The NextSiblingItem property retrieves the next sibling of the item in the parent's child list. Use the ItemPosition property to change the item's position. Use the AddItem, InsertItem, PutItems or DataSource property to add new items to the control. Use ChildCount to get the number of child items.

The following VB.NET sample enumerates all items and bars in the control (/NET or /WPF version):

```
With Exg2antt1
   Dim i, h As Integer, key As Object
   For i = 0 To .Items.ItemCount - 1
      h = .Items(i)
      key = .Items.get_FirstItemBar(h)
      While TypeOf key Is String
         Debug.Print("Key = " & key & ", Item " & .Items.get_CellCaption(h, 0))
         key = CStr(.Items.get_NextItemBar(h, key))
      End While
   Next
End With
```

The following C# sample enumerates all items and bars in the control (/NET or /WPF version):

```
for (int i = 0; i < exg2antt1.Items.ItemCount; i++)
{
   int h = exg2antt1.Items[i];
   object key = exg2antt1.Items.get_FirstItemBar(h);
```

```
    while (key != null)
    {
        System.Diagnostics.Debug.Print("Key = " + key + ", Item " +
exg2antt1.Items.get_CellCaption(h, 0));
        key = exg2antt1.Items.get_NextItemBar(h, key);
    }
}
```

The following VB sample enumerates all items in the control:

```
Dim i As Long, n As Long
With G2antt1.Items
    n = .ItemCount
    For i = 0 To n - 1
        Debug.Print .ItemByIndex(i)
    Next
End With
```

The following C++ sample enumerates all items in the control:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
for ( long i = 0; i < items.GetItemCount(); i++ )
{
    COleVariant vtItem( items.GetItemByIndex( i ) ), vtColumn( long(0) );
    CString strCaption = V2S( &items.GetCellValue( vtItem, vtColumn ) ), strOutput;
    strOutput.Format( "Cell: '%s'\n", strCaption );
    OutputDebugString( strOutput );
}
```

The following VB.NET sample enumerates all items in the control:

```
With AxG2antt1
    Dim i As Integer
    For i = 0 To .Items.ItemCount - 1
        Debug.Print(.Items.CellValue(.Items(i), 0))
    Next
End With
```

The following C# sample enumerates all items in the control:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
for (int i = 0; i < items.ItemCount; i++)
{
   object caption = items.get_CellValue(items[i], 0);
   string strCaption = caption != null ? caption.ToString() : "";
   System.Diagnostics.Debug.WriteLine(strCaption);
}
```

The following VFP sample enumerates all items in the control:

```
with thisform.G2antt1.Items
   local i
   for i = 0 to .ItemCount - 1
      .DefaultItem = .ItemByIndex( i )
      wait window nowait .CellValue(0,0)
   next
endwith
```

property Items.ItemData(Item as HITEM) as Variant

Retrieves or sets the extra data for a specific item.

Type	Description
Item as HITEM	A long expression that indicates the item's handle that has associated some extra data.
Variant	A variant value that indicates the item's extra data.

Use the ItemData property to assign an extra value to an item. Use CellData property to associate an extra data with a cell. The ItemData and CellData are of Variant type, so you will be able to save here what ever you want: numbers, objects, strings, and so on. The user data is only for user use. The control doesn't use this value. Use the Data property to assign an extra data to a column. For instance, you can use the RemoveItem event to release any extra data that is associated to the item.

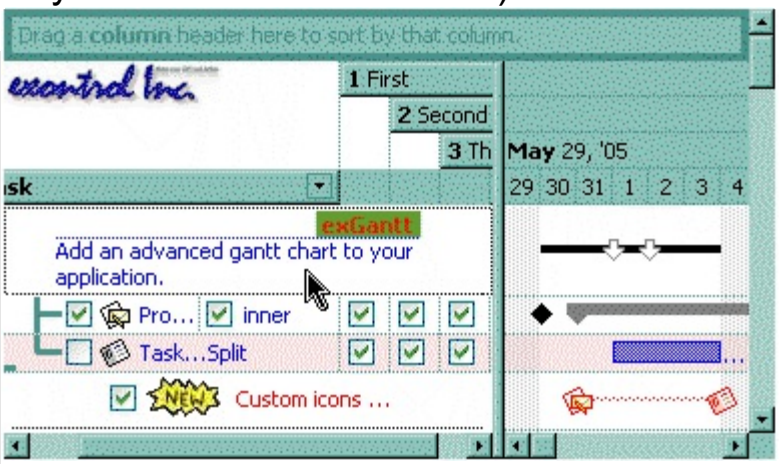
property Items.ItemDivider(Item as HITEM) as Long

Specifies whether the item acts like a divider item. The value indicates the index of column used to define the divider's title.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
Long	A long expression that indicates the column's index.

A divider item uses the item's client area to display a single cell. The ItemDivider property specifies the index of the cell being displayed. In other words, the divider item merges the item cells into a single cell. Use the ItemDividerLine property to define the line that underlines the divider item. Use the LockedItemCount property to lock items on the top or bottom side of the control. Use the MergeCells method to combine two or multiple cells in a single cell. Use the SelectableItem property to specify the user can select an item. A divider item has sense for a control with multiple columns.

The following VB sample adds a divider item that's locked to the top side of the control (Before running this sample please make sure that your control has columns):




```
With G2antt1
   .BeginUpdate
   .DrawGridLines = exNoLines
   With .Items
      .LockedItemCount(TopAlignment) = 1
      Dim h As HITEM
      h = .LockedItem(TopAlignment, 0)
      .ItemDivider(h) = 0
      .ItemHeight(h) = 22
      .CellValue(h, 0) = "<b>Total</b>:
$12.344.233"
      .CellValueFormat(h, 0) = exHTML
      .CellHAlignment(h, 0) = RightAlignment
   End With
   .EndUpdate
End With
```

The following C++ sample adds a divider item, that's not selectable too:

```
#include "Items.h"
```

```
CItems items = m_g2antt.GetItems();
long i = items.AddItem( COleVariant("divider item") );
items.SetItemDivider( i, 0 );
items.SetSelectableItem( i, FALSE );
```

The following C# sample adds a divider item, that's not selectable too:

```
int i = axG2antt1.Items.AddItem("divider item");
axG2antt1.Items.set_ItemDivider(i, 0);
axG2antt1.Items.set_SelectableItem(i, false);
```

The following VB.NET sample adds a divider item, that's not selectable too:

```
With AxG2antt1.Items
    Dim i As Integer
    i = .AddItem("divider item")
    .ItemDivider(i) = 0
    .SelectableItem(i) = False
End With
```

The following VFP sample adds a divider item, that's not selectable too:

```
with thisform.G2antt1.Items
    .DefaultItem = .AddItem("divider item")
    .ItemDivider(0) = 0
    .SelectableItem(0) = .f.
endwith
```

property Items.ItemDividerLine(Item as HITEM) as DividerLineEnum

Defines the type of line in the divider item.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
DividerLineEnum	A DividerLineEnum expression that indicates the type of the line in the divider item.

By default, the ItemDividerLine property is SingleLine. The ItemDividerLine property specifies the type of line that underlines a divider item. Use the ItemDivider property to define a divider item. Use the ItemDividerLine and ItemDividerAlignment properties to define the style of the line into the divider item. Use the CellMerge property to merge two or more cells.

property Items.ItemDividerLineAlignment(Item as HITEM) as DividerAlignmentEnum

Specifies the alignment of the line in the divider item.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
DividerAlignmentEnum	A DividerAlignmentEnum expression that specifies the line's alignment.

By default, the ItemDividerLineAlignment property is DividerBottom. The Use the ItemDividerLine and ItemDividerLineAlignment properties to define the style of the line into a divider item. Use the ItemDivider property to define a divider item.

property Items.ItemFiltered (Item as HITEM) as Boolean

Checks whether the item is included in the control's filter.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item.
Boolean	A boolean expression that indicates whether the item is filtered.

Use the ItemFiltered property to check whether an item is included in the control's filter. Use the FilterType property to specify the type of filter that's applied to a column. The ApplyFilter method should be called to update the control's content after changing the Filter or FilterType property. The ItemCount property counts the items in the control's list. Use the ItemByIndex property to access an item giving its index.

The following VB sample enumerates all items that are not included in the list when a filter is applied:

```
Dim i As Long
With G2antt1.Items
   For i = 0 To .ItemCount - 1
      Dim h As EXG2ANTTLibCtl.HITEM
      h = .ItemByIndex(i)
      If (Not .ItemFiltered(h)) Then
         Debug.Print .CellValue(h, 0)
      End If
   Next
End With
```

The following C++ sample enumerates all items that are not included in the list when a filter is applied:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
for ( long i = 0; i < items.GetItemCount(); i++ )
{
   long hItem = items.GetItemByIndex( i );
   if ( !items.GetItemFiltered( hItem ) )
   {
      COleVariant vtItem( hItem ), vtColumn( long(0) );
```

```
        CString strFormat;
        strFormat.Format( "'%s' is not included\r\n", V2S( &items.GetCellValue( vtItem,
vtColumn ) ) );
        OutputDebugString( strFormat );
    }
}
```

The following VB.NET sample enumerates all items that are not included in the list when a filter is applied:

```
Private Sub AxG2antt1_ClickEvent(ByVal sender As Object, ByVal e As System.EventArgs)
Handles AxG2antt1.ClickEvent
    Dim i As Long
    With AxG2antt1.Items
        For i = 0 To .ItemCount - 1
            Dim h As Integer = .ItemByIndex(i)
            If (Not .ItemFiltered(h)) Then
                Dim cellValue As Object = .CellValue(h, 0)
                Dim strValue As String = ""
                If Not (cellValue Is Nothing) Then
                    strValue = cellValue.ToString()
                End If
                Debug.WriteLine(strValue)
            End If
        Next
    End With
End Sub
```

The following C# sample enumerates all items that are not included in the list when a filter is applied:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
for (int i = 0; i < items.ItemCount; i++)
    if (!items.get_ItemFiltered(items[i]))
    {
        object cellValue = axG2antt1.Items.get_CellValue(i, 0);
        string strValue = cellValue != null ? cellValue.ToString() : "";
        System.Diagnostics.Debug.WriteLine(strValue);
```

```
}
```

The following VFP sample enumerates all items that are not included in the list when a filter is applied:

```
with thisform.G2antt1.Items
   local i
   For i = 0 To .ItemCount - 1
      local h
      h = .ItemByIndex(i)
      If (!.ItemFiltered(h)) Then
         wait window nowait .CellValue(h, 0)
      EndIf
   Next
endwith
```

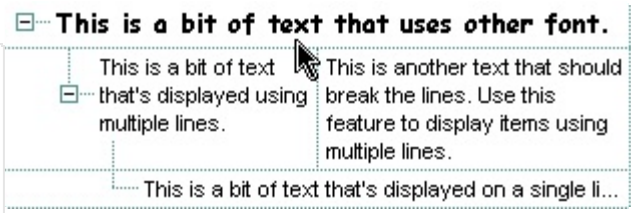
property Items.ItemFont (Item as HITEM) as IFontDisp

Retrieves or sets the item's font.

Type	Description
Item as HITEM	A long expression that specifies the item's handle.
IFontDisp	A Font object that specifies the item's font.

By default, the ItemFont property is nothing. If the ItemFont property is nothing, the item uses the control's font. Use the ItemFont property to define a different font for the item. Use the CellFont and ItemFont properties to specify different fonts for cells or items. Use the CellBold, CellItalic, CellUnderline, CellStrikeout, ItemBold, ItemUnderline, ItemStrikeout, ItemItalic or CellValueFormat to specify different font attributes. Use the ItemHeight property to specify the height of the item. Use the Refresh method to refresh the control's content on the fly. Use the BeginUpdate and EndUpdate methods if you are doing multiple changes, so no need for an update each time a change is done.

The following VB sample changes the font for the focused item:




```
With G2antt1.Items
    .ItemFont(.FocusItem) = G2antt1.Font
    With .ItemFont(.FocusItem)
        .Name = "Comic Sans MS"
        .Bold = True
    End With
End With
G2antt1.Refresh
```

The following C++ sample changes the font for the focused item:

```
#include "Items.h"
#include "Font.h"
CItems items = m_g2antt.GetItems();
items.SetItemFont( items.GetFocusItem(), m_g2antt.GetFont().m_lpDispatch );
COleFont font = items.GetItemFont( items.GetFocusItem() );
font.SetName( "Comic Sans MS" );
font.SetBold( TRUE );
m_g2antt.Refresh();
```

The following VB.NET sample changes the font for the focused item:

```
With AxG2antt1.Items
   .ItemFont(.FocusItem) = IFDH.GetIFontDisp(AxG2antt1.Font)
   With .ItemFont(.FocusItem)
      .Name = "Comic Sans MS"
      .Bold = True
   End With
End With
AxG2antt1.CtlRefresh()
```

where the IFDH class is defined like follows:

```
Public Class IFDH
   Inherits System.Windows.Forms.AxHost

   Sub New()
      MyBase.New("")
   End Sub

   Public Shared Function GetIFontDisp(ByVal font As Font) As Object
      GetIFontDisp = AxHost.GetIFontFromFont(font)
   End Function

End Class
```

The following C# sample changes the font for the focused item:

```
axG2antt1.Items.set_ItemFont( axG2antt1.Items.FocusItem, IFDH.GetIFontDisp(
axG2antt1.Font ) );
stdole.IFontDisp spFont = axG2antt1.Items.get_ItemFont(axG2antt1.Items.FocusItem );
spFont.Name = "Comic Sans MS";
spFont.Bold = true;
axG2antt1.CtlRefresh();
```

where the IFDH class is defined like follows:

```
internal class IFDH : System.Windows.Forms.AxHost
{
   public IFDH() : base("")
```

```
    {
    }

    public static stdole.IFontDisp GetIFontDisp(System.Drawing.Font font)
    {
        return System.Windows.Forms.AxHost.GetIFontFromFont(font) as stdole.IFontDisp;
    }
}
```

The following VFP sample changes the font for the focused item:

```
with thisform.G2antt1.Items
    .DefaultItem = .FocusItem
    .ItemFont(0) = thisform.G2antt1.Font
    with .ItemFont(0)
        .Name = "Comic Sans MS"
        .Bold = .t.
    endwith
endwith
thisform.G2antt1.Object.Refresh()
```

property Items.ItemForeColor(Item as HITEM) as Color

Retrieves or sets a foreground color for a specific item.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
Color	A color expression that defines the item's foreground color.

Use the CellForeColor property to change the item's foreground color. Use the ForeColor property to change the control's foreground color. Use the ClearItemForeColor property to clear the item's foreground color.

The following VB sample changes the foreground color for cells in the first column as user add new items:

```
Private Sub G2antt1_AddItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
   G2antt1.Items.CellForeColor(Item, o) = vbBlue
End Sub
```

In VB.NET or C# you require the following functions until the .NET framework will provide:

You can use the following VB.NET function:

```
   Shared Function ToUInt32(ByVal c As Color) As UInt32
      Dim i As Long
   i = c.R
   i = i + 256 * c.G
   i = i + 256 * 256 * c.B
   ToUInt32 = Convert.ToUInt32(i)
   End Function
```

You can use the following C# function:

```
private UInt32 ToUInt32(Color c)
{
   long i;
   i = c.R;
   i = i + 256 * c.G;
   i = i + 256 * 256 * c.B;
```

```
    return Convert.ToUInt32(i);
}
```

The following C# sample changes the foreground color of the focused item:

```
axG2antt1.Items.set_ItemForeColor(axG2antt1.Items.FocusItem, ToUInt32(Color.Red) );
```

The following VB.NET sample changes the foreground color of the focused item:

```
With AxG2antt1.Items
    .ItemForeColor(.FocusItem) = ToUInt32(Color.Red)
End With
```

The following C++ sample changes the foreground color of the focused item:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetItemForeColor( items.GetFocusItem(), RGB(255,0,0) );
```

The following VFP sample changes the foreground color of the focused item:

```
with thisform.G2antt1.Items
    .DefaultItem = .FocusItem
    .ItemForeColor( 0 ) = RGB(255,0,0)
endwith
```

property Items.ItemHasChildren (Item as HITEM) as Boolean

Adds an expand button to left side of the item even if the item has no child items.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
Boolean	A boolean expression that indicates whether the control adds an expand button to the left side of the item even if the item has no child items.

By default, the ItemHasChidren property is False. Use the ItemHasChildren property to build a virtual tree. Use the BeforeExpandItem event to add new child items to the expanded item. Use the ItemChild property to get the first child item, if exists. Use the ItemChild or ChildCount property to determine whether an item contains child items. The control displays a +/- sign to parent items, if the HasButtons property is not empty, the ItemChild property is not empty, or the ItemHasChildren property is True. Use the InsertItem method to insert a new child item. Use the CellData or ItemData property to assign an extra value to a cell or to an item.

The following VB sample inserts a child item as soon as user expands an item (the sample has effect only if your control contains items that have the ItemHasChildren property on True):

```
Private Sub G2antt1_BeforeExpandItem(ByVal Item As EXG2ANTTLibCtl.HITEM, Cancel As Variant)
  With G2antt1.Items
    If (.ItemHasChildren(Item)) Then
      If .ChildCount(Item) = 0 Then
        Dim h As Long
        h = .InsertItem(Item, , "new " & Item)
      End If
    End If
  End With
End Sub
```

The following VB.NET sample inserts a child item when the user expands an item that has the ItemHasChildren property on True:

```
Private Sub AxG2antt1_BeforeExpandItem(ByVal sender As Object, ByVal e As AxEXG2ANTTLib._IG2anttEvents_BeforeExpandItemEvent) Handles
```

```
AxG2antt1.BeforeExpandItem
   With AxG2antt1.Items
      If (.ItemHasChildren(e.item)) Then
         If .ChildCount(e.item) = 0 Then
            Dim h As Long
            h = .InsertItem(e.item, , "new " & e.item.ToString())
         End If
      End If
   End With
End Sub
```

The following C# sample inserts a child item when the user expands an item that has the ItemHasChildren property on True:

```
private void axG2antt1_BeforeExpandItem(object sender,
AxEXG2ANTTLib._IG2anttEvents_BeforeExpandItemEvent e)
{
   EXG2ANTTLib.Items items = axG2antt1.Items;
   if ( items.get_ItemHasChildren( e.item ) )
      if (items.get_ChildCount(e.item) == 0)
      {
         items.InsertItem(e.item, null, "new " + e.item.ToString());
      }
}
```

The following C++ sample inserts a child item when the user expands an item that has the ItemHasChildren property on True:

```
#include "Items.h"
void OnBeforeExpandItemG2antt1(long Item, VARIANT FAR* Cancel)
{
   CItems items = m_g2antt.GetItems();
   if ( items.GetItemHasChildren( Item ) )
      if ( items.GetChildCount( Item ) == 0 )
      {
         COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
         items.InsertItem( Item, vtMissing, COleVariant( "new item" ) );
      }
```

```
}
```

The following VFP sample inserts a child item when the user expands an item that has the ItemHasChildren property on True(BeforeExpandItem event):

```
*** ActiveX Control Event ***
LPARAMETERS item, cancel

with thisform.G2antt1.Items
   if ( .ItemHasChildren( item ) )
      if ( .ChildCount( item ) = 0 )
         .InsertItem(item,"","new " + trim(str(item)))
      endif
   endif
endwith
```

property Items.ItemHeight(Item as HITEM) as Long

Retrieves or sets the item's height.

Type	Description
Item as HITEM	A long expression that indicates the item's handle. If the Item is 0, setting the ItemHeight property changes the height for all items. For instance, the ItemHeight(0) = 24, changes the height for all items to be 24 pixels wide.
Long	A long value that indicates the item's height in pixels.

To change the default height of the item before inserting items to collection you can call DefaultItemHeight property of the control. The control supports items with different heights. When an item hosts an ActiveX control (was previously created by the InsertControlItem property), the ItemHeight property changes the height of contained ActiveX control. The CellSingleLine property specifies whether a cell displays its caption using multiple lines. The ItemHeight property has no effect, if the CellSingleLine property is False. If the CellSingleLine property is False, you can specify the maximum height for the item using the ItemMaxHeight property. Use the ScrollBySingleLine property when using items with different heights. Use the ItemAllowSizing property to specify whether the user can resize the item at runtime. Use the Height property to specify the height for a bar.

VBA Is it possible to change the height for all items at once?

```
With G2antt1
   .DefaultItemHeight = 12
   .Items.ItemHeight(0) = 12
End With
```

VB6 Is it possible to change the height for all items at once?

```
With G2antt1
   .DefaultItemHeight = 12
   .Items.ItemHeight(0) = 12
End With
```

VB.NET Is it possible to change the height for all items at once?

```
With Exg2antt1
   .DefaultItemHeight = 12
   .Items.set_ItemHeight(0,12)
```

```
End With
```

VB.NET for /COM Is it possible to change the height for all items at once?

```
With AxG2antt1
   .DefaultItemHeight = 12
   .Items.ItemHeight(0) = 12
End With
```

C++ Is it possible to change the height for all items at once?

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->PutDefaultItemHeight(12);
spG2antt1->GetItems()->PutItemHeight(0,12);
```

C++ Builder Is it possible to change the height for all items at once?

```
G2antt1->DefaultItemHeight = 12;
G2antt1->Items->set_ItemHeight(0,12);
```

C# Is it possible to change the height for all items at once?

```
exg2antt1.DefaultItemHeight = 12;
exg2antt1.Items.set_ItemHeight(0,12);
```

JavaScript Is it possible to change the height for all items at once?

```
<OBJECT classid="clsid:CD481F4D-2D25-4759-803F-752C568F53B7" id="G2antt1">
</OBJECT>

<SCRIPT LANGUAGE="JScript">
```

```
   G2antt1.DefaultItemHeight = 12

   G2antt1.Items.ItemHeight(0) = 12

</SCRIPT>
```

C# for /COM Is it possible to change the height for all items at once?

```
axG2antt1.DefaultItemHeight = 12;
axG2antt1.Items.set_ItemHeight(0,12);
```

X++ (Dynamics Ax 2009) Is it possible to change the height for all items at once?

```
public void init()
{
   super()


   exg2antt1.DefaultItemHeight(12)

   exg2antt1.Items().ItemHeight(0,12)

}
```

VFP Is it possible to change the height for all items at once?

```
with thisform.G2antt1
   .DefaultItemHeight = 12
   .Items.ItemHeight(0) = 12
endwith
```

dBASE Plus Is it possible to change the height for all items at once?

```
local oG2antt,var_Items

oG2antt = form.Activex1.nativeObject
oG2antt.DefaultItemHeight = 12
// oG2antt.Items.ItemHeight(0) = 12
var_Items = oG2antt.Items
```

```
with (oG2antt)
   TemplateDef = [Dim var_Items]
   TemplateDef = var_Items
   Template = [var_Items.ItemHeight(0) = 12]
endwith
```

XBasic (Alpha Five) Is it possible to change the height for all items at once?

```
Dim oG2antt as P
Dim var_Items as P

oG2antt = topparent:CONTROL_ACTIVEX1.activex
oG2antt.DefaultItemHeight = 12
' oG2antt.Items.ItemHeight(0) = 12
var_Items = oG2antt.Items
oG2antt.TemplateDef = "Dim var_Items"
oG2antt.TemplateDef = var_Items
oG2antt.Template = "var_Items.ItemHeight(0) = 12"
```

Delphi 8 (.NET only) Is it possible to change the height for all items at once?

```
with AxG2antt1 do
begin
   DefaultItemHeight := 12;
   Items.ItemHeight[0] := 12;
end
```

Delphi (standard) Is it possible to change the height for all items at once?

```
with G2antt1 do
begin
   DefaultItemHeight := 12;
   Items.ItemHeight[0] := 12;
end
```

Visual Objects Is it possible to change the height for all items at once?

```
oDCOCX_Exontrol1:DefaultItemHeight := 12
```

```
oDCOCX_Exontrol1:Items:[ItemHeight,0] := 12
```

PowerBuilder Is it possible to change the height for all items at once?

```
OleObject oG2antt

oG2antt = ole_1.Object
oG2antt.DefaultItemHeight = 12
oG2antt.Items.ItemHeight(0,12)
```

property Items.ItemItalic(Item as HITEM) as Boolean

Retrieves or sets a value that indicates whether the item should appear in italic.

Type	Description
Item as HITEM	A long expression that indicates the item's handle that uses italic font attribute.
Boolean	A boolean expression that indicates whether the item should appear in italic.

Use ItemBold, ItemItalic, ItemUnderline or ItemStrikeOut property to apply different font attributes to the item. Use the CellItalic, CellUnderline, CellBold or CellStrikeOut property to apply different font attributes to the cell. Use the CellValueFormat property to specify an HTML caption. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula.

The following VB sample makes italic the selected item:

```
Private Sub G2antt1_SelectionChanged()
  If Not (h = 0) Then G2antt1.Items.ItemItalic(h) = False
  h = G2antt1.Items.SelectedItem()
  G2antt1.Items.ItemItalic(h) = True
End Sub
```

The following VB sample makes italic the focused item:

```
With G2antt1.Items
  .ItemItalic(.FocusItem) = True
End With
```

The following C++ sample makes italic the focused item:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetItemItalic( items.GetFocusItem() , TRUE );
```

The following C# sample makes italic the focused item:

```
axG2antt1.Items.set_ItemItalic(axG2antt1.Items.FocusItem, true);
```

The following VB.NET sample makes italic the focused item:

```
With AxG2antt1.Items
    .ItemItalic(.FocusItem) = True
End With
```

The following VFP sample makes italic the focused item:

```
with thisform.G2antt1.Items
    .DefaultItem = .FocusItem
    .ItemItalic( 0 ) = .t.
endwith
```

property Items.ItemMaxHeight(Item as HITEM) as Long

Retrieves or sets a value that indicates the maximum height when the item's height is variable.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item. If the Item is 0, setting the ItemMaxHeight property changes the maximum-height for all items. For instance, the ItemMaxHeight(0) = 24, changes the maximum height for all items to be 24 pixels wide.
Long	A long value that indicates the maximum height when the item's height is variable.

By default, the ItemMaxHeight property is -1. The ItemMaxHeight property has effect only if it is greater than 0, and the item contains cells with CellSingleLine property on False. The ItemMinHeight property specifies the minimal height of the item while resizing. The CellSingleLine property specifies whether a cell displays its caption using multiple lines. The ItemHeight property has no effect, if the CellSingleLine property is False. If the CellSingleLine property is False, you can specify the maximum height for the item using the ItemMaxHeight property. Use the ItemAllowSizing property to specify whether the user can resize the item at runtime.

property Items.ItemMinHeight(Item as HITEM) as Long

Retrieves or sets a value that indicates the minimum height when the item's height is sizing.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item. If the Item is 0, setting the ItemMinHeight property changes the minimum-height for all items. For instance, the ItemMinHeight(0) = 24, changes the minimum height for all items to be 24 pixels wide.
Long	A long value that indicates the minimum height when the item's height is variable.

By default, the ItemMinHeight property is -1. The ItemMinHeight property has effect only if the item contains cells with CellSingleLine property on False. The ItemMaxHeight property specifies the maximum height of the item while resizing. The CellSingleLine property specifies whether a cell displays its caption using multiple lines. The ItemHeight property has no effect, if the CellSingleLine property is False. If the CellSingleLine property is False, you can specify the minimum height for the item using the ItemMinHeight property. Use the ItemAllowSizing property to specify whether the user can resize the item at runtime.

property Items.ItemNonworkingUnits(Item as HITEM, [InsideZoom as Variant]) as String

Gets or sets a value that indicates the formula to specify the use non-working units for the item.

Type	Description
Item as HITEM	A long expression that specifies handle of the item being changed.
InsideZoom as Variant	A Boolean expression that specifies whether the format is applied to normal view or when the item displays inside zoom units. By default, it is False, so the format is applied to all units, including the inside zoom units. If True, the specified format is applied only to visible inside units in the item. If False, the pattern and the color for non-working days is applied, else if True, the pattern and the color for non-working hours is applied.
String	A String expression that specifies the formula to determine the non-working area for the item.

By default, the ItemNonworkingUnits property is empty. The ItemNonworkingUnits property specifies custom non-working units for specified items. If the ItemNonworkingUnits property is not empty and invalid, the item provides no non-working units (all working units). If the ItemNonworkingUnits property is empty the NonworkingDays and NonworkingHours properties specify the default non-working area of the item.

The ItemNonworkingUnits property supports the following keywords:

- **value**, indicates the date-time unit to check, as a DATE type

This property/method supports predefined constants and operators/functions as described here.

For instance:

- "0", all only working units, no not-working units
- "month(value) = 1", all January is non-working
- "weekday(value) = 0", all Sundays are non-working only

The control supports the following ways of specify the non-working parts for items:

- NonworkingDays and NonworkingHours properties indicate the nonworking parts of the chart being applied to all items with the exception of those that use the

ItemNonworkingUnits property.

- AddNonworkingDate method adds custom dates as being nonworking date which is applied to all items with the exception of those that use the ItemNonworkingUnits property.
- ItemNonworkingUnits property defines the repetitive expression to specify the non-working parts in the item.
- ItemBar(exBarTreatAsNonworking) indicates whether the bar defines actually the non-working part of the item in addition to ItemNonworkingUnits property (which is required also)

property Items.ItemObject (Item as HITEM) as Object

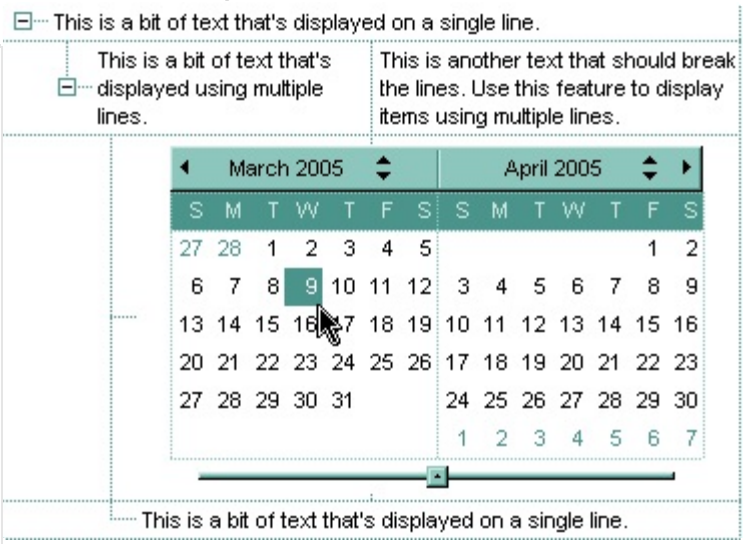
Retrieves the item's ActiveX object associated, if the item was previously created by InsertControlItem property.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item that was previously created by InsertControlItem property.
Object	An object that indicates the ActiveX hosted by the item.

Use the ItemObject to retrieve the ActiveX control created by the InsertControlItem method. Use the ItemControlID property to retrieve the control's identifier. Use the ItemHeight property to specify the item's height. If the item hosts an ActiveX control, the ItemHeight property specifies the height of the ActiveX control also.

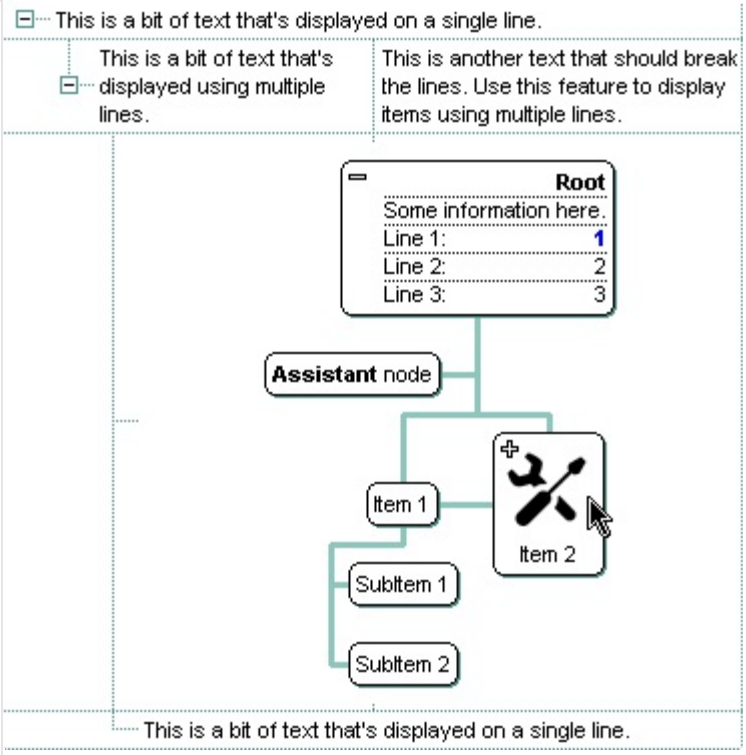
The following VB sample adds the Exontrol's ExCalendar Component:

```
With G2antt1
   .BeginUpdate
   .ScrollBySingleLine = True
   With G2antt1.Items
      Dim h As HITEM
      h = .InsertControlItem(,
"Exontrol.Calendar")
      .ItemHeight(h) = 182
      With .ItemObject(h)
         .Appearance = 0
         .BackColor = vbWhite
         .ForeColor = vbBlack
         .ShowTodayButton = False
      End With
   End With
   .EndUpdate
End With
```



The following C++ sample adds the Exontrol's ExOrgChart Component:

```
#include "Items.h"

#pragma warning( disable : 4146 )
#import <ExOrgChart.dll>

CItems items = m_g2antt.GetItems();
m_g2antt.BeginUpdate();
m_g2antt.SetScrollBySingleLine( TRUE );
COleVariant vtMissing; V_VT( &vtMissing ) =
VT_ERROR;
long h = items.InsertControlItem( 0,
"Exontrol.ChartView", vtMissing );
items.SetItemHeight( h, 182 );
EXORGCHARTLib::IChartViewPtr spChart(
items.GetItemObject(h) );
if ( spChart != NULL )
{
   spChart->BeginUpdate();
   spChart->BackColor = RGB(255,255,255);
   spChart->ForeColor = RGB(0,0,0);
   EXORGCHARTLib::INodesPtr spNodes =
spChart->Nodes;
   spNodes->Add( "Child 1", "Root", "1",
vtMissing, vtMissing );
   spNodes->Add( "SubChild 1", "1", vtMissing,
vtMissing, vtMissing );
   spNodes->Add( "SubChild 2", "1", vtMissing,
vtMissing, vtMissing );
   spNodes->Add( "Child 2", "Root", vtMissing,
vtMissing, vtMissing );
   spChart->EndUpdate();
}
m_g2antt.EndUpdate();
```



The sample uses the #import statement to include the ExOrgChart's Type Library. In this sample, the ItemObject property retrieves an IChartView object. The path to the library should be provided in case it is not located in your system folder.

The following C# sample adds the Exontrol's ExG2antt Component:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Items items = axG2antt1.Items;
axG2antt1.ScrollBySingleLine = true;
int h = items.InsertControlItem(0, "Exontrol.G2antt","");
items.set_ItemHeight(h, 182);
object g2anttInside = items.get_ItemObject(h);
if ( g2anttInside != null )
{
   EXG2ANTTLib.G2antt g2antt = g2anttInside as EXG2ANTTLib.G2antt;
   if (g2antt != null)
   {
      g2antt.BeginUpdate();
      g2antt.LinesAtRoot = EXG2ANTTLib.LinesAtRootEnum.exLinesAtRoot;
      g2antt.Columns.Add("Column 1");
      g2antt.Columns.Add("Column 2");
      g2antt.Columns.Add("Column 3");
      EXG2ANTTLib.Items itemsInside = g2antt.Items;
      int hInside = itemsInside.AddItem("Item 1");
      itemsInside.set_CellValue(hInside, 1, "SubItem 1");
      itemsInside.set_CellValue(hInside, 2, "SubItem 2");
      hInside = itemsInside.InsertItem(hInside, null, "Item 2");
      itemsInside.set_CellValue(hInside, 1, "SubItem 1");
      itemsInside.set_CellValue(hInside, 2, "SubItem 2");
      g2antt.EndUpdate();
   }
}
axG2antt1.EndUpdate();
```

The following VB.NET sample adds the Exontrol's ExOrgChart Component:

```
With AxG2antt1
   .BeginUpdate()
   .ScrollBySingleLine = True
   With .Items
      Dim hItem As Integer
      hItem = .InsertControlItem(, "Exontrol.ChartView")
```

```
    .ItemHeight(hItem) = 182
    With .ItemObject(hItem)
      .BackColor = ToUInt32(Color.White)
      .ForeColor = ToUInt32(Color.Black)
      With .Nodes
        .Add("Child 1", , "1")
        .Add("SubChild 1", "1")
        .Add("SubChild 2", "1")
        .Add("Child 2")
      End With
    End With
  End With
  .EndUpdate()
End With
```

The following VFP sample adds the Exontrol's ExGrid Component:

```
with thisform.G2antt1
  .BeginUpdate()
  .ScrollBySingleLine = .t.
  with .Items
    .DefaultItem = .InsertControlItem(0, "Exontrol.Grid")
    .ItemHeight( 0 ) = 182
    with .ItemObject( 0 )
      .BeginUpdate()
      with .Columns
        with .Add("Column 1").Editor()
          .EditType = 1 && EditType editor
        endwith
      endwith
      with .Items
        .AddItem("Text 1")
        .AddItem("Text 2")
        .AddItem("Text 3")
      endwith
      .EndUpdate()
    endwith
```

```
    endwith
    .EndUpdate()
endwith
```

property Items.ItemParent (Item as HITEM) as HITEM

Returns the handle of the parent item.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
HITEM	A long expression that indicates the handle of the parent item.

Use the ItemParent property to retrieve the parent item. Use the InsertItem property to insert child items. Use the InsertControlItem property to insert ActiveX controls. The SetParent method changes the item's parent at runtime. To verify if an item can be parent for another item you can call AcceptSetParent property. If the item has no parent the ItemParent property retrieves 0. If the ItemParent gets 0 for an item, than the item is called root. The control is able to handle more root items. To get the collection of root items you can use RootCount and RootItem properties. Use the ItemChild property to retrieve the first child item.

property Items.ItemPosition(Item as HITEM) as Long

Retrieves or sets a value that indicates the item's position in the children list.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
Long	A long expression that indicates the item's position in the children list.

The ItemPosition property gets the item's position in the children items list. You can use the ItemPosition property to change the item's position after it been added to collection. When the control sorts the tree, the item for each position can be changed, so you can use the item's handle or item's index to identify an item. Use the SortChildren method to sort the child items. Use the SortOrder property to sort a column. The FormatColumn event is fired before displaying a cell, so you can handle the FormatColumn to display anything on the cell at runtime. This way you can display the row position, you can display the value using the currency format, and so on. The FireFormatColumn property allows the control to fire the FormatColumn event for the column. The Position property specifies the position of the column.

- If your chart does *not* display a tree or a hierarchy this property is ok to be used with FormatColumn event to display the position

The following VB sample handles the FormatColumn event to display the row position:

```
Private Sub G2antt1_FormatColumn(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal
ColIndex As Long, Value As Variant)
  Value = G2antt1.Items.ItemPosition(Item)
End Sub
```

- If your chart displays a tree or a hierarchy the position of the item must be determined relative to the FirstVisibleItem as shown in the following VB sample:

```
Private Sub G2antt1_FormatColumn(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal
ColIndex As Long, Value As Variant)
  Value = G2antt1.ScrollPos(True) + RelPos(Item)
End Sub

Private Function RelPos(ByVal hVisible As Long) As Long
  With G2antt1.Items
    Dim h As Long, i As Long, n As Long
```

```
      i = 0
      n = .VisibleCount + 1
      h = .FirstVisibleItem
      While (i <= n) And h <> 0 And h <> hVisible
         i = i + 1
         h = .NextVisibleItem(h)
      Wend
      RelPos = i
   End With
End Function
```

property Items.ItemStrikeOut(Item as HITEM) as Boolean

Retrieves or sets a value that indicates whether the item should appear in strikeout.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
Boolean	A boolean expression that indicates whether the item should appear in strikeout.

If the ItemStrikeOut property is True, the cell's font is displayed with a horizontal line through it. Use ItemBold, ItemItalic, ItemUnderline or ItemStrikeOut property to apply different font attributes to the item. Use the CellItalic, CellUnderline, CellBold or CellStrikeOut property to apply different font attributes to the cell. Use the CellValueFormat property to specify an HTML caption. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula.

The following VB sample draws a horizontal line through the selected item:

```
Private Sub G2antt1_SelectionChanged()
  If Not (h = 0) Then G2antt1.Items.ItemStrikeOut(h) = False
  h = G2antt1.Items.SelectedItem()
  G2antt1.Items.ItemStrikeOut(h) = True
End Sub
```

The following VB sample draws a horizontal line through the focused item:

```
With G2antt1.Items
  .ItemStrikeOut(.FocusItem) = True
End With
```

The following C++ sample draws a horizontal line through the focused item:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetItemStrikeOut( items.GetFocusItem() , TRUE );
```

The following C# sample draws a horizontal line through the focused item:

```
axG2antt1.Items.set_ItemStrikeOut(axG2antt1.Items.FocusItem, true);
```

The following VB.NET sample draws a horizontal line through the focused item:

```
With AxG2antt1.Items
   .ItemStrikeOut(.FocusItem) = True
End With
```

The following VFP sample draws a horizontal line through the focused item:

```
with thisform.G2antt1.Items
   .DefaultItem = .FocusItem
   .ItemStrikeOut( 0 ) = .t.
endwith
```

property Items.ItemToIndex (Item as HITEM) as Long

Retrieves the index of item into Items collection given its handle.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
Long	A long expression that indicates the index of the item in Items collection.

Use the ItemToIndex property to get the item's index in the Items collection. Use ItemPosition property to change the item's position. Use the ItemByIndex property to get an item giving its index. The ItemCount property counts the items in the control. The ChildCount property counts the child items.

property Items.ItemUnderline(Item as HITEM) as Boolean

Retrieves or sets a value that indicates whether the item should appear in underline.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
Boolean	A boolean expression that indicates whether the item should appear in underline.

Use ItemBold, ItemItalic, ItemUnderline or ItemStrikeOut property to apply different font attributes to the item. Use the CellItalic, CellUnderline, CellBold or CellStrikeOut property to apply different font attributes to the cell. Use the CellValueFormat property to specify an HTML caption. Use the ConditionalFormats method to apply formats to a cell or range of cells, and have that formatting change depending on the value of the cell or the value of a formula.

The following VB sample underlines the selected item:

```
Private Sub G2antt1_SelectionChanged()
   If Not (h = 0) Then G2antt1.Items.ItemUnderline(h) = False
   h = G2antt1.Items.SelectedItem()
   G2antt1.Items.ItemUnderline(h) = True
End Sub
```

The following VB sample underlines the focused item:

```
With G2antt1.Items
   .ItemUnderline(.FocusItem) = True
End With
```

The following C++ sample underlines the focused item:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetItemUnderline( items.GetFocusItem() , TRUE );
```

The following C# sample underlines the focused item:

```
axG2antt1.Items.set_ItemUnderline(axG2antt1.Items.FocusItem, true);
```

The following VB.NET sample underlines the focused item:

```
With AxG2antt1.Items
    .ItemUnderline(.FocusItem) = True
End With
```

The following VFP sample underlines the focused item:

```
with thisform.G2antt1.Items
    .DefaultItem = .FocusItem
    .ItemUnderline( 0 ) = .t.
endwith
```

property Items.ItemWidth(Item as HITEM) as Long

Retrieves or sets a value that indicates the item's width while it contains an ActiveX control.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
Long	A long expression that indicates the item's width, when the item contains an ActiveX control.

By default, the ItemWidth property is -1. If the ItemWidth property is -1, the control resizes the ActiveX control to fit the control's client area. Use the ItemHeight property to specify the item's height. The property has effect only if the item contains an ActiveX control. Use the InsertControlItem property to insert ActiveX controls. Use the ItemObject property to retrieve the ActiveX object that's hosted by an item. Use the CellWidth property to specify the width of the cell, when it contains inner cells. Use the SplitCell property to split a cell.

The ItemWidth property is interpreted like follows:

- If the ItemWidth property is greater than zero, the ItemWidth property indicates the width in pixels of the ActiveX control. The TreeColumnIndex property indicates the column where the ActiveX control is shown. For instance, ItemWidth = 64, indicates that the width of the inside ActiveX control is 64 pixels.
- If the ItemWidth property is zero, the ActiveX control uses the full item area to display the inside ActiveX control.
- If the ItemWidth property is -1, the TreeColumnIndex property indicates the column where the ActiveX control is shown and the inside ActiveX control is shown to the end of the control.
- If the ItemWidth property is less than -32000, the formula -(ItemWidth+32000) indicates the index of the column where the inside ActiveX is displayed. For instance, -32000 indicates that the cell in the first column displays the inside ActiveX control, -32001 indicates that the cell in the second column displays the inside ActiveX control, -32002 indicates that the cell in the third column displays the inside ActiveX control, and so on.
- If the ItemWidth property is -InnerCell or ItemCell, the ItemWidth property indicates the handle of the cell that shows the inside ActiveX. This option should be used when you need to display the ActiveX control in an inner cell. Use the SplitCell property to create inner cells, to divide a cell or to split a cell. For instance, .ItemWidth(.FirstVisibleItem) = -.InnerCell(.FirstVisibleItem, 1, 1) indicates that the inside ActiveX control is shown in the second inner cell in the second column, in the first visible item. Use the CellWidth property to specify the width of the inner cell.

property Items.ItemWindowHost (Item as HITEM) as Long

Retrieves the window's handle that hosts an ActiveX control when the item was created using InsertControlItem method.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item that was previously created by InsertControlItem method.
Long	A long value that indicates the window handle that hosts the item's ActiveX.

The ItemWindowHost property retrieves the handle of the window that's the container for the item's ActiveX control. Use the InserControlItem method to insert an ActiveX control. Use the ItemObject property to access the ActiveX properties and methods. Use the hWnd property to get the handle of the control's window. The Microsoft Windows operating environment identifies each form and control in an application by assigning it a handle, or hWnd. The hWnd property is used with Windows API calls. Many Windows operating environment functions require the hWnd of the active window as an argument.

property Items.ItemWindowHostCreateStyle(Item as HITEM) as Long

Retrieves or sets a value that indicates a combination of window styles used to create the ActiveX window host.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item that was previously created by InsertControlItem method.
Long	A long value that indicates the container window's style.

The ItemWindowHostCreateStyle property specifies the window styles of the ActiveX's container window, when a new ActiveX control is inserted using the InsertControlItem method. The ItemWindowHostCreateStyle property has no effect for non ActiveX items. The ItemWindowHostCreateStyle property must be called during the AddItem event, like in the following samples. Generally, the ItemWindowHostCreateStyle property is useful to include WS_HSCROLL and WS_VSCROLL styles for a IWebBrowser control (WWW browser control), to include scrollbars in the browsed web page.

Some of ActiveX controls requires additional window styles to be added to the container window. For instance, the Web Brower added by the G2antt1.Items.InsertControlItem(, "https://www.exontrol.com") won't add scroll bars, so you have to do the following:

First thing is to declare the WS_HSCROLL and WS_VSCROLL constants at the top of your module:

```
Private Const WS_VSCROLL = &H200000
Private Const WS_HSCROLL = &H100000
```

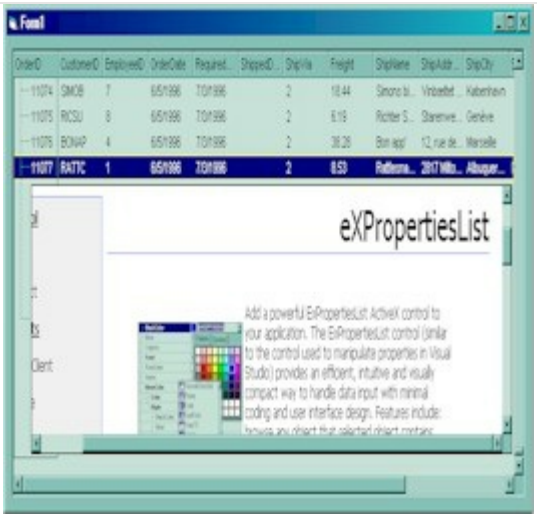
Then you need to to insert a Web control use the following lines:

```
Dim hWeb As HITEM
hWeb = G2antt1.Items.InsertControlItem(, "https://www.exontrol.com")
G2antt1.Items.ItemHeight(hWeb) = 196
```

Next step is adding the AddItem event handler:

```
Private Sub G2antt1_AddItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
    If (G2antt1.Items.ItemControlID(Item) = "https://www.exontrol.com") Then
        ' Some of controls like the WEB control, requires some additional window styles ( like WS_HSCROLL and WS_VSCROLL window styles )
        ' for the window that host that WEB control, to allow scrolling the web page
```

```
      G2antt1.Items.ItemWindowHostCreateStyle(Item) =
G2antt1.Items.ItemWindowHostCreateStyle(Item) + WS_HSCROLL + WS_VSCROLL
   End If
End Sub
```



property Items.LastVisibleItem ([Partially as Variant]) as HITEM

Retrieves the handle of the last visible item.

Type	Description
Partially as Variant	A Boolean expression that indicates whether the item is partially visible. By default, the Partially parameter is False.
HITEM	A long expression that indicates handle of the last visible item.

To get the first visible item use FirstVisibleItem property. The LastVisibleItem property retrieves the handle for the last visible item. Use the FirstVisibleItem, NextVisibleItem and IsItemVisible properties to get the items that fit the client area. Use the NextVisibleItem property to get the next visible item. Use the IsVisibleItem property to check whether an item fits the control's client area.

The following VB sample enumerates the items that fit the control's client area:

```
On Error Resume Next
Dim h As HITEM
Dim i As Long, j As Long, nCols As Long
nCols = G2antt1.Columns.Count
With G2antt1.Items
   h = .FirstVisibleItem
   While Not (h = 0) And .IsItemVisible(h)
      Dim s As String
      s = ""
      For j = 0 To nCols - 1
         s = s + .CellValue(h, j) + Chr(9)
      Next
      Debug.Print s
      h = .NextVisibleItem(h)
   Wend
End With
```

The following C++ sample enumerates the items that fit the control's client area:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
```

```
long hItem = items.GetFirstVisibleItem();
while ( hItem && items.GetIsItemVisible( hItem ) )
{
   OutputDebugString( V2S( &items.GetCellValue( COleVariant( hItem ), COleVariant(
long(0) ) ) ) );
   hItem = items.GetNextVisibleItem( hItem );
}
```

The following VB.NET sample enumerates the items that fit the control's client area:

```
With AxG2antt1.Items
   Dim hItem As Integer
   hItem = .FirstVisibleItem
   While Not (hItem = 0)
      If (.IsItemVisible(hItem)) Then
         Debug.Print(.CellValue(hItem, 0))
         hItem = .NextVisibleItem(hItem)
      Else
         Exit While
      End If
   End While
End With
```

The following C# sample enumerates the items that fit the control's client area:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
int hItem = items.FirstVisibleItem;
while ( ( hItem != 0 ) && (items.get_IsItemVisible(hItem)) )
{
   object strCaption = items.get_CellValue(hItem, 0);
   System.Diagnostics.Debug.WriteLine( strCaption != null ? strCaption.ToString() : "" );
   hItem = items.get_NextVisibleItem(hItem);
}
```

The following VFP sample enumerates the items that fit the control's client area:

```
with thisform.G2antt1.Items
   .DefaultItem = .FirstVisibleItem
   do while ( ( .DefaultItem <> 0 ) and ( .IsItemVisible( 0 ) ) )
```

```
    wait window .CellValue( 0, 0 )
    .DefaultItem = .NextVisibleItem( 0 )
  enddo
endwith
```

property Items.Link(LinkKey as Variant, Property as LinkPropertyEnum) as Variant

Gets or sets a property for a link.

Type	Description
LinkKey as Variant	A String expression that indicates the key of the link being accessed. The LinkKey may include a pattern with wild characters as *,?,# or [], if the Key starts with "<" and ends on ">" aka "<K*>" which indicates all links with the key K or starts on K. The pattern may include a space which divides multiple patterns for matching. For instance "<A* *K>" indicates all keys that start on A and all keys that end on K.
Property as LinkPropertyEnum	A LinkPropertyEnum expression that specifies the option being accessed.
Variant	A Variant value that indicates the newly value for the property.

Use the Link property to access different properties for a specified link. Use the AddLink method to add a new link between two bars. For instance, the Link(exLinkShowDir) property indicates whether the arrow of the link that specifies the direction, is shown or hidden. Use the RemoveLink method to remove a specific link. Use the FirstLink and NextLink properties to enumerate the links in the control. Use the BeginUpdate and EndUpdate methods to maintain performance while adding columns, items, bars or links. Use the HTMLPicture property to add custom size pictures. Use the LinkFromPoint property to get the key of the link from the cursor. Use the Link(,exLinkToolTip) property to specify the tooltip to be shown when the cursor hovers the link. Use the Link(exLinkGroupBars) to group the linked bars. Use the SelectOnClick property to disable selecting new items when the user clicks the chart area.

Based on the values of the Link Key parameter the Link property changes a property for none, one or multiple links as follows:

- **Link("<*>",Property) = Value** changes the Property of all links in the chart.
- **Link("<pattern>",Property) = Value** changes the Property of all links in the chart that match a specified pattern using wild characters as *,?,# or []

The pattern may include the space character which indicates multiple patterns to be used when matching. For instance "A* *K" indicates all keys that starts on A and all keys that ends on K. If not using a pattern, the Link changes the property for specified key in the chart.

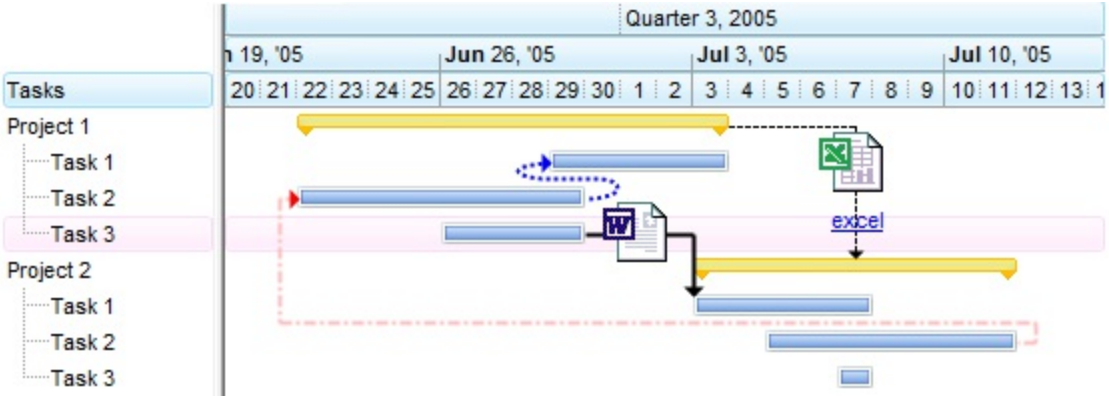
Currently, the single read-only property that supports pattern for the LinkKey parameter is **exLinksCount**, which counts the links as follows:

- **Link("<*>",exLinksCount)** counts all links in the chart.
- **Link(0,"<pattern>",exLinksCount)** counts all links in the chart that match a specified pattern using wild characters as *,?,# or []

The pattern may include the space character which indicates multiple patterns to be used when matching. For instance "A* *K" indicates all keys that start on A and all keys that end on K.

The /NET Assembly version defines get/set shortcut properties as follow (they start with get_ or set_ keywords):

- **LinkStartItem** : Integer, retrieves or sets a value that indicates the handle of the item where the link start
- **LinkStartBar** : Object, retrieves or sets a value that indicates the key of the bar where the link starts
- **LinkEndItem** : Integer, retrieves or sets a value that indicates the handle of the item where the link ends
- **LinkEndBar** : Object, retrieves or sets a value that indicates the key of the bar where the link ends
- **LinkVisible** : Boolean, specifies whether the link is visible or hidden
- **LinkUserData** : Object, specifies an extra data associated with the link
- **LinkStartPos** : AlignmentEnum, specifies the position where the link starts in the source item
- **LinkEndPos** : AlignmentEnum, specifies the position where the link ends in the target item
- **LinkColor** : Color, specifies the color to paint the link
- **LinkArrowColor** : Color, specifies the color to paint the arrow of the link
- **LinkArrowColor32** : Color, specifies the color to paint the arrow of the link
- **LinkStyle** : LinkStyleEnum, specifies the style to paint the link
- **LinkWidth** : Integer, specifies the width in pixels of the link
- **LinkShowDir** : Boolean, specifies whether the link shows the direction
- **LinkShowRound** : Boolean, specifies whether the link is round or rectangular
- **LinkText** : String, specifies the HTML text being displayed on the link
- **LinkToolTip** : String, specifies the HTML text being shown when the cursor hovers the link
- **LinkSelected** : Boolean, specifies whether the link is selected or unselected
- **LinkGroupBars** : GroupBarsOptionsEnum, groups or ungroups the bars being linked with the specified options
- **LinksCount** : Integer, specifies the number of the links within the chart



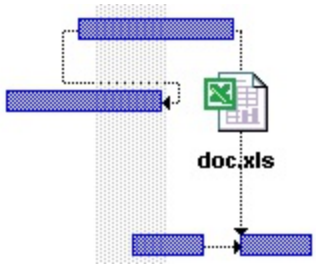
So instead using the get_Link or set_Link properties you can use these functions.

For instance, the following VB/NET sample changes the link's color:

```
With Exg2antt1.Items
    .set_LinkColor("L1", Color.Red)
End With
```

For instance, the following C# sample changes the link's color:

```
exg2antt1.Items.set_LinkColor("L1", Color.Red);
```



The following VB sample displays a text plus a picture on a link:

```
G2antt1.Items.Link("Link", exLinkText) = " <img>excel</img><br><br><b>doc.xls"
```

property Items.LockedItem (Alignment as VAlignmentEnum, Index as Long) as HITEM

Retrieves the handle of the locked item.

Type	Description
Alignment as VAlignmentEnum	A VAlignmentEnum expression that indicates whether the locked item requested is on the top or bottom side of the control.
Index as Long	A long expression that indicates the position of item being requested.
HITEM	A long expression that indicates the handle of the locked item

A locked or fixed item is always displayed on the top or bottom side of the control no matter if the control's list is scrolled up or down. Use the LockedItem property to access a locked item by its position. Use the LockedItemCount property to add or remove items fixed/locked to the top or bottom side of the control. Use the ShowLockedItems property to show or hide the locked items. Use the IsItemLocked property to check whether an item is locked or unlocked. Use the CellValue property to specify the caption for a cell. Use the InsertControlItem property to assign an ActiveX control to a locked item only

The following VB sample adds an item that's locked to the top side of the control:

```
With G2antt1
   Dim a As EXG2ANTTLibCtl.VAlignmentEnum
   a = EXG2ANTTLibCtl.VAlignmentEnum.TopAlignment
   .BeginUpdate
   With .Items
      .LockedItemCount(a) = 1
      Dim h As EXG2ANTTLibCtl.HITEM
      h = .LockedItem(a, 0)
      .CellValue(h, 0) = "<b>locked</b> item"
      .CellValueFormat(h, 0) = exHTML
   End With
   .EndUpdate
End With
```

The following C++ sample adds an item that's locked to the top side of the control:

```
#include "Items.h"
m_g2antt.BeginUpdate();
CItems items = m_g2antt.GetItems();
items.SetLockedItemCount( 0 /*TopAlignment*/, 1);
long i = items.GetLockedItem( 0 /*TopAlignment*/, 0 );
COleVariant vtItem(i), vtColumn( long(0) );
items.SetCellValue( vtItem, vtColumn, COleVariant( "<b>locked</b> item" ) );
items.SetCellValueFormat( vtItem, vtColumn, 1/*exHTML*/ );
m_g2antt.EndUpdate();
```

The following VB.NET sample adds an item that's locked to the top side of the control:

```
With AxG2antt1
   .BeginUpdate()
   With .Items
      .LockedItemCount(EXG2ANTTLib.VAlignmentEnum.TopAlignment) = 1
      Dim i As Integer
      i = .LockedItem(EXG2ANTTLib.VAlignmentEnum.TopAlignment, 0)
      .CellValue(i, 0) = "<b>locked</b> item"
      .CellValueFormat(i, 0) = EXG2ANTTLib.CaptionFormatEnum.exHTML
   End With
   .EndUpdate()
End With
```

The following C# sample adds an item that's locked to the top side of the control:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Items items = axG2antt1.Items;
items.set_LockedItemCount(EXG2ANTTLib.VAlignmentEnum.TopAlignment, 1);
int i = items.get_LockedItem(EXG2ANTTLib.VAlignmentEnum.TopAlignment, 0);
items.set_CellValue(i, 0, "<b>locked</b> item");
items.set_CellValueFormat(i, 0, EXG2ANTTLib.CaptionFormatEnum.exHTML);
axG2antt1.EndUpdate();
```

The following VFP sample adds an item that's locked to the top side of the control:

```
with thisform.G2antt1
   .BeginUpdate()
```

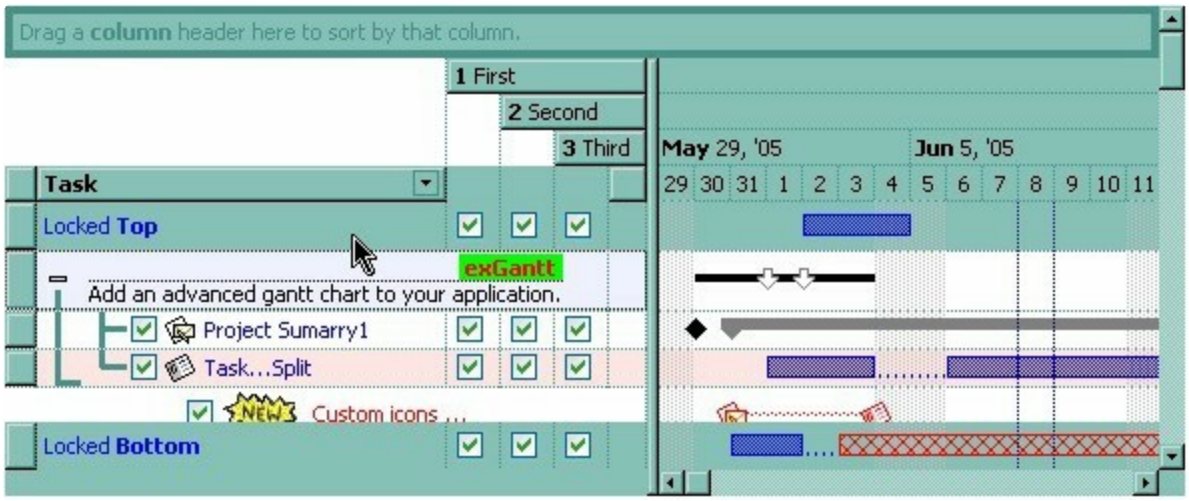
```
    With .Items
      .LockedItemCount(0) = 1
      .DefaultItem = .LockedItem(0, 0)
      .CellValue(0, 0) = "<b>locked</b> item"
      .CellValueFormat(0, 0) = 1 && EXG2ANTTLib.CaptionFormatEnum.exHTML
    EndWith
  .EndUpdate()
endwith
```

property Items.LockedItemCount(Alignment as VAlignmentEnum) as Long

Specifies the number of items fixed on the top or bottom side of the control.

Type	Description
Alignment as VAlignmentEnum	A VAlignmentEnum expression that specifies the top or bottom side of the control.
Long	A long expression that indicates the number of items locked to the top or bottom side of the control.

A locked or fixed item is always displayed on the top or bottom side of the control no matter if the control's list is scrolled up or down. Use the LockedItemCount property to add or remove items fixed/locked to the top or bottom side of the control. Use the LockedItem property to access a locked item by its position. Use the ShowLockedItems property to show or hide the locked items. Use the CellValue property to specify the caption for a cell. Use the CountLockedColumns property to lock or unlock columns in the control. Use the ItemBackColor property to specify the item's background color. Use the ItemDivider property to merge the cells. Use the MergeCells method to combine two or multiple cells in a single cell.



The following VB sample adds two items that are locked to the top side of the control, and one item that's locked to the bottom side of the control:

```
With G2antt1
  Dim h As EXG2ANTTLibCtl.HITEM
  Dim a As EXG2ANTTLibCtl.VAlignmentEnum
  a = EXG2ANTTLibCtl.VAlignmentEnum.TopAlignment
  .BeginUpdate
  With .Items
    .LockedItemCount(a) = 2
```

```
    For i = 0 To .LockedItemCount(a) - 1
      h = .LockedItem(a, i)
      .CellValue(h, 0) = "item <b>locked</b> to the top side of the control"
      .CellValueFormat(h, 0) = exHTML
      .ItemBackColor(h) = SystemColorConstants.vb3DFace
      .ItemForeColor(h) = SystemColorConstants.vbWindowText
    Next
    a = EXG2ANTTLibCtl.VAlignmentEnum.BottomAlignment
    .LockedItemCount(a) = 1
    h = .LockedItem(a, 0)
    .CellValue(h, 0) = "item <b>locked</b> to the bottom side of the control"
    .CellValueFormat(h, 0) = exHTML
    .ItemBackColor(h) = SystemColorConstants.vb3DFace
  End With
  .EndUpdate
End With
```

The following C++ sample adds an item that's locked to the top side of the control:

```
#include "Items.h"
m_g2antt.BeginUpdate();
CItems items = m_g2antt.GetItems();
items.SetLockedItemCount( 0 /*TopAlignment*/, 1);
long i = items.GetLockedItem( 0 /*TopAlignment*/, 0 );
COleVariant vtItem(i), vtColumn( long(0) );
items.SetCellValue( vtItem, vtColumn, COleVariant( "<b>locked</b> item" ) );
items.SetCellValueFormat( vtItem, vtColumn, 1/*exHTML*/ );
m_g2antt.EndUpdate();
```

The following VB.NET sample adds an item that's locked to the top side of the control:

```
With AxG2antt1
  .BeginUpdate()
  With .Items
    .LockedItemCount(EXG2ANTTLib.VAlignmentEnum.TopAlignment) = 1
    Dim i As Integer
    i = .LockedItem(EXG2ANTTLib.VAlignmentEnum.TopAlignment, 0)
    .CellValue(i, 0) = "<b>locked</b> item"
```

```
    .CellValueFormat(i, 0) = EXG2ANTTLib.CaptionFormatEnum.exHTML
  End With
  .EndUpdate()
End With
```

The following C# sample adds an item that's locked to the top side of the control:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Items items = axG2antt1.Items;
items.set_LockedItemCount(EXG2ANTTLib.VAlignmentEnum.TopAlignment, 1);
int i = items.get_LockedItem(EXG2ANTTLib.VAlignmentEnum.TopAlignment, 0);
items.set_CellValue(i, 0, "<b>locked</b> item");
items.set_CellValueFormat(i, 0, EXG2ANTTLib.CaptionFormatEnum.exHTML);
axG2antt1.EndUpdate();
```

The following VFP sample adds an item that's locked to the top side of the control:

```
with thisform.G2antt1
  .BeginUpdate()
    With .Items
      .LockedItemCount(0) = 1
      .DefaultItem = .LockedItem(0, 0)
      .CellValue(0, 0) = "<b>locked</b> item"
      .CellValueFormat(0, 0) = 1 && EXG2ANTTLib.CaptionFormatEnum.exHTML
    EndWith
  .EndUpdate()
endwith
```

property Items.MatchItemCount as Long

Retrieves the number of items that match the filter.

Type	Description
Long	A long expression that specifies the number of matching items in the control. The value could be a positive value if no filter is applied, or negative while filter is on.

The MatchItemCount property counts the number of items that matches the current filter criteria. At runtime, the MatchItemCount property is a positive integer if no filter is applied, and negative if a filter is applied. If positive, it indicates the number of items within the control (ItemCount property). If negative, a filter is applied, and the absolute value minus one, indicates the number of matching items after filter is applied. A matching item includes its parent items, if the control's FilterInclude property allows including child items.

The MatchItemCount property returns a value as explained bellow:

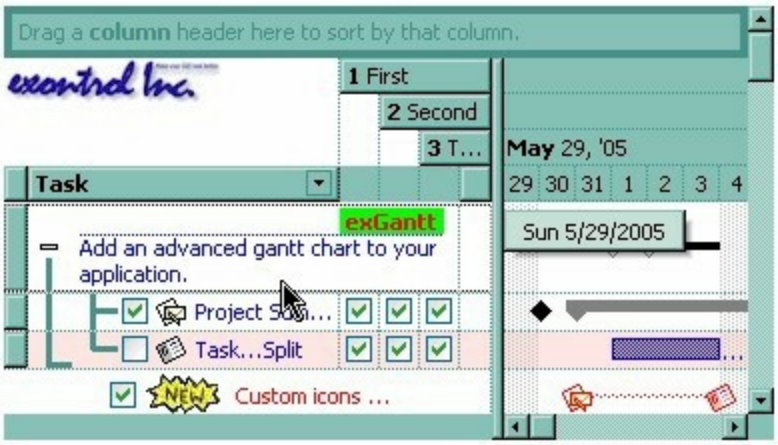
- 0, the control displays/contains no items, and no filter is applied to any column
- -1, the control displays no items, and there is a filter applied (no match found)
- positive number, indicates the number of items within the control (ItemCount property)
- negative number, the absolute value minus 1, indicates the number of items that matches the current filter (match found)

method Items.MergeCells ([Cell1 as Variant], [Cell2 as Variant], [Options as Variant])

Merges a list of cells.

Type	Description
Cell1 as Variant	A long expression that indicates the handle of the cell being merged, or a safe array that holds a collection of handles for the cells being merged. Use the ItemCell property to retrieves the handle of the cell. The **first cell (in the list, if exists)** specifies the cell being displayed in the new larger cell.
Cell2 as Variant	A long expression that indicates the handle of the cell being merged, or a safe array that holds a collection of handles for the cells being merged. Use the ItemCell property to retrieves the handle of the cell. The first cell in the list specifies the cell being displayed in the new larger cell.
Options as Variant	Reserved.

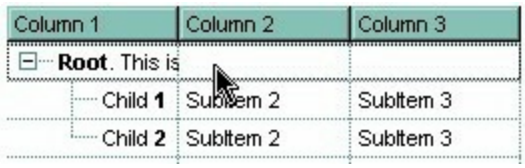
The MergeCells method combines two or more cells into one cell. The data in the **first specified cell** is displayed in the new larger cell. All the other cells' data is not lost. Use the CellMerge property to merge or unmerge a cell with another cell in the same item. Use the ItemDivider property to display a single cell in the entire item (merging all cells in the item). Use the UnmergeCells method to unmerge the merged cells. Use the CellValue property to specify the cell's caption. Use the ItemCell property to retrieves the handle of the cell. Use the BeginMethod and EndUpdate methods to maintain performance, when merging multiple cells in the same time. The MergeCells methods creates a list of cells from Cell1 and Cell2 parameters that need to be merged, and the first cell in the list specifies the displayed cell in the merged cell. Use the SplitCell property to split a cell. Use the SelectableItem property to specify the user can select an item.



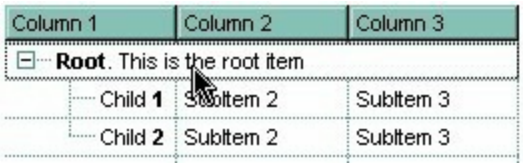
The following VB sample adds three columns, a root item and two child items:

```
With G2antt1
   .BeginUpdate
      .MarkSearchColumn = False
      .DrawGridLines = exAllLines
      .LinesAtRoot = exLinesAtRoot
      With .Columns.Add("Column 1")
         .Def(exCellValueFormat) = exHTML
      End With
      .Columns.Add "Column 2"
      .Columns.Add "Column 3"
      With .Items
         Dim h As Long
         h = .AddItem("Root. This is the root item")
         .InsertItem h, , Array("Child 1", "SubItem 2", "SubItem 3")
         .InsertItem h, , Array("Child 2", "SubItem 2", "SubItem 3")
         .ExpandItem(h) = True
         .SelectItem(h) = True
      End With
   .EndUpdate
End With
```

and it looks like follows (notice that the caption of the root item is truncated by the column that belongs to):



If we are merging the first three cells in the root item we get:



You can merge the first three cells in the root item using any of the following methods:

```
With G2antt1
   With .Items
```

```
    .CellMerge(.RootItem(0), 0) = Array(1, 2)
  End With
End With
```

```
With G2antt1
  .BeginUpdate
  With .Items
    Dim r As Long
    r = .RootItem(0)
    .CellMerge(r, 0) = 1
    .CellMerge(r, 0) = 2
  End With
  .EndUpdate
End With
```

```
With G2antt1
  .BeginUpdate
  With .Items
    Dim r As Long
    r = .RootItem(0)
    .MergeCells .ItemCell(r, 0), .ItemCell(r, 1)
    .MergeCells .ItemCell(r, 0), .ItemCell(r, 2)
  End With
  .EndUpdate
End With
```

```
With G2antt1
  With .Items
    Dim r As Long
    r = .RootItem(0)
    .MergeCells .ItemCell(r, 0), Array(.ItemCell(r, 1), .ItemCell(r, 2))
  End With
End With
```

```
With G2antt1
  With .Items
    Dim r As Long
```

```
    r = .RootItem(0)
    .MergeCells Array(.ItemCell(r, 0), .ItemCell(r, 1), .ItemCell(r, 2))
  End With
End With
```

The following VB sample merges the first three cells:

```
With G2antt1.Items
  .MergeCells .ItemCell(.FocusItem, 0), Array(.ItemCell(.FocusItem, 1), .ItemCell(.FocusItem,
2))
End With
```

The following C++ sample merges the first three cells:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
COleVariant vtFocusCell( items.GetItemCell(items.GetFocusItem(), COleVariant( (long)0 ) ) ),
vtMissing; V_VT( &vtMissing ) = VT_ERROR;
items.MergeCells( vtFocusCell, COleVariant( items.GetItemCell(items.GetFocusItem(),
COleVariant( (long)1 ) ) ), vtMissing );
items.MergeCells( vtFocusCell, COleVariant( items.GetItemCell(items.GetFocusItem(),
COleVariant( (long)2 ) ) ), vtMissing );
```

The following VB.NET sample merges the first three cells:

```
With AxG2antt1.Items
  .MergeCells(.ItemCell(.FocusItem, 0), .ItemCell(.FocusItem, 1))
  .MergeCells(.ItemCell(.FocusItem, 0), .ItemCell(.FocusItem, 2))
End With
```

The following C# sample merges the first three cells:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
items.MergeCells(items.get_ItemCell( items.FocusItem, 0 ), items.get_ItemCell(
items.FocusItem, 1 ),"");
items.MergeCells(items.get_ItemCell(items.FocusItem, 0),
items.get_ItemCell(items.FocusItem, 2),"");
```

The following VFP sample merges the first three cells:

```
with thisform.G2antt1.Items
    .MergeCells(.ItemCell(.FocusItem,0), .ItemCell(.FocusItem,1), "")
    .MergeCells(.ItemCell(.FocusItem,0), .ItemCell(.FocusItem,2), "")
endwith
```

Now, the question is what should I use in my program in order to merge some cells? For instance, if you are using handle to cells (HCELL type), we would recommend using the MergeCells method, else you could use as well the CellMerge property.

property Items.NextItemBar (Item as HITEM, Key as Variant) as Variant

Gets the key of the next bar in the item.

Type	Description
Item as HITEM	A HITEM expression that indicates the handle of the item where the bars are enumerated.
Key as Variant	A String expression that indicates the key of the bar.
Variant	A String expression that indicates the key of the next bar in the item, or empty if there is no next bar in the item

Use the FirstItemBar and NextItemBar methods to enumerate the bars inside the item. Use the ItemBar property to access properties of the specified bar. Use the AddBar method to add new bars to the item. Use the AddLink method to link a bar with another. Use the AllowCreateBar method to create new bars using the mouse. Use the RemoveBar method to remove a bar from an item. Use the ClearBars method to remove all bars in the item. The FirstItemBar and NextItemBar methods enumerates bars in alphabetic order of the keys.

The following VB.NET sample enumerates all items and bars in the control (/NET or /WPF version):

```
With Exg2antt1
   Dim i, h As Integer, key As Object
   For i = 0 To .Items.ItemCount - 1
      h = .Items(i)
      key = .Items.get_FirstItemBar(h)
      While TypeOf key Is String
         Debug.Print("Key = " & key & ", Item " & .Items.get_CellCaption(h, 0))
         key = CStr(.Items.get_NextItemBar(h, key))
      End While
   Next
End With
```

The following C# sample enumerates all items and bars in the control (/NET or /WPF version):

```
for (int i = 0; i < exg2antt1.Items.ItemCount; i++)
{
   int h = exg2antt1.Items[i];
```

```
    object key = exg2antt1.Items.get_FirstItemBar(h);
    while (key != null)
    {
        System.Diagnostics.Debug.Print("Key = " + key + ", Item " +
exg2antt1.Items.get_CellCaption(h, 0));
        key = exg2antt1.Items.get_NextItemBar(h, key);
    }
}
```

The following VB sample enumerates the bars in the item (h indicates the handle of the item):

```
With G2antt1
    If Not (h = 0) Then
        Dim k As Variant
        k = .Items.FirstItemBar(h)
        While Not IsEmpty(k)
            Debug.Print "Key = " & k
            k = .Items.NextItemBar(h, k)
        Wend
    End If
End With
```

The following C++ sample enumerates the bars in the item (h indicates the handle of the item):

```
CItems items = m_g2antt.GetItems();
COleVariant vtBar = items.GetFirstItemBar(h) ;
while ( V_VT( &vtBar ) != VT_EMPTY )
{
    OutputDebugString( V2S( &vtBar ) );
    OutputDebugString( "\n" );
    vtBar = items.GetNextItemBar( h, vtBar );
}
```

where the V2S function converts a Variant expression to a string:

```
static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
```

```
    if ( pv )
    {
        if ( pv->vt == VT_ERROR )
            return szDefault;

        COleVariant vt;
        vt.ChangeType( VT_BSTR, pv );
        return V_BSTR( &vt );
    }
    return szDefault;
}
```

The following VB.NET sample enumerates the bars in the item (h indicates the handle of the item):

```
With AxG2antt1
    If Not (h = 0) Then
        Dim k As Object
        k = .Items.FirstItemBar(h)
        While TypeOf k Is String
            System.Diagnostics.Debug.Print(k.ToString)
            k = .Items.NextItemBar(h, k)
        End While
    End If
End With
```

The following C# sample enumerates the bars in the item (h indicates the handle of the item):

```
object k = axG2antt1.Items.get_FirstItemBar(h);
while ( k != null )
{
    System.Diagnostics.Debug.Print(k.ToString());
    k = axG2antt1.Items.get_NextItemBar(h, k);
}
```

The following VFP sample enumerates the bars in the item (h indicates the handle of the item):

```
With thisform.G2antt1
   If Not (h = 0) Then
      local k
      k = .Items.FirstItemBar(h)
      do While !empty(k)
         ?k
         k = .Items.NextItemBar(h, k)
      enddo
   Endif
EndWith
```

In VFP, please make sure that you are using non empty values for the keys. For instance, if you are omitting the Key parameter of the AddBar method, an empty key is missing. If you need to use the FirstItemBar and NextItemBar properties, you have to use non empty keys for the bars.

property Items.NextLink (LinkKey as Variant) as Variant

Gets the key of the next link.

Type	Description
LinkKey as Variant	A string expression that indicates the key of the previous link
Variant	A string expression that indicates the key of the next link, or empty value if there is no next link.

Use the FirstLink and NextLink properties to enumerate the links in the control. The NextLink property retrieves an empty value, if there is no next link in the control. Use the AddLink property to link two bars. Use the ShowLinks property to show or hide the links. Use the Link property to access a property of the link.

The following VB sample enumerates the links:

```
With G2antt1.Items
   Dim k As Variant
   k = .FirstLink()
   While Not IsEmpty(k)
      Debug.Print "LinkKey = " & k
      k = .NextLink(k)
   Wend
End With
```

The following C++ sample enumerates the links:

```
CItems items = m_g2antt.GetItems();
COleVariant vtLinkKey = items.GetFirstLink() ;
while ( V_VT( &vtLinkKey ) != VT_EMPTY )
{
   OutputDebugString( V2S( &vtLinkKey ) );
   OutputDebugString( "\n" );
   vtLinkKey = items.GetNextLink( vtLinkKey );
}
```

where the V2S function converts a Variant expression to a string:

```
static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
```

```
{
    if ( pv )
    {
        if ( pv->vt == VT_ERROR )
            return szDefault;

        COleVariant vt;
        vt.ChangeType( VT_BSTR, pv );
        return V_BSTR( &vt );
    }
    return szDefault;
}
```

The following VB.NET sample enumerates the links:

```
With AxG2antt1.Items
    Dim k As Object
    k = .FirstLink
    While (TypeOf k Is String)
        System.Diagnostics.Debug.Print(k.ToString)
        k = .NextLink(k)
    End While
End With
```

The following C# sample enumerates the links:

```
object k = axG2antt1.Items.FirstLink;
while (k != null)
{
    System.Diagnostics.Debug.Print(k.ToString());
    k = axG2antt1.Items.get_NextLink(k);
}
```

The following VFP sample enumerates the links:

```
With thisform.G2antt1.Items
    local k
    k = .FirstLink
    do While !empty(k)
```

```
    ?k
    k = .NextLink(k)
  enddo
endwith
```

property Items.NextSiblingItem (Item as HITEM) as HITEM

Retrieves the next sibling of the item in the parent's child list.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
HITEM	A long expression that indicates the handle of the next sibling item.

The NextSiblingItem property retrieves the next sibling of the item in the parent's child list. Use ItemChild and NextSiblingItem properties to enumerate the collection of child items.

The following VB function recursively enumerates the item and all its child items:

```
Sub RecItem(ByVal c As EXG2ANTTLibCtl.G2antt, ByVal h As HITEM)
  If Not (h = 0) Then
    Dim hChild As HITEM
    With c.Items
      Debug.Print .CellValue(h, 0)
      hChild = .ItemChild(h)
      While Not (hChild = 0)
        RecItem c, hChild
        hChild = .NextSiblingItem(hChild)
      Wend
    End With
  End If
End Sub
```

The following C++ function recursively enumerates the item and all its child items:

```
void RecItem( CG2antt* pG2antt, long hItem )
{
  COleVariant vtColumn( (long)0 );
  if ( hItem )
  {
    CItems items = pG2antt->GetItems();

    CString strCaption = V2S( &items.GetCellValue( COleVariant( hItem ), vtColumn ) ),
strOutput;
```

```
    strOutput.Format( "Cell: '%s'\n", strCaption );
    OutputDebugString( strOutput );

    long hChild = items.GetItemChild( hItem );
    while ( hChild )
    {
      RecItem( pG2antt, hChild );
      hChild = items.GetNextSiblingItem( hChild );
    }
  }
}
```

The following VB.NET function recursively enumerates the item and all its child items:

```
Shared Sub RecItem(ByVal c As AxEXG2ANTTLib.AxG2antt, ByVal h As Integer)
  If Not (h = 0) Then
    Dim hChild As Integer
    With c.Items
      Debug.WriteLine(.CellValue(h, 0))
      hChild = .ItemChild(h)
      While Not (hChild = 0)
        RecItem(c, hChild)
        hChild = .NextSiblingItem(hChild)
      End While
    End With
  End If
End Sub
```

The following C# function recursively enumerates the item and all its child items:

```
internal void RecItem(AxEXG2ANTTLib.AxG2antt  g2antt, int hItem)
{
  if (hItem != 0)
  {
    EXG2ANTTLib.Items items = g2antt.Items;
    object caption = items.get_CellValue( hItem, 0 );
    System.Diagnostics.Debug.WriteLine(caption != null ? caption.ToString() : "");
```

```
        int hChild = items.get_ItemChild(hItem);
        while (hChild != 0)
        {
            RecItem(g2antt, hChild);
            hChild = items.get_NextSiblingItem(hChild);
        }
    }
}
```

The following VFP function recursively enumerates the item and all its child items (recitem method):

```
LPARAMETERS h

with thisform.G2antt1
    If ( h != 0 ) Then
        local hChild
        With .Items
            .DefaultItem = h
            wait window .CellValue(0, 0)
            hChild = .ItemChild(h)
            do While (hChild != 0)
                thisform.recitem(hChild)
                hChild = .NextSiblingItem(hChild)
            enddo
        EndWith
    EndIf
endwith
```

property Items.NextVisibleItem (Item as HITEM) as HITEM

Retrieves the handle of next visible item.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
HITEM	A long expression that indicates the handle of the next visible item.

Use the NextVisibleItem property to access the visible items. The NextVisibleItem property retrieves 0 if there are no more visible items. Use the IsItemVisible property to check whether an item fits the control's client area. Use the FirstVisibleItem property to retrieve the first visible item.

The following VB sample enumerates all visible items:

```
Private Sub VisItems(ByVal c As EXG2ANTTLibCtl.G2antt)
  Dim h As HITEM
  With c.Items
    h = .FirstVisibleItem
    While Not (h = 0)
      Debug.Print .CellValue(h, 0)
      h = .NextVisibleItem(h)
    Wend
  End With
End Sub
```

The FormatColumn event is fired before displaying a cell, so you can handle the FormatColumn to display anything on the cell at runtime. This way you can display the row position, you can display the value using the currency format, and so on. The FireFormatColumn property allows the control to fire the FormatColumn event for the column. The Position property specifies the position of the column.

- If your chart does *not* display a tree or a hierarchy this property is ok to be used with FormatColumn event to display the position

The following VB sample handles the FormatColumn event to display the row position:

```
Private Sub G2antt1_FormatColumn(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal
ColIndex As Long, Value As Variant)
  Value = G2antt1.Items.ItemPosition(Item)
```

```
End Sub
```

- If your chart displays a tree or a hierarchy the position of the item must be determined relative to the FirstVisibleItem as shown in the following VB sample:

```
Private Sub G2antt1_FormatColumn(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal
ColIndex As Long, Value As Variant)
   Value = G2antt1.ScrollPos(True) + RelPos(Item)
End Sub

Private Function RelPos(ByVal hVisible As Long) As Long
   With G2antt1.Items
      Dim h As Long, i As Long, n As Long
      i = 0
      n = .VisibleCount + 1
      h = .FirstVisibleItem
      While (i <= n) And h <> 0 And h <> hVisible
         i = i + 1
         h = .NextVisibleItem(h)
      Wend
      RelPos = i
   End With
End Function
```

The following C++ sample enumerates all visible items:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
long hItem = items.GetFirstVisibleItem();
while ( hItem )
{
   OutputDebugString( V2S( &items.GetCellValue( COleVariant( hItem ), COleVariant(
long(0) ) ) ) );
   hItem = items.GetNextVisibleItem( hItem );
}
```

The following C# sample enumerates all visible items:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
```

```
int hItem = items.FirstVisibleItem;
while ( hItem != 0 )
{
   object strCaption = items.get_CellValue(hItem, 0);
   System.Diagnostics.Debug.WriteLine( strCaption != null ? strCaption.ToString() : "" );
   hItem = items.get_NextVisibleItem(hItem);
}
```

The following VB.NET sample enumerates all visible items:

```
With AxG2antt1.Items
   Dim hItem As Integer
   hItem = .FirstVisibleItem
   While Not (hItem = 0)
      Debug.Print(.CellValue(hItem, 0))
      hItem = .NextVisibleItem(hItem)
   End While
End With
```

The following VFP sample enumerates all visible items:

```
with thisform.G2antt1.Items
   .DefaultItem = .FirstVisibleItem
   do while ( .DefaultItem <> 0 )
      wait window .CellValue( 0, 0 )
      .DefaultItem = .NextVisibleItem( 0 )
   enddo
endwith
```

property Items.PathSeparator as String

Returns or sets the delimiter character used for the path returned by the FullPath and FindPath properties.

Type	Description
String	A string expression that indicates the delimiter character used for the path returned by the FullPath and FindPath properties.

By default the PathSeparator is "\". The PathSeparator property is used by properties like FullPath and FindPath.

property Items.PrevSiblingItem (Item as HITEM) as HITEM

Retrieves the previous sibling of the item in the parent's child list.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
HITEM	A long expression that indicates the handle of the previous sibling item

The PrevSiblingItem retrieves 0 if there are no more previous sibling items. The NextSiblingItem property retrieves the next sibling of the item in the parent's child list. Use the FirstVisibleItem property to retrieve the first visible item. Use the ItemParent property to retrieve the parent of the item.

property Items.PrevVisibleItem (Item as HITEM) as HITEM

Retrieves the handle of previous visible item.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
HITEM	A long expression that indicates the handle of the previous visible item

The PrevVisibleItem property retrieves 0 if there are no previous visible items. The NextVisibleItem property retrieves the next visible item. Use the FirstVisibleItem property to retrieve the first visible item. Use the ItemParent property to retrieve the parent of the item.

method Items.RemoveAllItems ()

Removes all items from the control.

Type	Description

Use the RemoveAllItems method to remove all items in the control. Use the Clear method to remove all columns in the control. Use the RemoveItem method to remove a single item in the control.

method Items.RemoveBar (Item as HITEM, [Key as Variant])

Removes a bar from an item.

Type	Description
Item as HITEM	A long expression that indicates the the handle of the item where the bar is removed. If the Item parameter is 0, the RemoveBar method removes all bars with specified key from all items. In this case the DefaultItem (/COM only) property should be 0 (by default), else it refers a single item being indicated by the DefaultItem property.
Key as Variant	A String expression that indicates the key of the bar to be removed. If missing, the Key parameter is empty. The Key may include a pattern with wild characters as *,?,# or [], if the Key starts with "<" and ends on ">" aka "<K*>" which indicates all bars with the key K or starts on K. The pattern may include a space which divides multiple patterns for matching. For instance "<A* *K>" indicates all keys that start on A and all keys that end on K.

Use the RemoveBar method to remove a bar from an item. *If the Item parameter is not 0 (indicates a valid handle), the RemoveBar removes a single bar (if found, with the Key being specified by the Key parameter). If the Item parameter is 0, the RemoveBar method removes all bars with specified key from all items.* Use the BeginUpdate / EndUpdate methods to refresh the control's content after removing a bar or several bars. Use the ClearBars method to remove all bars in the item.

Based on the values of Item and Key parameters the RemoveBar property remove none, one or multiple bars as follow:

- **RemoveBar(0,"<*>")** removes all bars in the chart
- **RemoveBar(0,"<pattern>")** removes all bars in the chart that match a specified pattern using wild characters as *,?,# or []
- **RemoveBar(Item,"<*>")** removes all bars in the specified Item
- **RemoveBar(Item,"<pattern>")** removes all bars from the giving Item that match a specified pattern using wild characters as *,?,# or []

The pattern may include the space character which indicates multiple patterns to be used when matching. For instance "A* *K" indicates all keys that start on A and all keys that end on K.

Here's few samples of using the RemoveBar method:

- *RemoveBar(Item,"K1") removes the bar K1 from the specified Item*
- *RemoveBar(0,"K1") removes the bar K1 from the entire chart*
- *RemoveBar(0,"<A* K*>") removes all bars from the chart with the Key A or K or starts with A or K*
- *RemoveBar(0,"<*K>") removes all bars from the chart with the Key K or ends on K*
- *RemoveBar(Item,"<K*>") removes all bars from the specified Item with the Key K or starts on K*
- *RemoveBar(Item,"<K??>") removes all bars from the specified Item with the Key of 3 characters and starts with K*

Use the AddBar method to add new bars to the item. Use the Remove method to remove a type of bar from the Bars collection. Use the Add method to add new types of bars to the Bars collection. Use the FirstVisibleDate property to specify the first visible date in the chart area. Use the Key parameter to identify a bar inside an item. Use the ItemBar property to access a bar inside the item. Use the PaneWidth property to specify the width of the chart. Use the NonworkingDays property to specify the non-working days. The RemoveBar method removes the links related to bar. Use the RemoveSelection method to remove the objects (bars, links) in the chart's selection. Use the ItemBar(exBarSelected) property to select or unselect programmatically a bar.

Use the ItemBar(exBarItemParent) property to move a bar from an item to another item. The ItemBar(exBarItemParent) property indicates the handle of the item that displays the bar. For instance, a bar can be moved from an item to another, only if in the second item there is no another bar with the same key (ItemBar(exBarKey) property), as an item can contains two bars with the same key. The control fires the BarParentChange event just before moving the bar to another item. Use this event to control the items where your bar can be moved. A bar can be moved to another item, ONLY if the second item does not contain a bar with the same key. The exBarKey property specifies the key of the bar.

method Items.RemoveItem (Item as HITEM)

Removes a specific item.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item being removed.

The RemoveItem method removes an item. The RemoveItem method does not remove the item, if it contains child items. The following sample removes the first item: G2antt1.Items.RemoveItem G2antt1.Items(0). Use the RemoveAllItems method to remove all items in the control. Use the BeginUpdate and EndUpdate methods to maintain performance while removing the items. The RemoveItem method can't remove an item that's locked. Instead you can use the LockedItemCount property to add or remove locked items. Use the IsItemLocked property to check whether an item is locked. The RemoveItem method removes all bars and links related to the item. The RemoveSelection method removes the selected items (including the descendents). The RemoveSelection method removes the selected objects (bars or links) within the chart. The RemoveSelection method removes the selected links/bars from the chart if exists, else it removes the selected items (including the descendents).

The following VB sample removes recursively an item:

```
Private Sub RemoveItemRec(ByVal t As EXG2ANTTLibCtl.G2antt, ByVal h As HITEM)
  If Not h = 0 Then
    With t.Items
      t.BeginUpdate
      Dim hChild As HITEM
      hChild = .ItemChild(h)
      While (hChild <> 0)
        Dim hNext As HITEM
        hNext = .NextSiblingItem(hChild)
        RemoveItemRec t, hChild
        hChild = hNext
      Wend
      .RemoveItem h
      t.EndUpdate
    End With
  End If
End Sub
```

The following C++ sample removes recursively an item:

```
void RemoveItemRec( CG2antt* pG2antt, long hItem )
{
   if ( hItem )
   {
      pG2antt->BeginUpdate();
      CItems items = pG2antt->GetItems();
      long hChild = items.GetItemChild( hItem );
      while ( hChild )
      {
         long nNext = items.GetNextSiblingItem( hChild );
         RemoveItemRec( pG2antt, hChild );
         hChild = nNext;
      }
      items.RemoveItem( hItem );
      pG2antt->EndUpdate();
   }
}
```

The following VB.NET sample removes recursively an item:

```
Shared Sub RemoveItemRec(ByVal t As AxEXG2ANTTLib.AxG2antt, ByVal h As Integer)
   If Not h = 0 Then
      With t.Items
         t.BeginUpdate()
         Dim hChild As Integer = .ItemChild(h)
         While (hChild <> 0)
            Dim hNext As Integer = .NextSiblingItem(hChild)
            RemoveItemRec(t, hChild)
            hChild = hNext
         End While
         .RemoveItem(h)
         t.EndUpdate()
      End With
   End If
End Sub
```

The following C# sample removes recursively an item:

```
internal void RemoveItemRec(AxEXG2ANTTLib.AxG2antt g2antt, int hItem)
{
   if (hItem != 0)
   {
      EXG2ANTTLib.Items items = g2antt.Items;
      g2antt.BeginUpdate();
      int hChild = items.get_ItemChild(hItem);
      while (hChild != 0)
      {
         int hNext = items.get_NextSiblingItem(hChild);
         RemoveItemRec(g2antt, hChild);
         hChild = hNext;
      }
      items.RemoveItem(hItem);
      g2antt.EndUpdate();
   }
}
```

The following VFP sample removes recursively an item (removeitemrec method):

```
LPARAMETERS h

with thisform.G2antt1
   If ( h != 0 ) Then
      .BeginUpdate()
      local hChild
      With .Items
         hChild = .ItemChild(h)
         do While (hChild != 0)
            local hNext
            hNext = .NextSiblingItem(hChild)
            thisform.removeitemrec(hChild)
            hChild = hNext
         enddo
         .RemoveItem( h )
      EndWith
```

```
      .EndUpdate()
   EndIf
endwith
```

method Items.RemoveLink (LinkKey as Variant)

Removes a link.

Type	Description
LinkKey as Variant	A String expression that indicates the key of the link being removed.

Use the RemoveLink method to remove the specified link. Use the Link(exLinkVisible) property to hide a specific link between two bars. Use the AddLink method to add a link between two bars. Use the ClearLinks method to remove all links in the control. Use the ShowLinks property to hide all links in the control. Use the RemoveItem method to remove an item. The RemoveItem method removes all links related to the item. Use the RemoveSelection method to remove the objects (bars, links) in the chart's selection. Use the Link(exLinkSelected) property to select or unselect programmatically a link. Use the Link(exLinkGroupBars) on exGroupBarsNone to ungroup the linked bars. Use the UngroupBars method to ungroup one or two bars. Use the RemoveLinksOf method to remove all links that start or end on the specified bar.

method Items.RemoveLinksOf (Item as HITEM, BarKey as Variant)

Removes the links that goes or ends on the specified bar.

Type	Description
Item as HITEM	A long expression that specifies the handle of the item that hosts the bar whose links are being removed.
BarKey as Variant	A VARIANT expression that specify the key of the bar in the item whose links are removed.

The RemoveLinksOf method removes all links that start or end on the specified bar. If the Item and BarKey does not indicate an existing bar the RemoveLinksOf method has no effect. Use the RemoveLink method to remove the specified link. Use the Link(exLinkGroupBars) on exGroupBarsNone to ungroup the linked bars. Use the UngroupBars method to ungroup one or two bars. Use the ClearLinks method to remove all links in the control. Use the ShowLinks property to hide all links in the control.

method Items.RemoveSelection ()

Removes the selected items (including the descendents).

Type	Description

The RemoveSelection method removes the selected items (including the descendents). The RemoveItem method removes a specific item (in case it includes no descendents). The UnselectAll method unselects all items. The RemoveSelection method removes the selected objects (bars or links) within the chart. The RemoveSelection method removes the selected links/bars from the chart if exists, else it removes the selected items (including the descendents)

property Items.RootCount as Long

Retrieves the number of root objects into Items collection.

Type	Description
Long	A long value that indicates the count of root items in the Items collection.

A root item is an item that has no parent (ItemParent() = 0). Use the RootItem property of the Items object to enumerates the root items. Use the AddItem to add root items to the control. Use the InsertItem method to insert child items.

The following VB sample enumerates all root items:

```
Dim i As Long, n As Long
With G2antt1.Items
   n = .RootCount
   For i = 0 To n - 1
      Debug.Print .CellValue(.RootItem(i), 0)
   Next
End With
```

The following C++ sample enumerates all root items:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
for ( long i = 0 ; i < items.GetRootCount(); i++ )
{
   COleVariant vtItem( items.GetRootItem(i) ), vtColumn( long(0) );
   OutputDebugString( V2S( &items.GetCellValue( vtItem, vtColumn ) ) );
}
```

The following VB.NET sample enumerates all root items:

```
With AxG2antt1.Items
   Dim i As Integer
   For i = 0 To .RootCount - 1
      Debug.Print(.CellValue(.RootItem(i), 0))
   Next
End With
```

The following C# sample enumerates all root items:

```
for (int i = 0; i < axG2antt1.Items.RootCount; i++)
{
    object strCaption = axG2antt1.Items.get_CellValue(axG2antt1.Items.get_RootItem(i), 0);
    System.Diagnostics.Debug.WriteLine(strCaption != null ? strCaption.ToString() : "");
}
```

The following VFP sample enumerates all root items:

```
with thisform.G2antt1.Items
    local i
    for i = 0 to .RootCount - 1
        .DefaultItem = .RootItem(i)
        wait window nowait .CellValue(0,0)
    next
endwith
```

property Items.RootItem ([Position as Long]) as HITEM

Retrieves the handle of the root item giving its index into the root items collection.

Type	Description
Position as Long	A long value that indicates the position of the root item being accessed.
HITEM	A long expression that indicates the handle of the root item.

A root item is an item that has no parent (ItemParent() = 0). Use the RootCount property of to count the root items. Use the AddItem to add root items to the control. Use the InsertItem method to insert child items.

The following VB sample enumerates all root items:

```
Dim i As Long, n As Long
With G2antt1.Items
   n = .RootCount
   For i = 0 To n - 1
      Debug.Print .CellValue(.RootItem(i), 0)
   Next
End With
```

The following C++ sample enumerates all root items:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
for ( long i = 0 ; i < items.GetRootCount(); i++ )
{
   COleVariant vtItem( items.GetRootItem(i) ), vtColumn( long(0) );
   OutputDebugString( V2S( &items.GetCellValue( vtItem, vtColumn ) ) );
}
```

The following VB.NET sample enumerates all root items:

```
With AxG2antt1.Items
   Dim i As Integer
   For i = 0 To .RootCount - 1
      Debug.Print(.CellValue(.RootItem(i), 0))
```

```
    Next
End With
```

The following C# sample enumerates all root items:

```
for (int i = 0; i < axG2antt1.Items.RootCount; i++)
{
    object strCaption = axG2antt1.Items.get_CellValue(axG2antt1.Items.get_RootItem(i), 0);
    System.Diagnostics.Debug.WriteLine(strCaption != null ? strCaption.ToString() : "");
}
```

The following VFP sample enumerates all root items:

```
with thisform.G2antt1.Items
    local i
    for i = 0 to .RootCount - 1
        .DefaultItem = .RootItem(i)
        wait window nowait .CellValue(0,0)
    next
endwith
```

method Items.SchedulePDM (Item as HITEM, Key as Variant)

Schedules the chart using the Precedence Diagram Method.

Type	Description
Item as HITEM	A long expression that specifies the handle of the item where the SchedulePDM starts, or 0, if the Key indicates an unique key of the bar that starts scheduling the SchedulePDM.
Key as Variant	A VARIANT expression that specifies the key of the bar where the SchedulePDM begins. If the Item parameter is 0, the chart looks for the first bar with specified key.

Return	Description
Long	A long expression that specifies whether the operation is successful (0 or any positive value indicating a warning) or failed (negative value). Possible values are • **0**, success • 1, (warning) no bar provided to SchedulePDM method • 2, (warning) single bar in SchedulePDM call • 3, (warning) SchedulePDM method is called during the BarResize event (possible a recursive call/stack overflow) • 4, (warning) no links between scheduled bars • -1, (error) possible cycling • -2, (error) can not move the base bar • -3, (error) scheduling the A and B bars fails • -4, (error) no IN bars • -5, (error) base bar is not initialized • -6, (error) source bar is not initialized (possible cycling) • -7, (error) target bar is not initialized (possible cycling) • -8, (error) no TRANSLATION bar • -9, (error) bar linked to itself. For instance, a bar linked to its summary bar

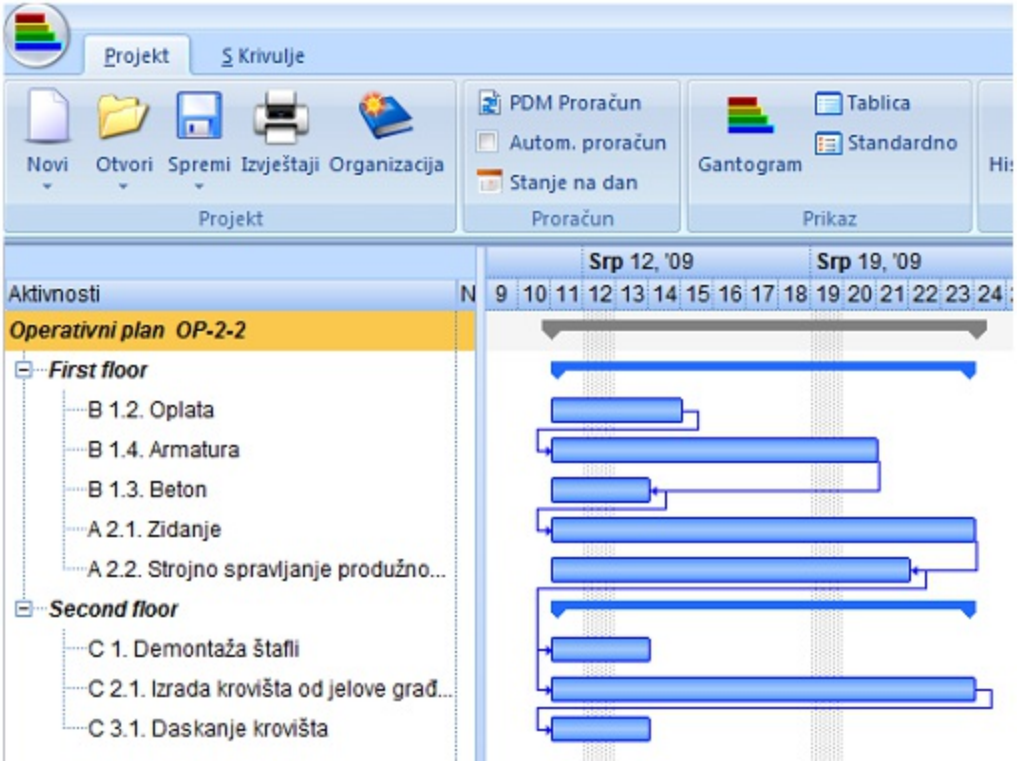
The SchedulePDM method arranges the activities on the plan based on the links / relationships / dependencies. The SchedulePDM calculates early and late dates, based on bar's position, link types and link lag. The SchedulePDM starts from the giving bar, and

continue arranging related bars, unti all related bars are arranged. If a bar has no related bars (no incoming or outgoing links) the procedure still looking for grouped or summary bars, until it finds relative bars.

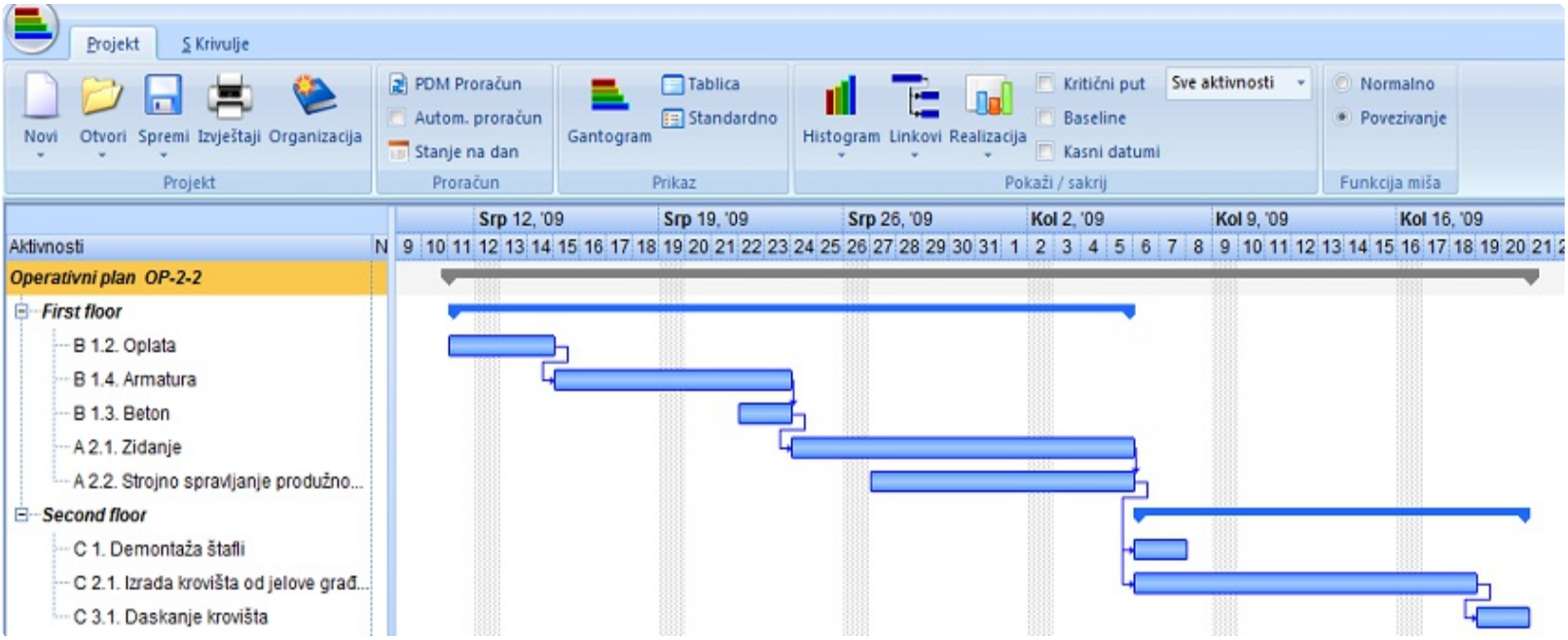
Tasks may have multiple predecessors or multiple successors. Before you begin establishing dependencies, its important to understand that there are four types:

- Finish to Start (FS), the predecessor ends before the successor can begin
- Start to Start (SS), the predecessor begins before the successor can begin
- Finish to Finish (FF), the predecessor ends before the successor can end
- Start to Finish (SF), the predecessor begins before the successor can end

The following screen show shows the chart before calling the SchedulePDM on the Oplata bar:



The following screen show shows the chart after calling the SchedulePDM on the Oplata bar:



The SchedulePDM method handles the following bars:

- simple bars
- not-moveable, not resizable bars: exBarCanMove or exBarCanResize properties.
- bars with margins: exBarMinStart, exBarMaxStart, exBarMinEnd, exBarMaxEnd specifies the margins or the range for the bar.
- working bars: activities that keep constant the working units during moving being indicated by exBarKeepWorkingCount property

- summary bars: DefineSummaryBars property defines a summary bar and child bars.

The type of the link (exLinkType) between two bars is:

- FS (Finish-Start), if the exLinkStartPos is 2(Right) and exLinkEndPos is 0(Left) (by default)
- FF (Finish-Finish), if the exLinkStartPos is 2(Right) and exLinkEndPos is 2(Right)
- SS (Start-Start), if the exLinkStartPos is 0(Left) and exLinkEndPos is 0(Left)
- SF (Start-Finish), if the exLinkStartPos is 0(Left) and exLinkEndPos is 2(Right)

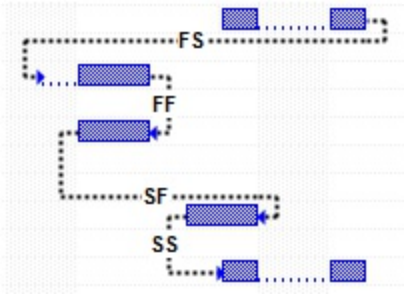
The following properties defines the lag of the link (indicates a delay between two activities):

- exLinkPDMWorkingDelay, specifies that the linked activities are delayed by specified working-units
- exLinkPDMDelay, specifies that the linked activities are delayed by specified units

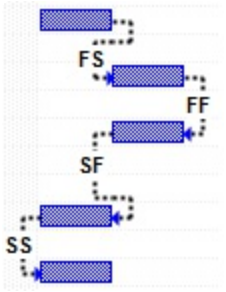
The ChartStartChaning(exPDM) event is fired once the SchedulePDM method is called. The ChartEndChaning(exPDM) event is fired once the SchedulePDM method is called. If Undo/Redo is available, the entire operation is hold as a block, so the chart can be restored by calling the Undo operation, or by pressing the CTRL + Z on chart. You can check the

ChartUndoListAction property to lists the actions being performed during the SchedulePDM method. The DefSchedulePDM property defines options to be used by the SchedulePDM method. If required any option to be used the DefSchedulePDM should be called before the SchedulePDM method else it will have no effect. For instance, use the Def SchedulePDM property to specify a start date for the project, so the SchedulePDM method will use it, to arrange all bars so no bars will start before the specified date. The same if you require to specify the end of the project. The SchedulePDM method invokes the BarResize event for all affected bars.

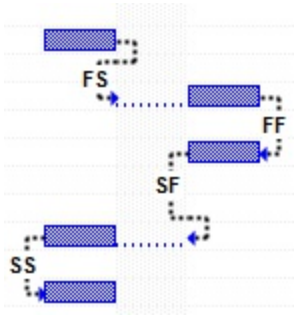
The following screen shot shows the chart using different type of links before calling the SchedulePDM:



The following screen shot shows the chart using different type of links after calling the SchedulePDM:



and if we move the first bar and call again the SchedulePDM we get (the sample preserve the working units for each bar):



The following screen shot shows the activities, when exLinkPDMWorkingDelay property is set for links (SF has 1 working day, FS has 2 working days and the FF has 3 working days delay) :

The following snippet of code, ensures that the SchedulePDM method is not called during the BarResize event: (prevent recursive calls).

```
Dim iSchedulePDM As Long
Private Sub G2antt1_BarResize(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal Key As Variant)
   Debug.Print "BarResize invoked"
   If (iSchedulePDM = 0) Then
      iSchedulePDM = iSchedulePDM + 1
         G2antt1.Items.SchedulePDM Item, Key
      iSchedulePDM = iSchedulePDM - 1
   End If
End Sub
```

The following approach, prevent recursive calls of SchedulePDM method during the BarResize event:

```
iPDMRunning = 0

event ChartStartChaning(Operation)
   if ( Operation == exPDM (12) )
      iPDMRunning++

event ChartEndChaning(Operation)
   if ( Operation == exPDM (12) )
      iPDMRunning--

event BarResize(Item,Key)
   if ( iPDMRunning == 0 )
      Call SchedulePDM(Item,Key)
```

The following VB sample displays a message when the SchedulePDM starts and ends:

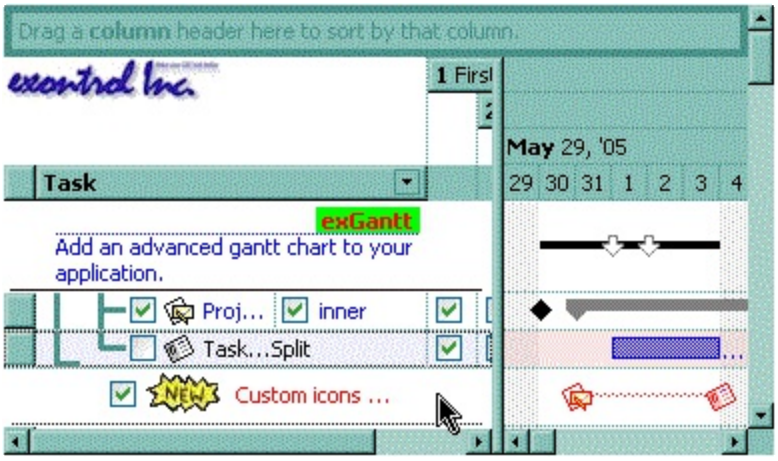
```
Private Sub G2antt1_ChartStartChanging(ByVal Operation As
EXG2ANTTLibCtl.BarOperationEnum)
   If (Operation = exPDM) Then
      Debug.Print "SchedulePDM starts"
   End If
End Sub

Private Sub G2antt1_ChartEndChanging(ByVal Operation As
EXG2ANTTLibCtl.BarOperationEnum)
   If (Operation = exPDM) Then
      Debug.Print "SchedulePDM ends"
   End If
End Sub
```

property Items.SelectableItem(Item as HITEM) as Boolean

Specifies whether the user can select the item.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item being selectable.
Boolean	A boolean expression that specifies whether the item is selectable.

By default, all items are selectable, excepts the locked items that are not selectable. A selectable item is an item that user can select using the keys or the mouse. The SelectableItem property specifies whether the user can select an item. The SelectableItem property doesn't change the item's appearance. The LockedItemCount property specifies the number of locked items to the top or bottom side of the control. Use the ItemDivider property to define a divider item. Use the ItemForeColor property to specify the item's foreground color. Use the ItemBackColor property to specify the item's background color. Use the ItemFont, ItemBold, ItemItalic, ItemUnderline or ItemStrikeOut property to assign a different font to the item. Use the EnableItem property to disable an item. A disabled item looks grayed, but it is selectable. For instance, the user can't change the check box state in a disabled item. Use the SelectItem property to select an item. The ItemFromPoint property gets the item from point. For instance, if the user clicks a non selectable item the SelectionChanged event is not fired. A non selectable item is not focusable as well. It means that if the incremental searching is on, the non selectable items are ignored. Use the SelectCount property to get the number of selected items. Use the SelForeColor and SelBackColor properties to customize the colors for selected items.



The following VB sample makes not selectable the first visible item:

```
With G2antt1.Items
   .SelectableItem(.FirstVisibleItem) = False
End With
```

The following C++ sample makes not selectable the first visible item:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetSelectableItem( items.GetFirstVisibleItem(), FALSE );
```

The following VB.NET sample makes not selectable the first visible item:

```
With AxG2antt1.Items
    .SelectableItem(.FirstVisibleItem) = False
End With
```

The following C# sample makes not selectable the first visible item:

```
axG2antt1.Items.set_SelectableItem(axG2antt1.Items.FirstVisibleItem, false);
```

The following VFP sample makes not selectable the first visible item:

```
with thisform.G2antt1.Items
    .DefaultItem = .FirstVisibleItem
    .SelectableItem(0) = .f.
endwith
```

method Items.SelectAll ()

Selects all items.

Type	Description

Use the SelectAll method to select all visible items in the tree. The SelectAll method has effect only if the SingleSel property is False, if the control supports multiple items selection. Use the UnselectAll method to unselect all items in the list. Use the SelectItem property to select or unselect a specified item. Use the SelectedItem property to retrieve a value that indicates whether the item is selected or unselected. Use the SelectCount property to retrieve the number of selected items

property Items.SelectCount as Long

Counts the number of items that are selected into control.

Type	Description
Long	A long expression that identifies the number of selected items.

The SelectCount property counts the selected items in the control. The SelectCount property gets 0, if no items are selected in the control. The ExG2antt control supports multiple selection. Use the SingleSel property of the control to allow multiple selection. Use the SelectedItem property to retrieve the handle of the selected item(s). The control fires the SelectionChanged event when user changes the selection in the control. Use the SelectItem property to select programmatically an item. Use the SelForeColor and SelBackColor properties to specify colors for selected items. If the control supports only single selection (SingleSel property is True), the FocusItem retrieves the selected item too.

If the control's SingleSel is false, then the following statement retrieves the handle for the selected item: G2antt1.Items.SelectedItem().

If the control supports multiple selection then the following VB sample shows how to enumerate all selected items:

```
Dim h As HITEM
Dim i As Long, j As Long, nCols As Long, nSels As Long
nCols = G2antt1.Columns.Count
With G2antt1.Items
   nSels = .SelectCount
   For i = 0 To nSels - 1
      Dim s As String
      For j = 0 To nCols - 1
         s = s + .CellValue(.SelectedItem(i), j) + Chr(9)
      Next
      Debug.Print s
   Next
End With
```

The following VB sample unselects all items in the control:

```
With G2antt1
```

```
    .BeginUpdate
    With .Items
        While Not .SelectCount = 0
            .SelectItem(.SelectedItem(0)) = False
        Wend
    End With
    .EndUpdate
End With
```

The following C++ sample enumerates the selected items:

```
CItems items = m_g2antt.GetItems();
long n = items.GetSelectCount();
if ( n != 0 )
{
    for ( long i = 0; i < n; i++ )
    {
        long h = items.GetSelectedItem( i );
        COleVariant vtString;
        vtString.ChangeType( VT_BSTR, &items.GetCellValue( COleVariant( h ), COleVariant(
(long)0 ) ) );
        CString str = V_BSTR( &vtString );
        MessageBox( str );
    }
}
```

The following C++ sample unselects all items in the control:

```
m_g2antt.BeginUpdate();
CItems items = m_g2antt.GetItems();
while ( items.GetSelectCount() )
    items.SetSelectItem( items.GetSelectedItem( 0 ), FALSE );
m_g2antt.EndUpdate();
```

The following VB.NET sample enumerates the selected items:

```
With AxG2antt1.Items
    Dim nCols As Integer = AxG2antt1.Columns.Count, i As Integer
    For i = 0 To .SelectCount - 1
```

```
    Debug.Print(.CellValue(.SelectedItem(i), 0))
  Next
End With
```

The following VB.NET sample unselects all items in the control:

```
With AxG2antt1
  .BeginUpdate()
  With .Items
    While Not .SelectCount = 0
      .SelectItem(.SelectedItem(0)) = False
    End While
  End With
  .EndUpdate()
End With
```

The following C# sample enumerates the selected items:

```
for (int i = 0; i < axG2antt1.Items.SelectCount; i++)
{
  object strCaption = axG2antt1.Items.get_CellValue(axG2antt1.Items.get_SelectedItem(i),
0);
  System.Diagnostics.Debug.WriteLine(strCaption != null ? strCaption.ToString() : "");
}
```

The following C# sample unselects all items in the control:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Items items = axG2antt1.Items;
while (items.SelectCount != 0)
  items.set_SelectItem(items.get_SelectedItem(0), false);
axG2antt1.EndUpdate();
```

The following VFP sample enumerates the selected items:

```
with thisform.G2antt1.Items
  local i
  for i = 0 to .SelectCount - 1
    .DefaultItem = .SelectedItem(i)
```

```
      wait window nowait .CellValue(0,0)
   next
endwith
```

The following VFP sample unselects all items in the control:

```
With thisform.G2antt1
   .BeginUpdate()
   with .Items
      do while ( .SelectCount() # 0 )
         .DefaultItem = .SelectedItem(0)
         .SelectItem(0) = .f.
      enddo
   endwith
   .EndUpdate()
EndWith
```

property Items.SelectedItem ([Index as Long]) as HITEM

Retrieves the selected item's handle given its index in selected items collection.

Type	Description
Index as Long	Identifies the index of the selected item into the selected items collection.
HITEM	A long expression that indicates the handle of the selected item.

Use the SelectedItem property to get the handle of the selected item(s) in the control. Use the SelectCount property to find out how many items are selected in the control. The control fires the SelectionChanged event when user changes the selection in the control. Use the SelectItem property to select programmatically an item. If the control supports only single selection, you can use the FocusItem property to get the selected/focused item because they are always the same. Use the SingleSel property to enable single or multiple selection. Use the SelForeColor and SelBackColor properties to specify colors for selected items.

The following sample shows hot to print the caption for the selected item: Debug.Print G2antt1.Items.CellValue(G2antt1.Items.SelectedItem(0), 0).

The following sample applies an italic font attribute to the selected item:

```
Private Sub G2antt1_SelectionChanged()
  If Not (h = 0) Then G2antt1.Items.ItemItalic(h) = False
  h = G2antt1.Items.SelectedItem()
  G2antt1.Items.ItemItalic(h) = True
End Sub
```

The following VB sample enumerates the selected items:

```
Dim i As Long
With G2antt1.Items
  For i = 0 To .SelectCount - 1
    Debug.Print .CellValue(.SelectedItem(i), 0)
  Next
End With
```

The following VB sample unselects all items in the control:

```
With G2antt1
   .BeginUpdate
   With .Items
      While Not .SelectCount = 0
         .SelectItem(.SelectedItem(0)) = False
      Wend
   End With
   .EndUpdate
End With
```

The following VC sample displays the selected items:

```
#include "Items.h"

static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
   if ( pv )
   {
      if ( pv->vt == VT_ERROR )
         return szDefault;

      COleVariant vt;
      vt.ChangeType( VT_BSTR, pv );
      return V_BSTR( &vt );
   }
   return szDefault;
}

CItems items = m_g2antt.GetItems();
for ( long i = 0; i < items.GetSelectCount(); i++ )
{
   COleVariant vtItem( items.GetSelectedItem( i ) );
   CString strOutput;
   strOutput.Format( "%s\n", V2S( &items.GetCellValue( vtItem, COleVariant( (long)0 ) ) ) );
   OutputDebugString( strOutput );
}
```

The following C++ sample unselects all items in the control:

```
m_g2antt.BeginUpdate();
CItems items = m_g2antt.GetItems();
while ( items.GetSelectCount() )
   items.SetSelectItem( items.GetSelectedItem( 0 ), FALSE );
m_g2antt.EndUpdate();
```

The following VB.NET sample displays the selected items:

```
With AxG2antt1.Items
   Dim i As Integer
   For i = 0 To .SelectCount - 1
      Debug.WriteLine(.CellValue(.SelectedItem(i), 0))
   Next
End With
```

The following VB.NET sample unselects all items in the control:

```
With AxG2antt1
   .BeginUpdate()
   With .Items
      While Not .SelectCount = 0
         .SelectItem(.SelectedItem(0)) = False
      End While
   End With
   .EndUpdate()
End With
```

The following C# sample displays the selected items:

```
for ( int i = 0; i < axG2antt1.Items.SelectCount - 1; i++ )
{
   object cell = axG2antt1.Items.get_CellValue( axG2antt1.Items.get_SelectedItem( i), 0 );
   System.Diagnostics.Debug.WriteLine( cell != null ? cell.ToString() : "" );
}
```

The following C# sample unselects all items in the control:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Items items = axG2antt1.Items;
```

```
while (items.SelectCount != 0)
   items.set_SelectItem(items.get_SelectedItem(0), false);
axG2antt1.EndUpdate();
```

The following VFP sample displays the selected items:

```
with thisform.G2antt1.Items
   for i = 0 to .SelectCount - 1
      .DefaultItem = .SelectedItem( i )
      wait window nowait .CellValue( 0, 0 )
   next
endwith
```

The following VFP sample unselects all items in the control:

```
With thisform.G2antt1
   .BeginUpdate()
   with .Items
      do while ( .SelectCount() # 0 )
         .DefaultItem = .SelectedItem(0)
         .SelectItem(0) = .f.
      enddo
   endwith
   .EndUpdate()
EndWith
```

property Items.SelectedObjects (Objects as SelectObjectsEnum) as Variant

Retrieves a collection of selected objects in the chart.

Type	Description
Objects as SelectObjectsEnum	A combination of SelectObjectsEnum values that indicates the objects being returned. For instance, the SelectedObject(exSelectBarsOnly) retrieves only selected bars, or SelectedObject(exSelectBarsOnly Or exObjectsJustAdded) retrieves only the bars that were added to the selection since the selection was changed.
Variant	A Collection of strings, each string indicating the bars or the links, or a String that indicates the first selected bar or link, if the Objects parameter includes the exSelectSingleObject. The bars are returned as HANDLE,"KEY", since the link is returned as "KEY". *Shortly, the SelectedObjects property retrieves a string that indicates the first selected bar or link, if the Objects parameter includes the exSelectSingleObject value, else it returns a collection of strings, that indicates the selected bars or links in the chart*

Use the SelectedObject property to retrieve a collection of selected bars or/and links. The ChartSelectionChanged event notifies your application when the user select objects like bars or links in the chart area. Use the AllowSelectObjects property to specify whether the user can select bars or/and links at runtime, using the mouse. Use the ItemBar(exBarSelected) property to select or unselect programmatically a bar. Use the Link(exLinkSelected) property to select or unselect programmatically a link. Use the RemoveSelection property to remove objects in the chart's selection. Use the ExecuteTemplate property to execute and returns the result of a x-script for the /COM version.

On the bottom of the page you can find samples for using the /NET Assembly or /WPF component. The newer versions of the /NET, WPF version provides the get_SelectedBars and get_SelectedLinks properties that returns a collection of selected bars and links, beside the get_SelectedObjects property that may returns all objects being selected in the Chart area.

In the /COM version let's say that you want to retrieve the name of the bars, in this case you need to build the s-script "*Items.ItemBar(<%BAR%>,0)*" where the <%BAR%> should be replaced with the strings in the collection being returned by the SelectedObjects property, and 0 indicates the exBarName. Once you built this string, you just call the

ExecuteTemplate property and so the name of the bar is returned for each bar selected as shown in the bellow samples. Instead, if links are returned, and you want to access a property of the link, you need to build the x-script "Items.Link(<%LINK%>,12)" where it gets the text being assigned to selected links, as 12 indicates the exLinkText.

The following VB sample displays the list of selected bars:

```
Dim c As Variant
For Each c In G2antt1.Items.SelectedObjects(exSelectBarsOnly)
   Debug.Print c
Next
```

The following VB sample displays the name of the bars being selected:

```
Dim c As Variant
With G2antt1
   For Each c In .Items.SelectedObjects(exSelectBarsOnly)
      Debug.Print .ExecuteTemplate("Items.ItemBar(" & c & "," & exBarName & ")")
   Next
End With
```

The following VB sample removes the selected links only:

```
With G2antt1
   .BeginUpdate
   With .Items
      For Each l In .SelectedObjects(exSelectLinksOnly)
         G2antt1.Template = "Items.RemoveLink(" & l & ")"
      Next
   End With
   .EndUpdate
End With
```

The following VB sample removes the selected bars only:

```
With G2antt1
   .BeginUpdate
   With .Items
      For Each b In .SelectedObjects(exSelectBarsOnly)
         G2antt1.Template = "Items.RemoveBar(" & b & ")"
```

```
    Next
  End With
  .EndUpdate
End With
```

When a bar is removed, any link related to it will be removed.

The following VB.NET sample displays the list of selected bars (applicable to COM inserted to NET forms):

```
  Dim c As String
  For Each c In
AxG2antt1.Items.SelectedObjects(EXG2ANTTLib.SelectObjectsEnum.exSelectBarsOnly)
    Debug.Print(c)
  Next
```

The following VB.NET sample displays the name of the bars being selected (applicable to COM inserted to NET forms):

```
  Dim c As String
  With AxG2antt1
    For Each c In
.Items.SelectedObjects(EXG2ANTTLib.SelectObjectsEnum.exSelectBarsOnly)
      Dim t As String = "Items.ItemBar(" + c + "," +
Int(EXG2ANTTLib.ItemBarPropertyEnum.exBarName).ToString() + ")"
      Debug.Print(.ExecuteTemplate(t))
    Next
  End With
```

The following C# sample displays the list of selected bars (applicable to COM inserted to NET forms):

```
  foreach (string c in
axG2antt1.Items.get_SelectedObjects(EXG2ANTTLib.SelectObjectsEnum.exSelectBarsOnly)
as Array )
  {
    System.Diagnostics.Debug.WriteLine( c );
  }
```

The following C# sample displays the name of the bars being selected (applicable to NET

assemblies inserted to NET forms):

```
exontrol.EXG2ANTTLib.Items items = exg2antt1.Items;
foreach(string bar in
items.get_SelectedObjects(exontrol.EXG2ANTTLib.SelectObjectsEnum.exSelectBarsOnly)
as Array)
{
   string[] s = bar.Split(new Char[] {','});
   System.Diagnostics.Debug.Print(items.get_ItemBar(Int32.Parse(s[0]), s[1].Substring(0,
s[1].Length-2), exontrol.EXG2ANTTLib.ItemBarPropertyEnum.exBarName).ToString());
}
```

The following VB.NET sample displays the name of the bars being selected (applicable to NET assemblies inserted to NET forms):

```
With Exg2antt1.Items
   Dim bar As String, s As String()
   For Each bar In
CType(.get_SelectedObjects(exontrol.EXG2ANTTLib.SelectObjectsEnum.exSelectBarsOnly),
Object())
      s = Split(bar, ",")
      Debug.Print(.get_ItemBar(CInt(s(0)), Mid(s(1), 2, Len(s(1)) - 2),
exontrol.EXG2ANTTLib.ItemBarPropertyEnum.exBarName).ToString())
   Next
End With
```

The following C++ sample displays the list of selected bars:

```
#include "Items.h"
{
   COleVariant vtSelected = m_g2antt.GetItems().GetSelectedObjects( 1 );
//exSelectBarsOnly
   if ( V_VT( &vtSelected ) & VT_ARRAY | VT_VARIANT )
   {
      SAFEARRAY* pArray = V_ARRAY( &vtSelected );
      void* pData = NULL;
      if ( SUCCEEDED( SafeArrayAccessData( pArray, &pData ) ) )
      {
```

```
        VARIANT* p = (VARIANT*)pData;
        for ( long i = 0; i < (long)pArray->rgsabound[0].cElements ; i++, p++ )
          OutputDebugString( V2S( p ) );
        SafeArrayUnaccessData( pArray );
      }
    }
}
```

The following C++ sample displays the name of the bars being selected:

```
#include "Items.h"
{
   COleVariant vtSelected = m_g2antt.GetItems().GetSelectedObjects( 1
/*exSelectBarsOnly*/ );
   if ( V_VT( &vtSelected ) & VT_ARRAY | VT_VARIANT )
   {
      SAFEARRAY* pArray = V_ARRAY( &vtSelected );
      void* pData = NULL;
      if ( SUCCEEDED( SafeArrayAccessData( pArray, &pData ) ) )
      {
         VARIANT* p = (VARIANT*)pData;
         for ( long i = 0; i < (long)pArray->rgsabound[0].cElements ; i++, p++ )
         {
            CString strT = "Items.ItemBar(" + V2S( p ) + ",0)"; /*builds the
Items.ItemBar(Handle,Key,exBarName) template*/
            OutputDebugString( V2S( &m_g2antt.ExecuteTemplate( strT ) ) );
         }
         SafeArrayUnaccessData( pArray );
      }
   }
}
```

where the V2S string may look like follows:

```
static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
   if ( pv )
   {
```

```
    if ( pv->vt == VT_ERROR )
        return szDefault;

    COleVariant vt;
    vt.ChangeType( VT_BSTR, pv );
    return V_BSTR( &vt );
  }
  return szDefault;
}
```

or

```
static string V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
  if ( pv )
  {
    if ( pv->vt == VT_ERROR )
      return szDefault;

    CComVariant vt;
    if ( SUCCEEDED( vt.ChangeType( VT_BSTR, pv ) ) )
    {
      USES_CONVERSION;
      return OLE2T(V_BSTR( &vt ));
    }
  }
  return szDefault;
}
```

The following VFP sample displays the list of selected bars:

```
local c
For Each c In thisform.G2antt1.Items.SelectedObjects(1)
  wait window c
Next
```

The following VFP sample displays the name of the bars being selected:

```
local c
```

```
For Each c In thisform.G2antt1.Items.SelectedObjects(1)
   local t
   t = "Items.ItemBar(" + c + ",0)"
   wait window thisform.G2antt1.ExecuteTemplate(t)
Next
```

In the /NET assembly you can use the Items.get_ItemBar or Items.set_ItemBar to access properties of the bar giving its handle and key. The key of the bar is contained between " characters so if you are using the ItemBar property make sure that you are removing the " characters from start and end position. The get_SelectedObjects property retrieves an array of string objects. If the string starts with the " character it means that it is a link, else it is a bar. The name of the link is contained between " characters, while the bar information contains the handle of the item and the key of the bar as (item,"key"), where the item is the handle of the item, while the key is the key of the bar.

The following C# sample changes the color of the selected bar(s):

```
private void exg2antt1_ChartSelectionChanged(object sender)
{
   foreach (string o in
exg2antt1.Items.get_SelectedObjects(exontrol.EXG2ANTTLib.SelectObjectsEnum.exSelectBa
 as Array)
   {
      String[] b = o.Split(",".ToCharArray());
      exg2antt1.Items.set_BarColor( int.Parse(b[0]), b[1].Substring(1,b[1].Length -2),
Color.Red );
   }
}
```

The following C# sample changes the color of the selected link(s):

```
private void exg2antt1_ChartSelectionChanged(object sender)
{
   foreach (string o in
exg2antt1.Items.get_SelectedObjects(exontrol.EXG2ANTTLib.SelectObjectsEnum.exSelectLir
 as Array)
   {
      exg2antt1.Items.set_Link(o.Substring(1, o.Length - 2),
exontrol.EXG2ANTTLib.LinkPropertyEnum.exLinkColor,
```

```
ColorTranslator.ToWin32(Color.Red));
    }
}
```

The following VB.NET sample changes the color of the selected bar(s):

```
Private Sub Exg2antt1_ChartSelectionChanged(ByVal sender As System.Object) Handles
Exg2antt1.ChartSelectionChanged
    Dim o As String
    For Each o In
Exg2antt1.Items.get_SelectedObjects(exontrol.EXG2ANTTLib.SelectObjectsEnum.exSelectBa

        Dim b As String() = o.Split(",".ToCharArray())
        Exg2antt1.Items.set_BarColor(CInt(b(0)), b(1).Substring(1, b(1).Length - 2), Color.Red)
    Next
End Sub
```

The following VB.NET sample changes the color of the selected link(s):

```
Private Sub Exg2antt1_ChartSelectionChanged(ByVal sender As System.Object) Handles
Exg2antt1.ChartSelectionChanged
    Dim o As String = ""
    For Each o In
Exg2antt1.Items.get_SelectedObjects(exontrol.EXG2ANTTLib.SelectObjectsEnum.exSelectLir

        Exg2antt1.Items.set_Link(o.Substring(1, o.Length - 2),
exontrol.EXG2ANTTLib.LinkPropertyEnum.exLinkColor,
ColorTranslator.ToWin32(Color.Red))
    Next
End Sub
```

The newer versions of the /NET,WPF version provides the get_SelectedBars and get_SelectedLinks properties that returns a collection of selected bars and links.

The following C# sample changes the color of the selected bar(s):

```
private void exg2antt1_ChartSelectionChanged(object sender)
{
    List<exontrol.EXG2ANTTLib.Items.SelectedBar> sBars =
```

```
exg2antt1.Items.get_SelectedBars();
   if (sBars != null)
      foreach (exontrol.EXG2ANTTLib.Items.SelectedBar bar in sBars)
         exg2antt1.Items.set_BarColor(bar.Item, bar.Key, Color.Red);
}
```

The following C# sample changes the color of the selected link(s):

```
private void exg2antt1_ChartSelectionChanged(object sender)
{
   List<string> sLinks = exg2antt1.Items.get_SelectedLinks();
   if (sLinks != null)
      foreach (string link in sLinks)
         exg2antt1.Items.set_Link(link,
exontrol.EXG2ANTTLib.LinkPropertyEnum.exLinkColor,
ColorTranslator.ToWin32(Color.Red));
}
```

The following VB.NET sample changes the color of the selected bar(s):

```
Private Sub Exg2antt1_ChartSelectionChanged(ByVal sender As System.Object) Handles
Exg2antt1.ChartSelectionChanged
   With Exg2antt1
      Dim sBars As List(Of exontrol.EXG2ANTTLib.Items.SelectedBar) =
.Items.get_SelectedBars()
      If Not (sBars Is Nothing) Then
         Dim bar As exontrol.EXG2ANTTLib.Items.SelectedBar
         For Each bar In sBars
            .Items.set_BarColor(bar.Item, bar.Key, Color.Red)
         Next
      End If
   End With
End Sub
```

The following VB.NET sample changes the color of the selected link(s):

```
Private Sub Exg2antt1_ChartSelectionChanged(ByVal sender As System.Object) Handles
Exg2antt1.ChartSelectionChanged
   With Exg2antt1
```

```
      Dim sLinks As List(Of String) = .Items.get_SelectedLinks()
      If Not (sLinks Is Nothing) Then
         Dim link As String
         For Each link In sLinks
            .Items.set_Link(link, exontrol.EXG2ANTTLib.LinkPropertyEnum.exLinkColor,
ColorTranslator.ToWin32(Color.Red))
         Next
      End If
   End With
End Sub
```

property Items.SelectItem(Item as HITEM) as Boolean

Selects or unselects a specific item.

Type	Description
Item as HITEM	A long expression that indicates the item's handle that is selected or unselected.
Boolean	A boolean expression that indicates the item's state. True if the item is selected, and False if the item is not selected.

Use the SelectItem to select or unselect a specified item (that's selectable). Use the SelectableItem property to specify the user can select an item. Use the SelectCount property to get the number of selected items. Use the SelectedItem property to get the selected item. Use the FocusItem property to get the focused item. If the control supports only single selection, you can use the FocusItem property to get the selected/focused item because they are always the same. The control fires the SelectionChanged event when user selects an item. Use the SelForeColor and SelBackColor properties to specify colors for selected items. Use the SingleSel property to allow multiple selection. Use the SelectPos property to select an item giving its position. Use the EnsureVisibleItem property to ensure that an item is visible. The AllowSelectObjects property allows users to select at runtime the bars and links in the chart area. Use the ItemBar(exBarSelected) property to select or unselect programmatically a bar. Use the Link(exLinkSelected) property to select or unselect programmatically a link.

The following VB sample shows how to select the first created item:
G2antt1.Items.SelectItem(G2antt1.Items(0)) = True

The following VB sample selects the first visible item:

```
With G2antt1.Items
   .SelectItem(.FirstVisibleItem) = True
End With
```

The following VB sample enumerates the selected items:

```
Dim i As Long
With G2antt1.Items
   For i = 0 To .SelectCount - 1
      Debug.Print .CellValue(.SelectedItem(i), 0)
   Next
```

```
End With
```

The following C++ sample selects the first visible item:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
items.SetSelectItem( items.GetFirstVisibleItem(), TRUE );
```

The following C++ sample unselects all items in the control:

```
m_g2antt.BeginUpdate();
CItems items = m_g2antt.GetItems();
while ( items.GetSelectCount() )
   items.SetSelectItem( items.GetSelectedItem( 0 ), FALSE );
m_g2antt.EndUpdate();
```

The following VB.NET sample selects the first visible item:

```
With AxG2antt1.Items
   .SelectItem(.FirstVisibleItem) = True
End With
```

The following VB.NET sample unselects all items in the control:

```
With AxG2antt1
   .BeginUpdate()
   With .Items
      While Not .SelectCount = 0
         .SelectItem(.SelectedItem(0)) = False
      End While
   End With
   .EndUpdate()
End With
```

The following C# sample selects the first visible item:

```
axG2antt1.Items.set_SelectItem(axG2antt1.Items.FirstVisibleItem, true);
```

The following C# sample unselects all items in the control:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Items items = axG2antt1.Items;
while (items.SelectCount != 0)
    items.set_SelectItem(items.get_SelectedItem(0), false);
axG2antt1.EndUpdate();
```

The following VFP sample selects the first visible item:

```
with thisform.G2antt1.Items
    .DefaultItem = .FirstVisibleItem
    .SelectItem(0) = .t.
endwith
```

The following VFP sample unselects all items in the control:

```
With thisform.G2antt1
    .BeginUpdate()
    with .Items
        do while ( .SelectCount() # 0 )
            .DefaultItem = .SelectedItem(0)
            .SelectItem(0) = .f.
        enddo
    endwith
    .EndUpdate()
EndWith
```

property Items.SelectPos as Variant

Selects items by position.

Type	Description
Variant	A long expression that indicates the position of item being selected, or a safe array that holds a collection of position of items being selected.

Use the SelectPos property to select items by position. Use the SelectItem property to select an item giving its handle. The SelectPos property selects an item giving its general position. The ItemPosition property gives the relative position, or the position of the item in the child items collection. The AllowSelectObjects property allows users to select at runtime the bars and links in the chart area. Use the ItemBar(exBarSelected) property to select or unselect programmatically a bar. Use the Link(exLinkSelected) property to select or unselect programmatically a link.

The following VB sample selects the first item in the control:

```
G2antt1.Items.SelectPos = 0
```

The following VB sample selects first two items:

```
G2antt1.Items.SelectPos = Array(0, 1)
```

The following C++ sample selects the first item in the control:

```
m_g2antt.GetItems().SetSelectPos( COleVariant( long(0) ) );
```

The following VB.NET sample selects the first item in the control:

```
With AxG2antt1.Items
   .SelectPos = 0
End With
```

The following C# sample selects the first item in the control:

```
axG2antt1.Items.SelectPos = 0;
```

The following VFP sample selects the first item in the control:

```
with thisform.G2antt1.Items
   .SelectPos = 0
```

endwith

method Items.SetParent (Item as HITEM, NewParent as HITEM)

Changes the parent of the given item.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item being moved.
NewParent as HITEM	A long expression that indicates the handle of the new parent item.

Use the SetParent property to change the parent item at runtime. Use the InsertItem property to insert child items. Use the InsertControlItem property to insert ActiveX controls. Use AcceptSetParent property to verify if the the parent of an item can be changed. The following VB sample changes the parent item of the first item: G2antt1.Items.SetParent G2antt1.Items(0), G2antt1.Items(1). Use the ItemParent property to retrieve the parent of the item.

property Items.SortableItem(Item as HITEM) as Boolean

Specifies whether the item is sortable.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item being sortable.
Boolean	A boolean expression that specifies whether the item is sortable.

By default, all items are sortable. A sortable item can change its position after sorting. An unsortable item keeps its position after user performs a sort operation. Thought, the position of an unsortable item can be changed using the ItemPosition property. Use the SortableItem to specify a group item, a total item or a separator item. An unsortable item is not counted by a total field. The SortType property specifies the type of repositioning is being applied on the column when a sort operation is performed. The SortOrder property specifies whether the column is sorted ascendant or descendent. Use the SortChildren method to sort the items. Use the AllowSort property to avoid sorting a column when the user clicks the column. The ItemDivider property indicates whether the item displays a single cell, instead showing all cells. The SelectableItem property specifies whether an item can be selected.

The following screen shots shows the control when no column is sorted: (Group 1 and Group 2 has the SortableItem property on False)

Name	A	B	C
		Group 1	
Child 1	1	2	3
Child 2	4	5	6
		Group 2	
Child 1	1	2	3
Child 2	4	5	6

The following screen shots shows the control when the column A is being sorted: (Group 1 and Group 2 keeps their original position after sorting)

Name	A	▽ B	C
		Group 1	
Child 2	4	5	6
Child 1	1	2	3
		Group 2	
Child 2	4	5	6
Child 1	1	2	3

method Items.SortChildren (Item as HITEM, ColIndex as Variant, Ascending as Boolean)

Sorts the child items of the given parent item in the control.

Type	Description
Item as HITEM	A long expression that indicates the item's handle that is going to be sorted.
ColIndex as Variant	A long expression that indicates the column's index or the cell's handle, a string expression that indicates the column's caption.
Ascending as Boolean	A boolean expression that defines the sort order.

The SortChildren will not recurse through the tree, only the immediate children of item will be sorted. If your control acts like a simple list you can use the following line of code to sort ascending the list by first column: G2antt1.Items.SortChildren 0, 0. To change the way how a column is sorted use SortType property of Column object. The SortChildren property doesn't display the sort icon on column's header. The control automatically sorts the children items when user clicks on column's header, depending on the SortOnClick property. The SortOrder property sorts the items and displays the sorting icon in the column's header. Use the AllowSort property to avoid sorting a column when the user clicks the column.

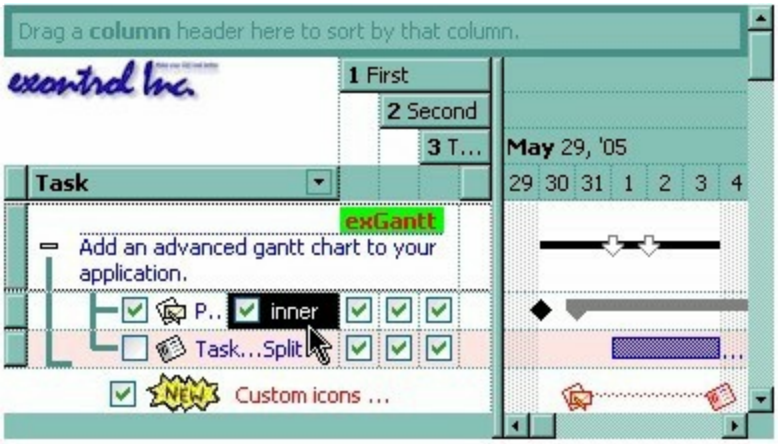
property Items.SplitCell ([Item as Variant], [ColIndex as Variant]) as Variant

Splits a cell, and returns the inner created cell.

Type	Description
Item as Variant	A long expression that indicates the handle of the item where a cell is being divided, or 0. If the Item parameter is 0, the ColIndex parameter must indicate the handle of the cell.
ColIndex as Variant	A long expression that indicates the index of the column where a cell is divided, or a long expression that indicates the handle of the cell being divided, if the Item parameter is missing or it is zero.
Variant	A long expression that indicates the handle of the cell being created.

The SplitCell method splits a cell in two cells. The newly created cell is called inner cell. The SplitCell method always returns the handle of the inner cell. If the cell is already divided using the SplitCell method, it returns the handle of the inner cell without creating a new inner cell. You can split an inner cell too, and so you can have a master cell divided in multiple cells. Use the CellWidth property to specify the width of the inner cell. Use the CellValue property to assign a caption to a cell. Use the InnerCell property to access an inner cell giving its index. Use the CellParent property to get the parent of the inner cell. Use the CellItem property to get the owner of the cell. Use the UnsplitCell method to remove the inner cell if it exists. Use the MergeCells property to combine two or more cells in a single cell. Use the SelectableItem property to specify the user can select an item. Include the exIncludeInnerCells flag in the FilterList property and so the drop down filter window lists the inner cells too.

("Merge" means multiple cells in a single cell, "Split" means multiple cells **inside** a single cell)



The following VB sample splits a single cell in two cells (Before running the following sample, please make sure that your control contains columns, and at least an item):

```
With G2antt1.Items
    Dim h As HITEM, f As HCELL
    h = .FirstVisibleItem
    f = .SplitCell(h, 0)
    .CellValue(, f) = "inner cell"
End With
```

The following C++ sample splits the first visible cell in two cells:

```
#include "Items.h"
CItems items = m_g2antt.GetItems();
COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
COleVariant vtSplit = items.GetSplitCell( COleVariant( items.GetFirstVisibleItem() ),
COleVariant( long(0) ) );
items.SetCellValue( vtMissing, vtSplit, COleVariant( "inner cell" ) );
```

The following VB.NET sample splits the first visible cell in two cells:

```
With AxG2antt1.Items
    Dim i As Object
    i = .SplitCell(.FirstVisibleItem, 0)
    .CellValue(Nothing, i) = "inner cell"
End With
```

The following C# sample splits the first visible cell in two cells:

```
EXG2ANTTLib.Items items = axG2antt1.Items;
object i = items.get_SplitCell(items.FirstVisibleItem, 0);
items.set_CellValue(null, i, "inner cell");
```

The following VFP sample splits the first visible cell in two cells:

```
with thisform.G2antt1.Items
    local i
    i = .SplitCell(.FirstVisibleItem,0)
    local s, crlf
    crlf = chr(13) + chr(10)
```

```
    s = "Items" + crlf
    s = s + "{" + crlf
    s = s + "CellValue(," + str(i) + ") = " + chr(34) + "inner cell" + chr(34) + crlf
    s = s + "}"
    thisform.G2antt1.Template = s
endwith
```

method Items.StartBlockUndoRedo ()

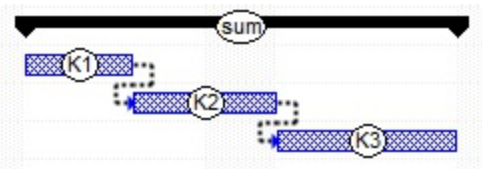
Starts recording the UI operations as a block of undo/redo operations.

Type	Description

The StartBlockUndoRedo method starts recording the UI operations as a block on undo/redo operations (equivalent of StartBlockUndoRedo method of the Chart object). The method has effect only if the AllowUndoRedo property is True. The EndBlockUndoRedo method collects all undo/redo operations since StartBlockUndoRedo method was called and add them to the undo/redo queue as a block. This way the next call on a Undo operation, the entire block is restored, so all UI operations are restored. For instance, if you have a procedure that moves several bars, and want all of them being grouped, you can use StartBlockUndoRedo to start recording the operations as a block, and call the EndBlockUndoRedo when procedure ends, so next call of an undo operation the bars are restored to their original position. The EndBlockUndoRedo method must be called the same number of times as the StartBlockUndoRedo method was called. For instance, if you have called the StartBlockUndoRedo twice the EndBlockUndoRedo method must be called twice too, and the collected operations are added to the chart's queue of undo/redo operations at the end.

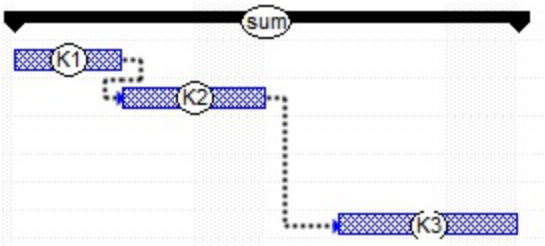
The chart fires the ChartStartChanging event when the user starts an UI operation, for instance, moving a bar. The ChartEndChanging event notifies your application once the user operation on the chart ends. By default, each undo/redo operation is added sequentially as they occur. You can call the StartBlockUndoRedo method during the ChartStartChanging event so all operations that are about to begin will be as a block when calling the EndBlockUndoRedo during the ChartEndChanging event. For instance, if a bar is related to multiple bars using grouping options, so if a bar is moved other bars must be moved, the undo/redo operations are added sequentially as they appear. So calling the Undo action will restore moving a bar once at the time. Using the StartBlockUndoRedo/EndBlockUndoRedo methods you can control the block of undo/redo operations being grouped in a block, so next time the Undo/Redo operation is performed, the entire block of operations is performed or restored at once. For instance, the SchedulePDM method performs multiple operations during bars, so all of them are grouped as a block.

For instance, we we have the following chart:



In this case the, the K1, K2 and K3 bars are grouped, so moving any bar will result in moving relative bars.

The following screen shot shows the chart after moving the bar K3 to a new position as well as a to a new parent,



so the undo/redo queue looks like:

StartBlock MoveBar;1;sum
MoveBar;2;K4
MoveBar;3;K3
EndBlock
ParentChangeBar;2;K3

In this case, we need to press twice the CTRL + Z to restore back the chart as it was before moving the bar K3.

Instead if we are using the StartBlockUndoRedo and EndBlockUndoRedo methods as follow:

```
Private Sub G2antt1_ChartStartChanging(ByVal Operation As
EXG2ANTTLibCtl.BarOperationEnum)
   G2antt1.Chart.StartBlockUndoRedo
End Sub

Private Sub G2antt1_ChartEndChanging(ByVal Operation As
EXG2ANTTLibCtl.BarOperationEnum)
   G2antt1.Chart.EndBlockUndoRedo
End Sub
```

We have the undo/redo queue as follows (if we perform the same operation):

StartBlock
MoveBar;1;sum
MoveBar;2;K4
MoveBar;3;K3
ParentChangeBar;2;K3
EndBlock

In this case, we need to press only once the CTRL + Z to restore back the chart as it was

before moving the bar K3.

property Items.StartUpdateBar (Item as HITEM, BarKey as Variant) as Long

Starts changing properties of the bar, so EndUpdateBar method adds programmatically updated properties to undo/redo queue.

Type	Description
Item as HITEM	A long expression that specifies the handle of the item that holds the bar to be updated.
BarKey as Variant	A VARIANT expression that holds the key of the bar being updated. Use the AddBar method to add programmatically bars.
Long	A Long expression that specifies the handle to be passed to EndUpdateBar so the updated properties of the bar are added to the Undo/Redo queue of the chart, so they can be used in undo/redo operations.

Use the StartUpdateBar and EndUpdateBar methods to add new entries in the chart's undo/redo queue for properties of the bar being updated by code. The ItemBar property accesses the properties of the bar. For instance, if your application provides UI dialogs or forms that help users changing the properties of the selected bar such as color, text, tooltips and so on, you can provide undo/redo operations for them by using the StartUpdateBar and EndUpdateBar methods. Shortly, the StartUpdateBar method starts recording the properties being changed until the EndUpdateBar method is called. The EndUpdateBar method actually adds a new entry to the undo/redo queue based on the changed properties. If there were no changes of the bar during the Star/End session, no new entry is added. The EndUpdateBar method adds UpdateBar entries to the undo/redo queue.

The AllowUndoRedo property specifies whether the chart supports undo/redo operations for objects in the chart such as bars or links. The ChartStartChanging(exUndo/exRedo) / ChartEndChanging(exUndo/exRedo) event notifies your application whenever an Undo/Redo operation is performed. The UndoListAction property lists the Undo actions that can be performed in the chart. The RedoListAction property lists the Redo actions that can be performed in the chart.

The following VB sample adds a new entry "UpdateBar" in the chart's undo/redo queue for changing the text of the bar (/COM version):

```
With G2antt1.Items
  Dim hItem As Long
  hItem = .FocusItem
```

```
    Dim barKey As Variant
    barKey = .FirstItemBar(hItem)

    Dim iChangeBar As Long
    iChangeBar = .StartUpdateBar(hItem, barKey)
    .ItemBar(hItem, barKey, exBarCaption) = "new caption"
    .EndUpdateBar (iChangeBar)
End With
```

The following VB/NET sample adds a new entry "UpdateBar" in the chart's undo/redo queue for changing the text of the bar (/NET Assembly version):

```
With Exg2antt1.Items
    Dim hItem As Long = .FocusItem
    Dim barKey As Object = .get_FirstItemBar(hItem)

    Dim iChangeBar As Long = .get_StartUpdateBar(hItem, barKey)
    .set_ItemBar(hItem, barKey, exontrol.EXG2ANTTLib.ItemBarPropertyEnum.exBarCaption,
"new caption")
    .EndUpdateBar(iChangeBar)
End With
```

These samples add new entries to undo/redo queue as : "UpdateBar;94980832;B1;3;;new caption " which indicates , the handle of the items where the bar has been changed, the bar of the key as being B1, the 3 indicates the exBarCaption predefined value, and so on. Once the sample is called, the bar's caption is changed, and using the CTRL + Z, you can restore back the old value, or pressing the CTRL + Y you can change back after restoring.

property Items.StartUpdateLink (LinkKey as Variant) as Long

Starts changing properties of the link, so EndUpdateLink method adds programmatically updated properties to undo/redo queue.

Type	Description
LinkKey as Variant	A VARIANT expression that specifies the key of the link being updated. Use the AddLink method to add programmatically new links between bars.
Long	A Long expression that specifies the handle to be passed to EndUpdateLink so the updated properties of the link are added to the Undo/Redo queue of the chart, so they can be used in undo/redo operations.

Use the StartUpdateLink and EndUpdateLink methods to add new entries in the chart's undo/redo queue for properties of the link being updated by code. The Link property accesses the properties of the link. For instance, if your application provides UI dialogs or forms that help users changing the properties of the selected link such as color, text, tooltips and so on, you can provide undo/redo operations for them by using the StartUpdateLink and EndUpdateLink methods. Shortly, the StartUpdateLink method starts recording the properties being changed until the EndUpdateLink method is called. The EndUpdateLink method actually adds a new entry to the undo/redo queue based on the changed properties. If there were no changes of the link during the Star/End session, no new entry is added. The EndUpdateLink method adds UpdateLink entries to the undo/redo queue.

The AllowUndoRedo property specifies whether the chart supports undo/redo operations for objects in the chart such as bars or links. The ChartStartChanging(exUndo/exRedo) / ChartEndChanging(exUndo/exRedo) event notifies your application whenever an Undo/Redo operation is performed. The UndoListAction property lists the Undo actions that can be performed in the chart. The RedoListAction property lists the Redo actions that can be performed in the chart.

The following VB sample adds a new entry "UpdateLink" in the chart's undo/redo queue for changing the text being displayed on the link (/COM version):

```
With G2antt1.Items
  Dim linkKey As Variant
  linkKey = .FirstLink
  Dim iChangeLink As Long
  iChangeLink = .StartUpdateLink(linkKey)
  .Link(linkKey, exLinkText) = "new text"
```

```
    .EndUpdateLink (iChangeLink)
End With
```

The following VB/NET sample adds a new entry "UpdateLink" in the chart's undo/redo queue for changing the text being displayed on the link (/NET Assembly version):

```
With Exg2antt1.Items
    Dim linkKey As Object = .get_FirstLink
    Dim iChangeLink As Long = .get_StartUpdateLink(linkKey)
    .set_Link(linkKey, exontrol.EXG2ANTTLib.LinkPropertyEnum.exLinkText, "new text")
    .EndUpdateLink(iChangeLink)
End With
```

These samples add new entries to undo/redo queue as : "UpdateLink;L1;12;;new text " which indicates , the link as being L1, the 12 indicates the exLinkText predefined value, and so on. Once the sample is called, the link's text is changed, and using the CTRL + Z, you can restore back the old value, or pressing the CTRL + Y you can change back after restoring.

method Items.UndefineSummaryBars (SummaryItem as HITEM, SummaryKey as Variant, ItemRemove as HITEM, KeyRemove as Variant)

Undefines the bars in a summary bar

Type	Description
SummaryItem as HITEM	A long expression that specifies the handle of the item that displays the summary bar.
SummaryKey as Variant	A VARIANT expression that indicates the key of the summary bar.
ItemRemove as HITEM	A long expression that specifies the item that holds the bar being removed from the summary bar. The ItemRemove parameter could be • a valid handle, indicating the item itself • 0 indicates all items • -1 indicates the direct descendents/children items of the SummaryItem (child items of the SummaryItem) • -2 means leaf descendents/items of the SummaryItem, where a leaf or terminal item is an item with no child items • -3 means all descendents/children items of the SummaryItem (recursively) For instance, UndefineSummaryBars(SummaryItem,SummaryKey,-1,"<K*>") excludes the bar with the key starting with K from direct descendents of the SummaryItem The 0, -1, -2 and -3 values are supported, starting from the version 12.0
KeyRemove as Variant	A VARIANT expression that indicates the key of the bar being removed from the summary bar. The KeyRemove parameter supports pattern if specified such as "<pattern>", where the pattern may contain wild card characters such as '?' for any single character, '*' for zero or more occurrences of any character, '#' for any digit character. For instance, UndefineSummaryBars(,,,"<K*>") excludes the bars with the key that starts with K, from the SummaryItem/SummaryKey bar

The "<pattern>" syntax is supported, starting from the version 12.0

The UndefineSummaryBars method does the reverse operation of the DefineSummaryBars, as it removes a bar from a summary bar. The DefineSummaryBars method defines bars being displayed under a summary bar. Once a bar that's included in a summary bar is moved or resized, its summary bar is automatically updated. Once a summary bar is moved all included bars are moved too. For instance, if your chart displays a "Summary" or "Project Summary" predefined bar, you can use the DefineSummaryBars method to define the bars included in the summary bar, so they automatically update the summary bars when moving or resizing. The DefineSummaryBars method defines a group of bars that belongs to another bar (called summary bar), so the margins of the summary bars are min and max of the margins of included bars. The margins of the bars are determined by ItemBar(exBarStart) and ItemBar(exBarEnd).

method Items.UngroupBars (ItemA as HITEM, KeyA as Variant, ItemB as HITEM, KeyB as Variant)

Ungroups two bars.

Type	Description
ItemA as HITEM	A long expression that indicates the handle of the item that contains the bar to ungroup.
KeyA as Variant	A long or string expression that specifies the key of the bar to ungroup. The Key parameter of the AddBar method specifies the key of the bar being added.
ItemB as HITEM	A long expression that indicates the handle of the item that contains the bar being ungrouped.
KeyB as Variant	A long or string expression that specifies the key of the bar being ungrouped. The Key parameter of the AddBar method specifies the key of the bar being added.

The UngroupBars method ungroups two bars. Use the UngroupBars method to ungroup bars being grouped using the GroupBars method. Use the Link(exLinkGroupBars) on exGroupBarsNone to ungroup the linked bars.

The UngroupBars method works as follow:

- If the ItemA and ItemB parameters are 0 all groups of bars are removed, so the chart has no grouping bars.
- If the ItemA and KeyA point to a valid bar, and ItemB parameter is 0, the bar A is removed from all groups, in other words the bar A does not belong to any existing group.
- If the (ItemA and KeyA) and (ItemB and KeyB) point to valid bars, the bar A and bar B are ungrouped.

Use the RemoveLink method to remove a specified link. Use the RemoveLinksOf method to remove all links that start or end on the specified bar.

method Items.UnmergeCells ([Cell as Variant])

Unmerges a list of cells.

Type	Description
Cell as Variant	A long expression that indicates the handle of the cell being unmerged, or a safe array that holds a collection of handles for the cells being unmerged. Use the ItemCell property to retrieves the handle of the cell.

Use the UnmergeCells method to unmerge merged cells. Use the MergeCells method or CellMerge property to combine (merge) two or more cells in a single one. The UnmergeCells method unmerges all the cells that was merged. The CellMerge property unmerges only a single cell. The rest of merged cells remains combined.

The following samples show few methods to unmerge cells:

```
With G2antt1
   With .Items
      .UnmergeCells .ItemCell(.RootItem(0), 0)
   End With
End With
```

```
With G2antt1
   With .Items
      Dim r As Long
      r = .RootItem(0)
      .UnmergeCells Array(.ItemCell(r, 0), .ItemCell(r, 1))
   End With
End With
```

```
With G2antt1
   .BeginUpdate
   With .Items
      .CellMerge(.RootItem(0), 0) = -1
      .CellMerge(.RootItem(0), 1) = -1
      .CellMerge(.RootItem(0), 2) = -1
   End With
   .EndUpdate
End With
```

method Items.UnselectAll ()

Unselects all items.

Type	Description

Use the UnselectAll method to unselect all items in the list. The UnselectAll method has effect only if the SingleSel property is False, if the control supports multiple items selection. Use the SelectAll method to select all items in the list. Use the SelectItem property to select or unselect a specified item. Use the SelectedItem property to retrieve a value that indicates whether the item is selected or unselected. Use the SelectCount property to retrieve the number of selected items

method Items.UnsplitCell ([Item as Variant], [ColIndex as Variant])

Unsplits a cell.

Type	Description
Item as Variant	A long expression that indicates the handle of the item, or 0. If the Item parameter is 0, the ColIndex parameter must indicate the handle of the cell.
ColIndex as Variant	A long expression that indicates the index of the column where a cell is divided, or a long expression that indicates the handle of the cell being divided, if the Item parameter is missing or it is zero.

Use the UnsplitCells method to remove the inner cells. The SplitCell method splits a cell in two cells, and retrieves the newly created cell. The UnsplitCell method has no effect if the cell contains no inner cells. The UnplitCells method remove recursively all inner cells. For instance, if a cell contains an inner cell, and this inner cell contains another inner cell, when calling the UnplitCells method for the master cell, all inner cells inside of the cell will be deleted. Use the CellParent property to get the parent of the inner cell. Use the CellItem property to get the owner of the cell. Use the InnerCell property to access an inner cell giving its index. Use the UnmergeCells method to unmerge merged cells. ("Merge" means multiple cells in a single cell, "Split" means multiple cells **inside** a single).

property Items.VisibleCount as Long

Retrieves the number of visible items.

Type	Description
Long	Counts the visible items.

Use FirstVisibleItem and NextVisibleItem properties to determine the items that fit the client area. Use the IsItemVisible property to check whether an item fits the control's client area. Use the ItemCount property to count the items in the control. Use the ChildCount property to count the child items. The ItemPosition property determines the position of the item in the parent's child collection. The FormatColumn event is fired before displaying a cell, so you can handle the FormatColumn to display anything on the cell at runtime. This way you can display the row position, you can display the value using the currency format, and so on. The FireFormatColumn property allows the control to fire the FormatColumn event for the column. The Position property specifies the position of the column.

If your chart displays a tree or a hierarchy the position of the item must be determined relative to the FirstVisibleItem as shown in the following VB sample:

```
Private Sub G2antt1_FormatColumn(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal
ColIndex As Long, Value As Variant)
  Value = G2antt1.ScrollPos(True) + RelPos(Item)
End Sub

Private Function RelPos(ByVal hVisible As Long) As Long
  With G2antt1.Items
    Dim h As Long, i As Long, n As Long
    i = 0
    n = .VisibleCount + 1
    h = .FirstVisibleItem
    While (i <= n) And h <> 0 And h <> hVisible
      i = i + 1
      h = .NextVisibleItem(h)
    Wend
    RelPos = i
  End With
End Function
```

property Items.VisibleItemCount as Long

Retrieves the number of visible items.

Type	Description
Long	A long expression that specifies the number of visible items in the control. The value could be a positive value if no filter is applied, or negative while filter is on.

The VisibleItemCount property counts the number of visible items in the list. For instance, you can use the VisibleItemCount property to get the number the control displays once the user applies a filter.

The VisibleItemCount property returns a value as explained bellow:

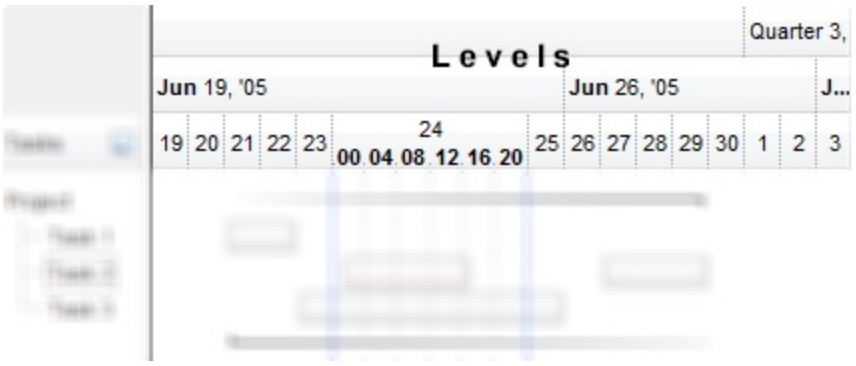
- 0, the control displays/contains no items, and no filter is applied to any column
- -1, the control displays no items, and there is a filter applied (no match found)
- positive number, indicates the number of visible items, and the control has no filter applied to any column
- negative number, the absolute value munus 1, indicates the number of visible items, and there is a filter applied (match found)

The VisibleCount property retrieves the number of items being displayed in the control's client area. Use FirstVisibleItem and NextVisibleItem properties to determine the items being displayed in the control's client area. Use the IsItemVisible property to check whether an item fits the control's client area. Use the ItemCount property to count the items in the control. Use the ChildCount property to count the child items

Level object

The Level object describes a level in the chart. Use the Chart object to access the control's Chart object. Use the Bars property to add new type of bars to the control. The levels are displayed in the chart's header area. Use the Level property to access a Level object.

The following screen shot shows the chart's levels:



The Level property supports the following properties and methods:

Name	Description
Alignment	Specifies the label's alignment.
BackColor	Specifies the level's background color.
Count	Counts the units in the level.
DrawGridLines	Specifies whether the grid lines are shown or hidden for specified level.
DrawTickLines	Specifies whether the tick lines are shown or hidden.
DrawTickLinesFrom	Indicates whether the level shows tick lines from specified level.
ForeColor	Specifies the level's foreground color.
FormatLabel	Formats the labels based on the specified formula.
GridLineColor	Specifies the grid line color for the specified level.
GridLineStyle	Specifies the style for the chart's vertical gridlines.
Label	Retrieves or sets a value that indicates the format of the level's label.
ReplaceLabel	Specifies a HTML replacement for the given label.
ToolTip	Specifies the format of the tooltip that's shown when the cursor hovers the level.
Unit	Retrieves or sets a value that indicates the unit of the level.

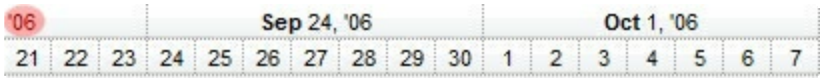
property Level.Alignment as AlignmentEnum

Specifies the label's alignment.

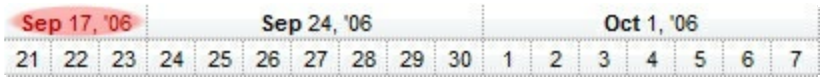
Type	Description
AlignmentEnum	An AlignmentEnum expression that indicates how the level's label is aligned in the chart's header. The Alignment property can combine the LeftAlignment, CenterAlignment and RightAlignment with exHOutside (0x10,16) which indicates that the label is always visible when user does scrolling the chart. Also, the Alignment property supports the exHNoClip (0x100, 256) which indicates that the labels within the level are not-clipped to the time-unit. In other words, it allows to prevent truncating the level's label when the width of the time-scale is too small.

By default, the Alignment property is CenterAlignment. Use the Alignment property to align labels in the chart's header. If the Alignment property includes the exHOutside, the label is being visible while the time unit is visible. For instance, if the Alignment property is CenterAlignment + exHOutside (17 = 1 + 16), the labels are always centered, and visible while the time-unit is visible, so the label is still visible while the time unit is partially visible, usually when the user does scroll left or right the chart. Use the Label property to specify the label of the level. Use the ForeColor and BackColor properties to change the level's appearance.

For instance the following screen shot shows the component if the Level.Alignment property is 1 (CenterAlignment):



while the next screen shot shows the component if the Level.Alignment property is 17 (CenterAlignment + exHOutside):

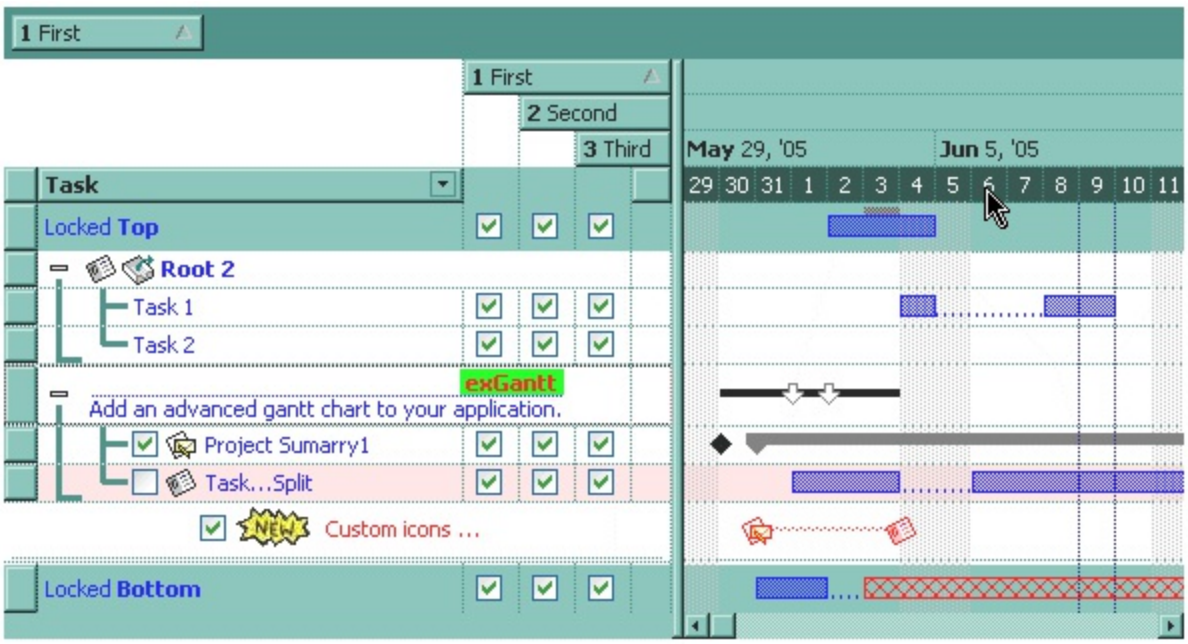


property Level.BackColor as Color

Specifies the level's background color.

Type	Description
Color	A Color expression that indicates the level's background color. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

Use the BackColor property to specify the background color for a specified level. Use the ForeColor property to specify the foreground color for a specified level. Use the BackColorLevelHeader property to specify the background color of the chart's header. Use the ForeColorLevelHeader property to specify the foreground color of the chart's header. Use the BackColor property to specify the chart's background color. Use the ForeColor property to specify the chart's foreground color. Use the ItemBackColor property to change the item's background color. Use the NonworkingDaysColor property the color of the brush to fill the nonworking days area.



The following VB sample changes the appearance for the last level:

```
With G2antt1.Chart
   With .Level(.LevelCount - 1)
      .BackColor = SystemColorConstants.vbDesktop
      .ForeColor = RGB(255, 255, 255)
```

```
    End With
End With
```

The following C++ sample changes the appearance for the last level:

```
CLevel level = m_g2antt.GetChart().GetLevel(m_g2antt.GetChart().GetLevelCount()-1);
level.SetBackColor( 0x80000000 | COLOR_DESKTOP );
level.SetForeColor( RGB(255,255,255) );
```

The following VB.NET sample changes the appearance for the last level:

```
With AxG2antt1.Chart
    With .Level(.LevelCount - 1)
        .BackColor = ToUInt32(SystemColors.Desktop)
        .ForeColor = RGB(255, 255, 255)
    End With
End With
```

where the ToUInt32 function converts a Color expression to an OLE_COLOR type:

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
    Dim i As Long
    i = c.R
    i = i + 256 * c.G
    i = i + 256 * 256 * c.B
    ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample changes the appearance for the last level:

```
EXG2ANTTLib.Level level = axG2antt1.Chart.get_Level(axG2antt1.Chart.LevelCount - 1);
level.BackColor = ToUInt32(SystemColors.Desktop);
level.ForeColor = ToUInt32(Color.FromArgb(255,255,255));
```

where the ToUInt32 function converts a Color expression to an OLE_COLOR type:

```
private UInt32 ToUInt32(Color c)
{
    long i;
    i = c.R;
```

```
    i = i + 256 * c.G;
    i = i + 256 * 256 * c.B;
    return Convert.ToUInt32(i);
}
```

The following VFP sample changes the appearance for the last level:

```
With thisform.G2antt1.Chart
    With .Level(.LevelCount - 1)
        .BackColor = 0x80000001
        .ForeColor = RGB(255, 255, 255)
    EndWith
EndWith
```

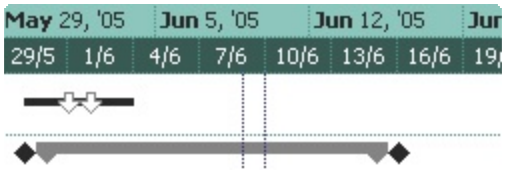
property Level.Count as Long

Counts the units in the level.

Type	Description
Long	A Long expression that indicates the number of units being displayed in the same place in the level.

By default, the Count property is 1. The Count property specifies the number of units being displayed in the level. **The Label property may change the Unit and the Count property**. The Unit property specifies the unit being used to display labels in the level. Use the Label property to assign a caption for the level. Use the NextDate property to get the next date. Use the Zoom method to zoom the chart to a specified interval of dates. Use the FormatDate property to format a date to a specified format. Use the CountVisibleUnits property and the ClientWidth property of the eXPrint component to specify that you need to display the chart on a single page. The StartPrintDate and EndPrintDate property specifies range of dates within the chart is printed.

The following screen shot shows a header that displays the dates from 3 by 3 days :

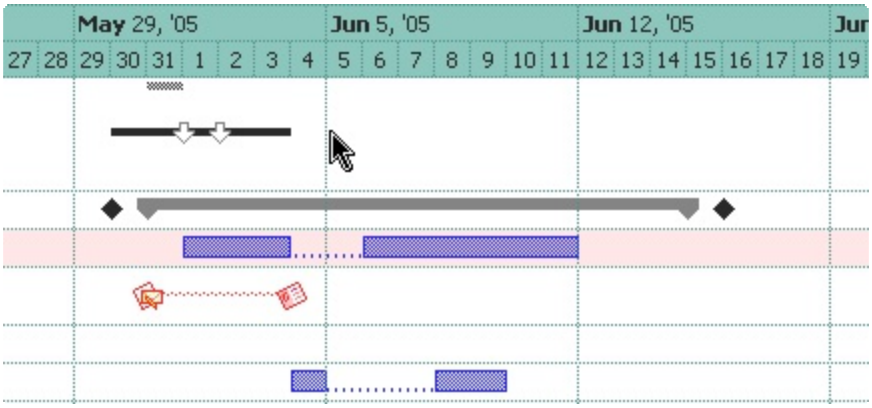


property Level.DrawGridLines as Boolean

Specifies whether the grid lines are shown or hidden for specified level.

Type	Description
Boolean	A Boolean expression that indicates whether the vertical grid lines between time units in the level are visible or hidden.

By default, the DrawGridLines property is False. Use the DrawGridLines property to specify whether the control draws the grid lines in the chart's area. The GridLineStyle property to specify the style for horizontal or/and vertical gridlines in the level view. Use the GridLineColor property to specify the color for the vertical grid lines between time units. The DrawGridLines property draws the vertical grid lines only if the DrawGridLines property of the Chart object is exVLines, exRowLines or exAllLines. If the DrawGridLines property is exNoLines, exHLines, the DrawGridLines property has no effect. Use the MarkTodayColor property to specify the color to mark the today date. Use the NonworkingDays property to specify the nonworking days. Use the NonworkingDaysPattern property to specify the brush to fill the nonworking days area. Use the DrawTickLines property to specify whether the grid lines between time units in the level are visible or hidden. The OverviewLevelLines property indicates the index of the level that displays the grid line in the chart's overview. Use the AdjustLevelsToBase property in case you are using a not-contiguous time scale, so the tick lines are not properly arranged.



Your application can provide some options to help user while performing moving or resizing the bars at runtime as follow:

- grid lines, that can be shown only when moving or resizing, using the ChartStartChanging and ChartEndChanging events
- select date, to specify the margins of the area you want to highlight
- ticker, that shows the cursor's position in the chart, or while resizing, it shows the size and the position of the bar
- ability to specify a resizing/moving unit, different that the displayed one ie while the chart displays days, you can specify the resizing unit on hours.

- inside zoom, that can be used to magnify the portion of the chart being selected

property Level.DrawTickLines as LevelLineEnum

Specifies whether the tick lines are shown or hidden.

Type	Description
LevelLineEnum	A LevelLineEnum expression that specifies the type of line to divide the time units in the level.

By default, the DrawTickLines property is exLevelDefaultLine (dotted line). The DrawTickLines / DrawTickLinesFrom property always draw the vertically lines in the level, while the DrawLevelSeparator property draws the horizontally lines in the level. Use the DrawTickLinesFrom method to show an alternative tick lines based on the time scale units of the another level. Use the DrawGridLines property to specify whether the control draws the grid lines in the chart's area. Use the GridLineColor property to specify the color for grid lines. Use the DrawGridLines property to draw grid lines for a specified level. Use the DrawLevelSeperator property to draw lines between levels inside the chart's header. Use the MarkTodayColor property to specify the color to mark the today date. Use the AdjustLevelsToBase property in case you are using a not-contiguous time scale, so the tick lines are not properly arranged.

method Level.DrawTickLinesFrom (Level as Long, Type as LevelLineEnum)

Indicates whether the level shows tick lines from specified level.

Type	Description
Level as Long	A long expression that specifies the index of the level that specifies the new base scale unit to show the tick lines
Type as LevelLineEnum	A LevelLineEnum expression that specifies the tick lines being shown.

Use the DrawTickLinesFrom method to show the tick lines based on the scale of another level. Use the DrawTickLinesFrom method when you need to display multiple tick lines based on the different levels. The DrawTickLines property specifies the style of lines being shown between time units of the level. The DrawTickLines / DrawTickLinesFrom property always draw the vertically lines in the level, while the DrawLevelSeparator property draws the horizontally lines in the level. Use the AdjustLevelsToBase property in case you are using a not-contiguous time scale, so the tick lines are not properly arranged.

The following screen shot shows on the first level days being separated with exLevelLowerHalf + exLevelSolidLine, while the second level displays hours exLevelMiddleLine Or exLevelDotLine, and call the .DrawTickLinesFrom 0, exLevelSolidLine, which means that the vertically solid lines from the second header are actually dictated by the first level, while the rest of units displays exLevelMiddleLine Or exLevelDotLine verticaly lines.

The following VB sample displays the tick lines from first level to second level:

```
With G2antt1
  .BeginUpdate
  With .Chart
    .DrawLevelSeparator = exLevelNoLine
    .UnitWidth = 24
    .FirstVisibleDate = #1/1/2001#
    .PaneWidth(0) = 0
    .LevelCount = 2
    With .Level(0)
      .Alignment = CenterAlignment
      .Label = "<%dddd%>"
```

```
        .DrawTickLines = 18
      End With
      With .Level(1)
        .Label = 65536
        .Count = 6
        .DrawTickLines = 66
        .DrawTickLinesFrom 0,exLevelSolidLine
      End With
    End With
    .EndUpdate
End With
```

The following VB.NET sample displays the tick lines from first level to second level:

```
With AxG2antt1
    .BeginUpdate
    With .Chart
      .DrawLevelSeparator = EXG2ANTTLib.LevelLineEnum.exLevelNoLine
      .UnitWidth = 24
      .FirstVisibleDate = #1/1/2001#
      .PaneWidth(0) = 0
      .LevelCount = 2
      With .Level(0)
        .Alignment = EXG2ANTTLib.AlignmentEnum.CenterAlignment
        .Label = "<%dddd%>"
        .DrawTickLines = 18
      End With
      With .Level(1)
        .Label = 65536
        .Count = 6
        .DrawTickLines = 66
        .DrawTickLinesFrom 0,EXG2ANTTLib.LevelLineEnum.exLevelSolidLine
      End With
    End With
    .EndUpdate
End With
```

The following C++ sample displays the tick lines from first level to second level:

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutDrawLevelSeparator(EXG2ANTTLib::exLevelNoLine);
   var_Chart->PutUnitWidth(24);
   var_Chart->PutFirstVisibleDate("1/1/2001");
   var_Chart->PutPaneWidth(0,0);
   var_Chart->PutLevelCount(2);
   EXG2ANTTLib::ILevelPtr var_Level = var_Chart->GetLevel(0);
      var_Level->PutAlignment(EXG2ANTTLib::CenterAlignment);
      var_Level->PutLabel("<%dddd%>");
      var_Level->PutDrawTickLines((EXG2ANTTLib::LevelLineEnum)18);
   EXG2ANTTLib::ILevelPtr var_Level1 = var_Chart->GetLevel(1);
      var_Level1->PutLabel(long(65536));
      var_Level1->PutCount(6);
      var_Level1->PutDrawTickLines((EXG2ANTTLib::LevelLineEnum)66);
      var_Level1->DrawTickLinesFrom(0,EXG2ANTTLib::exLevelSolidLine);
spG2antt1->EndUpdate();
```

The following C# sample displays the tick lines from first level to second level:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.DrawLevelSeparator = EXG2ANTTLib.LevelLineEnum.exLevelNoLine;
   var_Chart.UnitWidth = 24;
   var_Chart.FirstVisibleDate = "1/1/2001";
   var_Chart.set_PaneWidth(0 != 0,0);
   var_Chart.LevelCount = 2;
   EXG2ANTTLib.Level var_Level = var_Chart.get_Level(0);
```

```
    var_Level.Alignment = EXG2ANTTLib.AlignmentEnum.CenterAlignment;
    var_Level.Label = "<%dddd%>";
    var_Level.DrawTickLines = (EXG2ANTTLib.LevelLineEnum)18;
  EXG2ANTTLib.Level var_Level1 = var_Chart.get_Level(1);
    var_Level1.Label = 65536;
    var_Level1.Count = 6;
    var_Level1.DrawTickLines = (EXG2ANTTLib.LevelLineEnum)66;
    var_Level1.DrawTickLinesFrom(0,EXG2ANTTLib.LevelLineEnum.exLevelSolidLine);
axG2antt1.EndUpdate();
```

The following VFP sample displays the tick lines from first level to second level:

```
with thisform.G2antt1
  .BeginUpdate
  with .Chart
    .DrawLevelSeparator = 0
    .UnitWidth = 24
    .FirstVisibleDate = {^2001-1-1}
    .PaneWidth(0) = 0
    .LevelCount = 2
    with .Level(0)
      .Alignment = 1
      .Label = "<%dddd%>"
      .DrawTickLines = 18
    endwith
    with .Level(1)
      .Label = 65536
      .Count = 6
      .DrawTickLines = 66
      .DrawTickLinesFrom(0,2)
    endwith
  endwith
  .EndUpdate
endwith
```

The following Delphi sample displays the tick lines from first level to second level:

```
with AxG2antt1 do
```

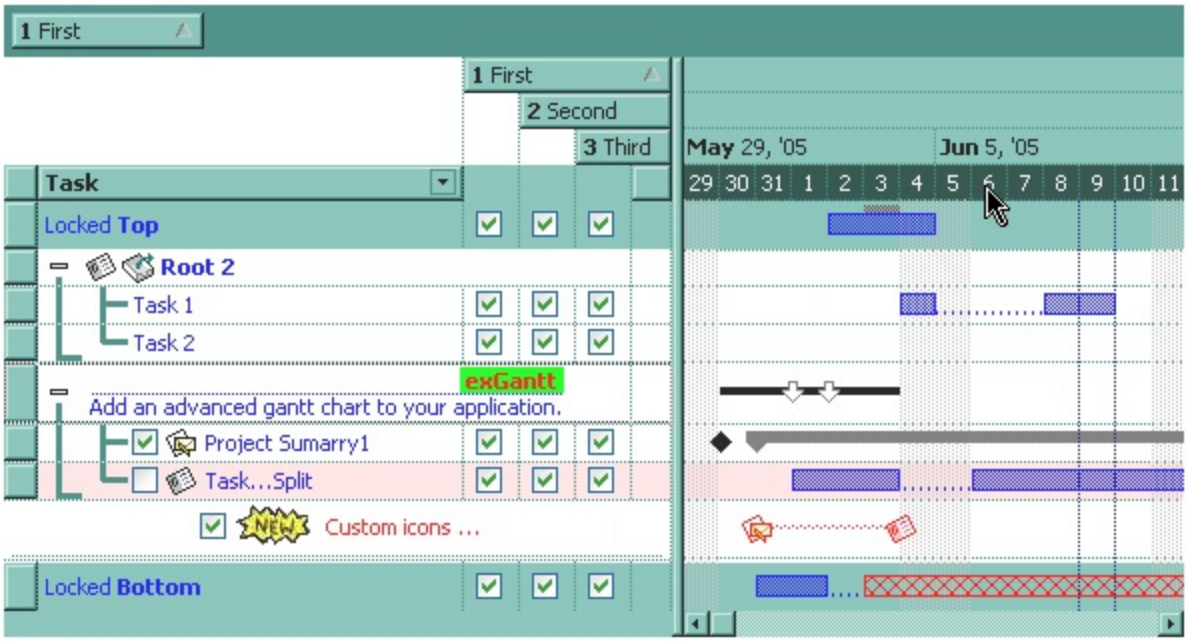
```
begin
   BeginUpdate();
   with Chart do
   begin
      DrawLevelSeparator := EXG2ANTTLib.LevelLineEnum.exLevelNoLine;
      UnitWidth := 24;
      FirstVisibleDate := '1/1/2001';
      PaneWidth[0 <> 0] := 0;
      LevelCount := 2;
      with Level[0] do
      begin
         Alignment := EXG2ANTTLib.AlignmentEnum.CenterAlignment;
         Label := '<%dddd%>';
         DrawTickLines := EXG2ANTTLib.LevelLineEnum(18);
      end;
      with Level[1] do
      begin
         Label := TObject(65536);
         Count := 6;
         DrawTickLines := EXG2ANTTLib.LevelLineEnum(66);
         DrawTickLinesFrom(0,EXG2ANTTLib.LevelLineEnum.exLevelSolidLine);
      end;
   end;
   EndUpdate();
end
```

property Level.ForeColor as Color

Specifies the level's foreground color.

Type	Description
Color	A Color expression that indicates the level's foreground color.

Use the ForeColor property to specify the foreground color for a specified level. Use the BackColor property to specify the background color for a specified level. Use the BackColorLevelHeader property to specify the background color of the chart's header. Use the ForeColorLevelHeader property to specify the foreground color of the chart's header. Use the BackColor property to specify the chart's background color. Use the ForeColor property to specify the chart's foreground color. Use the ItemBackColor property to change the item's background color. Use the NonworkingDaysColor property the color of the brush to fill the nonworking days area.



The following VB sample changes the appearance for the last level:

```
With G2antt1.Chart
   With .Level(.LevelCount - 1)
      .BackColor = SystemColorConstants.vbDesktop
      .ForeColor = RGB(255, 255, 255)
   End With
End With
```

The following C++ sample changes the appearance for the last level:

```
CLevel level = m_g2antt.GetChart().GetLevel(m_g2antt.GetChart().GetLevelCount()-1);
level.SetBackColor( 0x80000000 | COLOR_DESKTOP );
level.SetForeColor( RGB(255,255,255) );
```

The following VB.NET sample changes the appearance for the last level:

```
With AxG2antt1.Chart
   With .Level(.LevelCount - 1)
      .BackColor = ToUInt32(SystemColors.Desktop)
      .ForeColor = RGB(255, 255, 255)
   End With
End With
```

where the ToUInt32 function converts a Color expression to an OLE_COLOR type:

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
   Dim i As Long
   i = c.R
   i = i + 256 * c.G
   i = i + 256 * 256 * c.B
   ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample changes the appearance for the last level:

```
EXG2ANTTLib.Level level = axG2antt1.Chart.get_Level(axG2antt1.Chart.LevelCount - 1);
level.BackColor = ToUInt32(SystemColors.Desktop);
level.ForeColor = ToUInt32(Color.FromArgb(255,255,255));
```

where the ToUInt32 function converts a Color expression to an OLE_COLOR type:

```
private UInt32 ToUInt32(Color c)
{
   long i;
   i = c.R;
   i = i + 256 * c.G;
   i = i + 256 * 256 * c.B;
   return Convert.ToUInt32(i);
}
```

The following VFP sample changes the appearance for the last level:

```
With thisform.G2antt1.Chart
   With .Level(.LevelCount - 1)
      .BackColor = 0x80000001
      .ForeColor = RGB(255, 255, 255)
   EndWith
EndWith
```

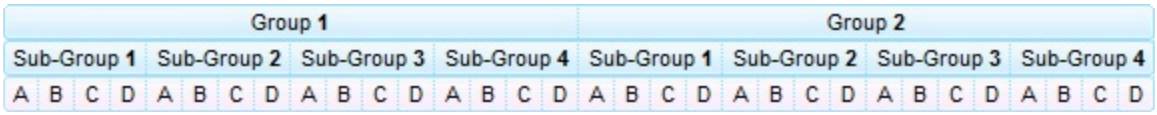
property Level.FormatLabel as String

Formats the labels based on the specified formula.

Type	Description
String	A String expression that specifies the formula to format the level.

By default, the FormatLabel property is Empty. The FormatLabel property has effect only if it is valid and not empty. The Label property defines the label being shown in the chart. Use the FormatLabel property to customize the labels being displayed in the chart's level. The **value** keyword in the FormatLabel property specifies the label of the level as it is before formatting (in string format). The **dvalue** keyword, indicates the date-time expression of the element in the label being formatted. Also, you can use the ReplaceLabel property to replace a specified label.

- For instance, you can use the FormatLabel property to customize the labels and the subdivisions in your chart as you can see in the following screen shot ("*'Group <b>'+int(1 +dvalue/16)*"):

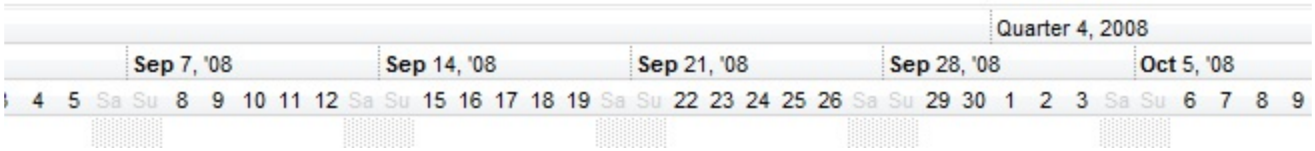


If the Unit property is exDay for several levels (more than 1 level), you can display your subdivisions using the Count property, based on 0, not based on the FirstVisibleDate property. You can get this layout using the

- Other sample would be, if you need to display the last Friday of each month using a different color, so in this case you can use the FormatLevel property as "*(weekday(dvalue)=5 ? month(dvalue+7)!=month(dvalue) ? '<b><bgcolor=000000><fgcolor=FFFFFF>') + value*" as in the following screen shot:

The screen shows in different colors the Jun 27 as being the last Friday of June, and the Jul 25 as being the last Friday on July.

- The following screen shot shows the week end days using a different foreground color (FormatLabel property is "*weekday(dvalue)=6 ? '<fgcolor=D0D0D0>Sa' : (weekday(dvalue)=0 ? '<fgcolor=D0D0D0>Su' : value)*")



The **value** keyword in the FormatLabel property specifies the label of the level as it is before formatting (in string format). The **dvalue** keyword, indicates the date-time expression of the element in the label being formatted.

This property/method supports predefined constants and operators/functions as described here.

The following VB sample shows how you can highlight the last Friday for each month::

```
With G2antt1
   With .Chart
      .PaneWidth(0) = 0
      .FirstVisibleDate = #1/17/2008#
      .LevelCount = 2
      .Level(1).FormatLabel = "(weekday(dvalue)=5 ? month(dvalue+7)!=month(dvalue) ?
'<b><bgcolor=000000><fgcolor=FFFFFF>' ) +" & _
" value"
   End With
End With
```

The following VB sample shows how you can define your own labels and subdivisions:

```
With G2antt1
   .BeginUpdate
   With .Chart
      .ToolTip = ""
      .PaneWidth(0) = 0
      .ScrollRange(exStartDate) = 0
      .ScrollRange(exEndDate) = 110
      .FirstVisibleDate = 0
      .ShowNonworkingDates = False
      .MarkTodayColor = .BackColor
      .LevelCount = 3
      With .Level(0)
         .ToolTip = ""
         .Alignment = CenterAlignment
```

```
        .Unit = exDay
        .Count = 16
        .FormatLabel = "'Group <b>'+int(1 +dvalue/16)"
      End With
      With .Level(1)
        .ToolTip = ""
        .Alignment = CenterAlignment
        .Unit = exDay
        .Count = 4
        .FormatLabel = " (abs(dvalue)/4) mod 4"
        .ReplaceLabel("0") = "Sub-Group <b>1</b>"
        .ReplaceLabel("1") = "Sub-Group <b>2</b>"
        .ReplaceLabel("2") = "Sub-Group <b>3</b>"
        .ReplaceLabel("3") = "Sub-Group <b>4</b>"
      End With
      With .Level(2)
        .ToolTip = ""
        .Unit = exDay
        .Count = 1
        .FormatLabel = "(abs(dvalue) mod 4)"
        .ReplaceLabel("0") = "A"
        .ReplaceLabel("1") = "B"
        .ReplaceLabel("2") = "C"
        .ReplaceLabel("3") = "D"
      End With
    End With
    .EndUpdate
End With
```

The following VB.NET sample shows how you can highlight the last Friday for each month::

```
With AxG2antt1
    With .Chart
        .PaneWidth(0) = 0
        .FirstVisibleDate = #1/17/2008#
        .LevelCount = 2
        .Level(1).FormatLabel = "(weekday(dvalue)=5 ? month(dvalue+7)!=month(dvalue) ?
```

```
'<b><bgcolor=000000><fgcolor=FFFFFF>' ) +" & _
" value"
   End With
End With
```

The following VB.NET sample shows how you can define your own labels and subdivisions:

```
With AxG2antt1
   .BeginUpdate
   With .Chart
      .ToolTip = ""
      .PaneWidth(0) = 0
      .ScrollRange(EXG2ANTTLib.ScrollRangeEnum.exStartDate) = 0
      .ScrollRange(EXG2ANTTLib.ScrollRangeEnum.exEndDate) = 110
      .FirstVisibleDate = 0
      .ShowNonworkingDates = False
      .MarkTodayColor = .BackColor
      .LevelCount = 3
      With .Level(0)
         .ToolTip = ""
         .Alignment = EXG2ANTTLib.AlignmentEnum.CenterAlignment
         .Unit = EXG2ANTTLib.UnitEnum.exDay
         .Count = 16
         .FormatLabel = "'Group <b>'+int(1 +dvalue/16)"
      End With
      With .Level(1)
         .ToolTip = ""
         .Alignment = EXG2ANTTLib.AlignmentEnum.CenterAlignment
         .Unit = EXG2ANTTLib.UnitEnum.exDay
         .Count = 4
         .FormatLabel = " (abs(dvalue)/4) mod 4"
         .ReplaceLabel("0") = "Sub-Group <b>1</b>"
         .ReplaceLabel("1") = "Sub-Group <b>2</b>"
         .ReplaceLabel("2") = "Sub-Group <b>3</b>"
         .ReplaceLabel("3") = "Sub-Group <b>4</b>"
      End With
      With .Level(2)
```

```
        .ToolTip = ""
        .Unit = EXG2ANTTLib.UnitEnum.exDay
        .Count = 1
        .FormatLabel = "(abs(dvalue) mod 4)"
        .ReplaceLabel("0") = "A"
        .ReplaceLabel("1") = "B"
        .ReplaceLabel("2") = "C"
        .ReplaceLabel("3") = "D"
      End With
   End With
   .EndUpdate
End With
```

The following C++ sample shows how you can highlight the last Friday for each month::

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'

   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutPaneWidth(0,0);
   var_Chart->PutFirstVisibleDate("1/17/2008");
   var_Chart->PutLevelCount(2);
   var_Chart->GetLevel(1)->PutFormatLabel(_bstr_t("(weekday(dvalue)=5 ?
month(dvalue+7)!=month(dvalue) ? '<b><bgcolor=000000><fgcolor=FFFFFF>' ) +") +
" value");
```

The following C++ sample shows how you can define your own labels and subdivisions:

```
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'EXG2ANTTLib' for the library: 'ExG2antt 1.0 Control Library'
```

```
   #import <ExG2antt.dll>
   using namespace EXG2ANTTLib;
*/
EXG2ANTTLib::IG2anttPtr spG2antt1 = GetDlgItem(IDC_G2ANTT1)-
>GetControlUnknown();
spG2antt1->BeginUpdate();
EXG2ANTTLib::IChartPtr var_Chart = spG2antt1->GetChart();
   var_Chart->PutToolTip(L"");
   var_Chart->PutPaneWidth(0,0);
   var_Chart->PutScrollRange(EXG2ANTTLib::exStartDate,long(0));
   var_Chart->PutScrollRange(EXG2ANTTLib::exEndDate,long(110));
   var_Chart->PutFirstVisibleDate(long(0));
   var_Chart->PutShowNonworkingDates(VARIANT_FALSE);
   var_Chart->PutMarkTodayColor(var_Chart->GetBackColor());
   var_Chart->PutLevelCount(3);
   EXG2ANTTLib::ILevelPtr var_Level = var_Chart->GetLevel(0);
      var_Level->PutToolTip("");
      var_Level->PutAlignment(EXG2ANTTLib::CenterAlignment);
      var_Level->PutUnit(EXG2ANTTLib::exDay);
      var_Level->PutCount(16);
      var_Level->PutFormatLabel(L"'Group <b>'+int(1 +dvalue/16)");
   EXG2ANTTLib::ILevelPtr var_Level1 = var_Chart->GetLevel(1);
      var_Level1->PutToolTip("");
      var_Level1->PutAlignment(EXG2ANTTLib::CenterAlignment);
      var_Level1->PutUnit(EXG2ANTTLib::exDay);
      var_Level1->PutCount(4);
      var_Level1->PutFormatLabel(L" (abs(dvalue)/4) mod 4");
      var_Level1->PutReplaceLabel(L"0",L"Sub-Group <b>1</b>");
      var_Level1->PutReplaceLabel(L"1",L"Sub-Group <b>2</b>");
      var_Level1->PutReplaceLabel(L"2",L"Sub-Group <b>3</b>");
      var_Level1->PutReplaceLabel(L"3",L"Sub-Group <b>4</b>");
   EXG2ANTTLib::ILevelPtr var_Level2 = var_Chart->GetLevel(2);
      var_Level2->PutToolTip("");
      var_Level2->PutUnit(EXG2ANTTLib::exDay);
      var_Level2->PutCount(1);
      var_Level2->PutFormatLabel(L"(abs(dvalue) mod 4)");
      var_Level2->PutReplaceLabel(L"0",L"A");
```

```
    var_Level2->PutReplaceLabel(L"1",L"B");
    var_Level2->PutReplaceLabel(L"2",L"C");
    var_Level2->PutReplaceLabel(L"3",L"D");
spG2antt1->EndUpdate();
```

The following C# sample shows how you can highlight the last Friday for each month::

```
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.set_PaneWidth(0 != 0,0);
   var_Chart.FirstVisibleDate = "1/17/2008";
   var_Chart.LevelCount = 2;
   var_Chart.get_Level(1).FormatLabel = "(weekday(dvalue)=5 ?
month(dvalue+7)!=month(dvalue) ? '<b><bgcolor=000000><fgcolor=FFFFFF>' ) +" +
" value";
```

The following C# sample shows how you can define your own labels and subdivisions:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Chart var_Chart = axG2antt1.Chart;
   var_Chart.ToolTip = "";
   var_Chart.set_PaneWidth(0 != 0,0);
   var_Chart.set_ScrollRange(EXG2ANTTLib.ScrollRangeEnum.exStartDate,0);
   var_Chart.set_ScrollRange(EXG2ANTTLib.ScrollRangeEnum.exEndDate,110);
   var_Chart.FirstVisibleDate = 0;
   var_Chart.ShowNonworkingDates = false;
   var_Chart.MarkTodayColor = var_Chart.BackColor;
   var_Chart.LevelCount = 3;
   EXG2ANTTLib.Level var_Level = var_Chart.get_Level(0);
      var_Level.ToolTip = "";
      var_Level.Alignment = EXG2ANTTLib.AlignmentEnum.CenterAlignment;
      var_Level.Unit = EXG2ANTTLib.UnitEnum.exDay;
      var_Level.Count = 16;
      var_Level.FormatLabel = "'Group <b>'+int(1 +dvalue/16)";
   EXG2ANTTLib.Level var_Level1 = var_Chart.get_Level(1);
      var_Level1.ToolTip = "";
      var_Level1.Alignment = EXG2ANTTLib.AlignmentEnum.CenterAlignment;
      var_Level1.Unit = EXG2ANTTLib.UnitEnum.exDay;
      var_Level1.Count = 4;
```

```
    var_Level1.FormatLabel = " (abs(dvalue)/4) mod 4";
    var_Level1.set_ReplaceLabel("0","Sub-Group <b>1</b>");
    var_Level1.set_ReplaceLabel("1","Sub-Group <b>2</b>");
    var_Level1.set_ReplaceLabel("2","Sub-Group <b>3</b>");
    var_Level1.set_ReplaceLabel("3","Sub-Group <b>4</b>");
  EXG2ANTTLib.Level var_Level2 = var_Chart.get_Level(2);
    var_Level2.ToolTip = "";
    var_Level2.Unit = EXG2ANTTLib.UnitEnum.exDay;
    var_Level2.Count = 1;
    var_Level2.FormatLabel = "(abs(dvalue) mod 4)";
    var_Level2.set_ReplaceLabel("0","A");
    var_Level2.set_ReplaceLabel("1","B");
    var_Level2.set_ReplaceLabel("2","C");
    var_Level2.set_ReplaceLabel("3","D");
axG2antt1.EndUpdate();
```

The following VFP sample shows how you can highlight the last Friday for each month::

```
with thisform.G2antt1
  with .Chart
    .PaneWidth(0) = 0
    .FirstVisibleDate = {^2008-1-17}
    .LevelCount = 2
    var_s = "(weekday(dvalue)=5 ? month(dvalue+7)!=month(dvalue) ? '<b>
<bgcolor=000000><fgcolor=FFFFFF>' ) + "
    var_s = var_s + "value"
    .Level(1).FormatLabel = var_s
  endwith
endwith
```

The following VFP sample shows how you can define your own labels and subdivisions:

```
with thisform.G2antt1
  .BeginUpdate
  with .Chart
    .ToolTip = ""
    .PaneWidth(0) = 0
    .ScrollRange(0) = 0
```

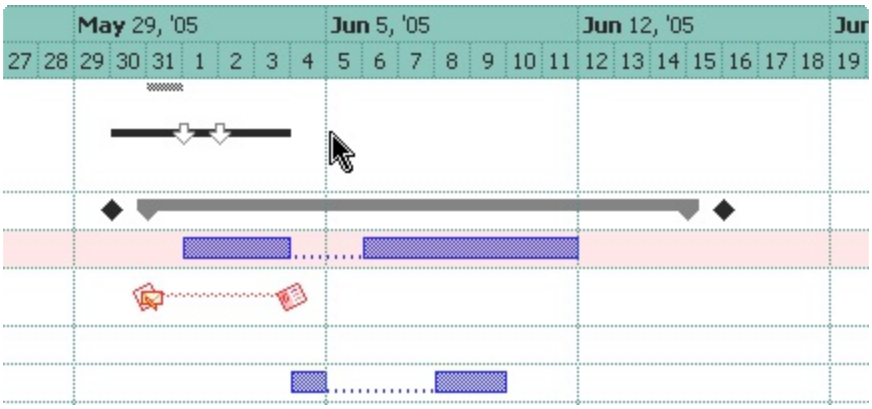
```
    .ScrollRange(1) = 110
    .FirstVisibleDate = 0
    .ShowNonworkingDates = .F.
    .MarkTodayColor = .BackColor
    .LevelCount = 3
    with .Level(0)
      .ToolTip = ""
      .Alignment = 1
      .Unit = 4096
      .Count = 16
      .FormatLabel = "'Group <b>'+int(1 +dvalue/16)"
    endwith
    with .Level(1)
      .ToolTip = ""
      .Alignment = 1
      .Unit = 4096
      .Count = 4
      .FormatLabel = " (abs(dvalue)/4) mod 4"
      .ReplaceLabel("0") = "Sub-Group <b>1</b>"
      .ReplaceLabel("1") = "Sub-Group <b>2</b>"
      .ReplaceLabel("2") = "Sub-Group <b>3</b>"
      .ReplaceLabel("3") = "Sub-Group <b>4</b>"
    endwith
    with .Level(2)
      .ToolTip = ""
      .Unit = 4096
      .Count = 1
      .FormatLabel = "(abs(dvalue) mod 4)"
      .ReplaceLabel("0") = "A"
      .ReplaceLabel("1") = "B"
      .ReplaceLabel("2") = "C"
      .ReplaceLabel("3") = "D"
    endwith
  endwith
  .EndUpdate
endwith
```

property Level.GridLineColor as Color

Specifies the grid line color for the specified level.

Type	Description
Color	A Color expression that indicates the color of the vertical grid lines in the chart area.

Use the GridLineColor property to specify the color for the vertical grid lines between time units. Use the DrawGridLines property to specify whether the control draws the grid lines in the chart's area. The DrawGridLines property draws the vertical grid lines only if the DrawGridLines property of the Chart object is exVLines, exRowLines or exAllLines. If the DrawGridLines property is exNoLines, exHLines, the DrawGridLines property has no effect. Use the MarkTodayColor property to specify the color to mark the today date.



property Level.GridLineStyle as GridLinesStyleEnum

Specifies the style for the chart's vertical gridlines.

Type	Description
GridLinesStyleEnum	A GridLinesStyleEnum expression that specifies the style to show the chart's vertical gridlines.

By default, the GridLineStyle property is exGridLinesDot. The GridLineStyle property has effect only if the chart's DrawGridLines property is not zero and one of the level's DrawGridLines property is True. Use the GridLineColor property to specify the color for vertical grid lines. Use the DrawTickLines property to specify whether the grid lines between time units in the level are visible or hidden

property Level.Label as Variant

Retrieves or sets a value that indicates the format of the level's label.

Type	Description
Variant	A String expression that indicates the format of the level's label, an UnitEnum expression that indicates the predefined format being used. The Label property defines predefined formats for labales.

The Label property defines the HTML labels being displayed on the chart's header. Use the Alignment property to specify the label's alignment. Use the Alignment property on exHOutside to prevent hiding the level's label while the user scrolls left or right the chart. Use the ToolTip property to specify the tooltip being displayed when the cursor hovers the level. Use the BackColor and ForeColor properties to change the level's appearance. The WeekDays property retrieves or sets a value that indicates the list of names for each week day, separated by space. Use the MonthNames property to specify the name of the months in the year. The FormatDate property formats a date. Use the ReplaceLabel property to customize the labels as adding icons/images/pictures or change the captions being displayed by default in the chart's header. Valid date values range from January 1, 100 A.D. (-647434) to December 31, 9999 A.D. (2958465). A date value of 0 represents December 30, 1899. Use the FormatLabel property to format the label giving a formula.

The Label property supports alternative HTML labels being separated by "<|>" and values for Count and Unit being separated by "<||>". By alternate HTML label we mean that you can define a list of HTML labels that may be displayed in the chart's header based on the space allocated for the time-unit. In other words, the control chooses automatically the alternate HTML label to be displayed for best fitting in the portion of the chart where the time-unit should be shown.

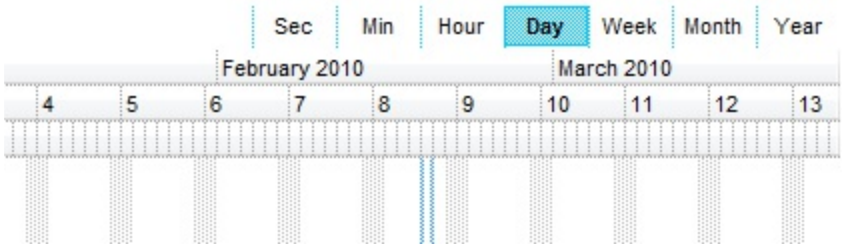
The Label property format is "**ALT1[<|>ALT2<|>...[<||>COUNT[<||>UNIT]]]**" where

- ALT defines a HTML label
- COUNT specifies the value for the Count property
- UNIT field indicates the value for the Unit property
- and the parts delimited by [] brackets may miss.

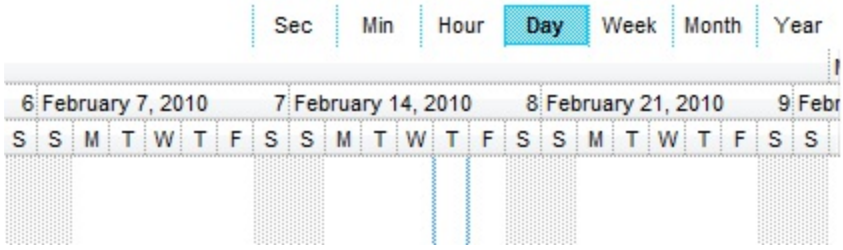
The Label property may change the Unit and the Count property. You can always use a different Unit or Count by setting the property after setting the Label property.

The following screen shots shows the chart's header using different values for UnitWidth property.

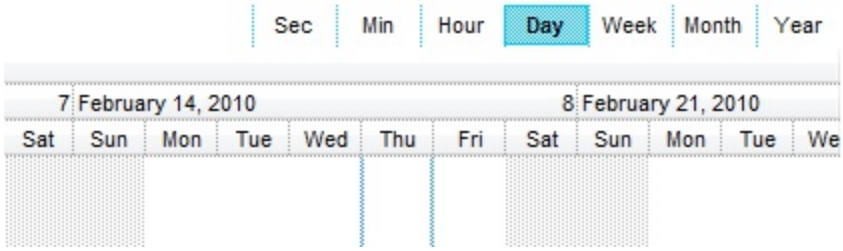
- The UnitWidth property is 6 pixels, so the base level displays nothing.



- The UnitWidth property is 18 pixels, so the base level displays the first letter of the weekday (S - S)



- The UnitWidth property is 36 pixels, so the base level displays the first 3 letters of the weekday (Sun - Sat)



For instance, Label = "<|><%d1%><|><%d2%><|><%d3%><|><%dddd%><|><%d3%>, <%m3%> <%d%>, '<%yy%><|><%dddd%>, <%mmmm%> <%d%>, <%yyyy%> <||>1<||>4096" indicates a list of 7 alternate HTML labels, the Count property set on 1 and the Unit property set on exDay (4096).

So, the header of the level in the chart shows one of the following alternate HTML labels:

- - displays nothing, if the space is less than 6 pixels.
- <%d1%> - First letter of the weekday (S to S)
- <%d2%> - First two letters of the weekday (Su to Sa)
- <%d3%> - First three letters of the weekday (Sun to Sat)
- <%dddd%> - Full name of the weekday (Sunday to Saturday)
- <%d3%>, <%m3%> <%d%>, '<%yy%> -
- <%dddd%>, <%mmmm%> <%d%>, <%yyyy%>

based on the space being allocated for the time unit. If the label is being shown on the base level, the UnitWidth property defines the space for the time-unit, so the control chooses the alternate HTML label which best fits the allocated space (width). The Font property defines the font to show the chart's labels which is also used to get the best fit label to be displayed. For any other level, the space is automatically calculated based on the base

level's width. In other words, when UnitWidth property is changed or the user rescale or zoom the chart area, the chart's header displays alternate labels. If the Label property defines no alternate labels, the single representation is shown no matter of the UnitWidth, Font and other zooming settings.

The Label property may change the Unit property as in the following scenario. Let's say that you need to display the weeks so you choose to have the week number "<%ww%>" or the first day in the week in format "<%d3%>, <%m3%> <%d%>, '<%yy%>" so the Label property should be "*<%ww%><|><%d3%>, <%m3%> <%d%>, '<%yy%>*". If you are using this format, the Unit property will always be set on exDay, as in the second alternate label the unit is day as the minimum scale unit being found is <%d3%> or <%d%> which indicates days. In order to correct this, you should specify the Unit to be used for the alternate labels as "*<%ww%><|><%d3%>, <%m3%> <%d%>, '<%yy%>***<||><||>256***".

For instance, if a level should display 15 to 15 minutes, you can do one of the following:

- call the Label = "<%**nn**%>" and after call the Count = 15.
- call the Label = "<%**nn**%><||>15", which means that the level displays minutes, and the Count property is automatically set on 15.

Any of these statements can be used to let the level displays minutes from 15 to 15.

The Label property supports the following built-in tags:

- <%**d**%> - Day of the month in one or two numeric digits, as needed (1 to 31).
- <%**dd**%> - Day of the month in two numeric digits (01 to 31).
- <%**d1**%> - First letter of the weekday (S to S). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**d2**%> - First two letters of the weekday (Su to Sa). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**d3**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**ddd**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays property to specify the name of the days in the week). You can use the <%**loc_ddd**%> that indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%**loc_ddd**%> - Indicates the day of week as a three-letter abbreviation using the current user regional and language settings.

- <%**dddd**%> - Full name of the weekday (Sunday to Saturday). (Use the WeekDays property to specify the name of the days in the week). You can use the <%**loc_dddd**%> that indicates day of week as its full name using the current user regional and language settings.
- <%**loc_dddd**%> - Indicates day of week as its full name using the current user regional and language settings.
- <%**i**%> - Displays the number instead the date. For instance, you can display numbers as 1000, 1001, 1002, 1003, instead dates. (the valid range is from -647,434 to 2,958,465)
- <%**w**%> - Day of the week (1 to 7).
- <%**ww**%> - Week of the year (1 to 53).
- <%**m**%> - Month of the year in one or two numeric digits, as needed (1 to 12).
- <%**mr**%> - Month of the year in Roman numerals, as needed (I to XII).
- <%**mm**%> - Month of the year in two numeric digits (01 to 12).
- <%**m1**%> - First letter of the month (J to D). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**m2**%> - First two letters of the month (Ja to De). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**m3**%> - First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**mmm**%> - First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmm**%> that indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%**loc_mmm**%> - Indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%**mmmm**%> - Full name of the month (January to December). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmmm**%> that indicates month as its full name using the current user regional and language settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user regional and language settings.
- <%**q**%> - Date displayed as the quarter of the year (1 to 4).
- <%**y**%> - Number of the day of the year (1 to 366).
- <%**yy**%> - Last two digits of the year (01 to 99).
- <%**yyyy**%> - Full year (0100 to 9999).
- <%**hy**%> - Date displayed as the half of the year (1 to 2).

- <%**loc_gg**%> - Indicates period/era using the current user regional and language settings.
- <%**loc_sdate**%> - Indicates the date in the short format using the current user regional and language settings.
- <%**loc_ldate**%> - Indicates the date in the long format using the current user regional and language settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user regional and language settings (/).
- <%**h**%> - Hour in one or two digits, as needed (0 to 23).
- <%**hh**%> - Hour in two digits (00 to 23).
- <%**h12**%> - Hour in 12-hour time format, in one or two digits - [0(12),11]
- <%**hh12**%> - hour in 12-hour time format, in two digits - [00(12),11]
- <%**n**%> - Minute in one or two digits, as needed (0 to 59).
- <%**nn**%> - Minute in two digits (00 to 59).
- <%**s**%> - Second in one or two digits, as needed (0 to 59).
- <%**ss**%> - Second in two digits (00 to 59).
- <%**AM/PM**%> - Twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate. (Use the AMPM property to specify the name of the AM and PM indicators). You can use the <%**loc_AM/PM**%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%**loc_A/P**%> that indicates the one character time marker such as A or P using the current user regional and language settings
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user regional and language settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user regional and language settings.
- <%**loc_time**%> - Indicates the time using the current user regional and language settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%**loc_tsep**%> - indicates the time separator using the current user regional and language settings (:)
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits,

for example, "13". No additional leading zeros are displayed

The following tags are displayed based on the user's Regional and Language Options:

- <%**loc_sdate**%> - Indicates the date in the short format using the current user settings.
- <%**loc_ldate**%> - Indicates the date in the long format using the current user settings.
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**loc_ddd**%> - Indicates day of week as a three-letters abbreviation using the current user settings.
- <%**loc_dddd**%> - Indicates day of week as its full name using the current user settings.
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**loc_mmm**%> - Indicates month as a three-letters abbreviation using the current user settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user settings.
- <%**loc_gg**%> - Indicates period/era using the current user settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user settings.
- <%**loc_time**%> - Indicates the time using the current user settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user settings.
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user settings.
- <%**loc_tsep**%> - Indicates the time separator using the current user settings
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the

Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

The Label property supports the following built-in HTML tags:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.

  The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using <a ;exp=> or <a ;e64=> anchor tags. The exp/e64 field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption.

  - exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
  - e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

  Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.

- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text

such as: Text with $_{subscript}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}

- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

shadow

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

outline anti-aliasing

The Label property may be a combination of any of these tags. For instance, the " <%mmm%> <%d%>, '<%yy%>" displays a date like: "**May** 29,'05".

The first level displays the month, the year and the number of the week in the year , the second level displays the name of the week day, and the third level displays the day of the month. The LevelCount property specifies the number of levels being displayed, in our case 3.

The following Template shows how to display your header using three levels as arranged in the picture above (just copy and paste the following script to Template page):

```
BeginUpdate()
Chart
{
   LevelCount = 3
   Level(0)
   {
      Label = "<b><%mmm%>, <%yyyy%></b> <r>Week: <%ww%>"
      Unit = 256 'exWeek
   }
   Level(1).Label = "<%d1%>"
   Level(2).Label = "<%d%>"
}
EndUpdate()
```

The following VB sample displays your header using 3 levels as shown above:

```
With G2antt1
   .BeginUpdate
   With .Chart
      .LevelCount = 3
      With .Level(0)
         .Label = "<b><%mmm%>, <%yyyy%></b> <r>Week: <%ww%>"
         .Unit = EXG2ANTTLibCtl.UnitEnum.exWeek
      End With
      .Level(1).Label = "<%d1%>"
      .Level(2).Label = "<%d%>"
   End With
   .EndUpdate
End With
```

The following VFP sample displays your header using 3 levels:

```
with thisform.g2antt1
.BeginUpdate()
with .Chart
   .LevelCount = 3
   with .Level(0)
      .Label = "<b><%mmm%>, <%yyyy%></b> <r>Week: <%ww%>"
      .Unit = 256
   endwith
   .Level(1).Label = "<%d1%>"
   .Level(2).Label = "<%d%>"
endwith
.EndUpdate()
endwith
```

The following VB.NET sample displays your header using 3 levels:

```
With AxG2antt1
   .BeginUpdate()
   With .Chart
      .LevelCount = 3
      With .Level(0)
         .Label = "<b><%mmm%>, <%yyyy%></b> <r>Week: <%ww%>"
         .Unit = EXG2ANTTLib.UnitEnum.exWeek
      End With
      .Level(1).Label = "<%d1%>"
      .Level(2).Label = "<%d%>"
   End With
   .EndUpdate()
End With
```

The following C# sample displays your header using 3 levels:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Chart chart = axG2antt1.Chart;
chart.LevelCount = 3;
chart.get_Level(0).Label = "<b><%mmm%>, <%yyyy%></b> <r>Week: <%ww%>";
chart.get_Level(0).Unit = EXG2ANTTLib.UnitEnum.exWeek;
chart.get_Level(1).Label = "<%d1%>";
```

```
chart.get_Level(2).Label = "<%d%>";
axG2antt1.EndUpdate();
```

The following C++ sample displays your header using 3 levels:

```
m_g2antt.BeginUpdate();
CChart chart = m_g2antt.GetChart();
chart.SetLevelCount( 3 );
chart.GetLevel(0).SetLabel(COleVariant( "<b><%mmm%>, <%yyyy%></b> <r>Week:
<%ww%>" ));
chart.GetLevel(0).SetUnit(256);
chart.GetLevel(1).SetLabel(COleVariant( "<%d1%>" ));
chart.GetLevel(2).SetLabel(COleVariant( "<%d%>" ));
m_g2antt.EndUpdate();
```

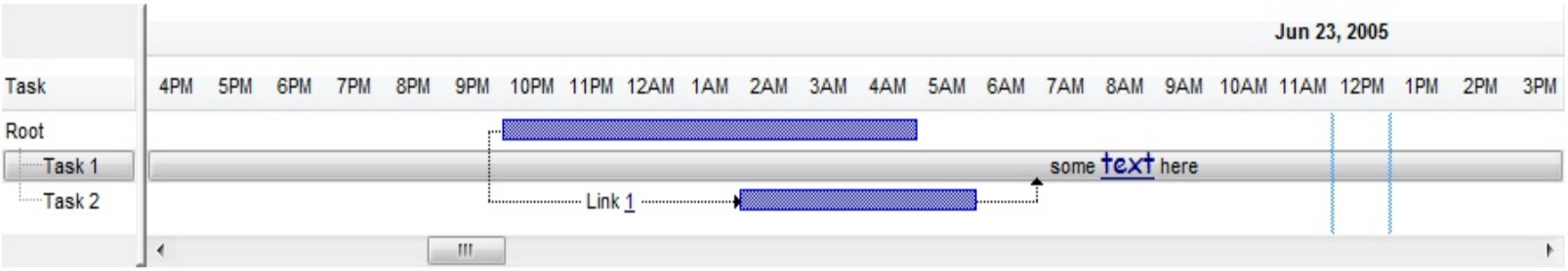
property Level.ReplaceLabel(Label as String) as String

Specifies a HTML replacement for the given label.

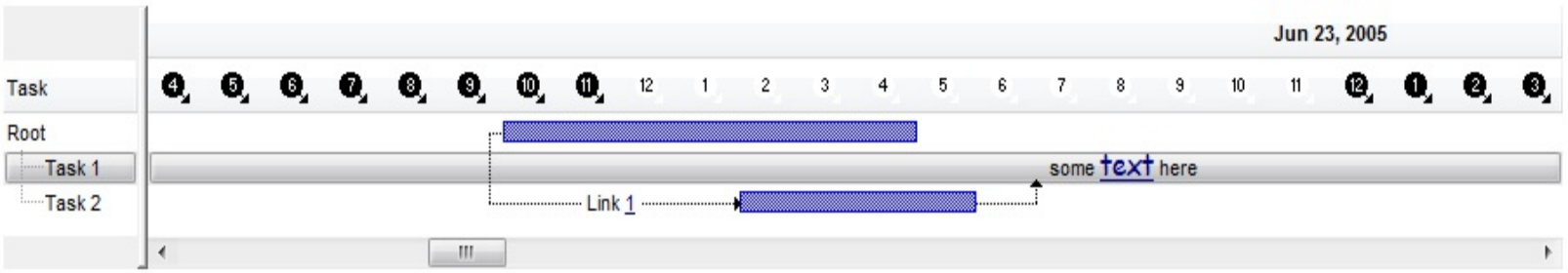
Type	Description
Label as String	A String expression that specifies the caption being replaced. If empty, the set method removes all replacements in the level.
String	A String expression that specifies the new caption, that can use built-n HTML tags as explained bellow.

By default, the Label property specifies the caption being displayed in the chart's header. Use the ReplaceLabel property to customize your chart's header. The ReplaceLabel property may be used to add icons or pictures (<img>), or change the captions of the levels in the chart's header. The ReplaceLabel property is a get/set property. When get property is called, the ReplaceLabel(Label) property returns the replacement HTML string for specified label. If the set property is called, the specified label is replaced with the newly value, so the newly value is displayed instead. You can remove all replacement by calling the set ReplaceLabel property with Label parameter as empty string. The Label parameter never includes the HTML built tags. For instance, if your Label property is "<%h%><%AM/PM%>", then the Label parameter should be: 12AM,1AM,2AM, and so on, as they are displayed on the chart's header. Use the FormatLabel property to format the label giving a formula.

The following screen shot shows the chart's header when no replacements are performed:

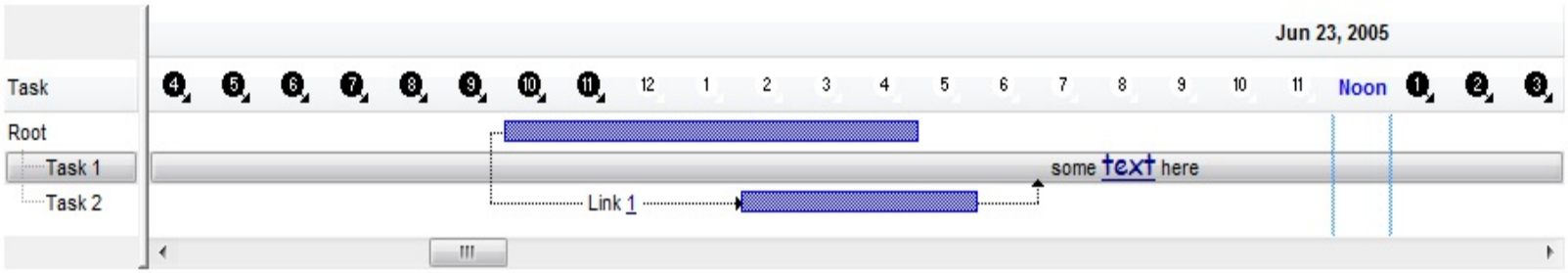


The following screen shot shows the chart's header when the hours were replaced with icons:



The following screen shot shows the chart's header when the hours were replaced with

icons, excepts the 12:00 PM were replaced by Noon caption:



The ReplaceLabel property supports the following built-in HTML elements:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.

  The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using <a ;exp=> or <a ;e64=> anchor tags. The exp/e64 field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption.

  - exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
  - e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

  Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold

- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{subscript}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

shadow

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

outline anti-aliasing

property Level.ToolTip as Variant

Specifies the format of the tooltip that's shown when the cursor hovers the level.

Type	Description
Variant	A String expression that indicates the format of the tooltip, or an UnitEnum expression that indicates the predefined tooltip being used. The LabelToolTip property specifies a predefined tooltip.

The ToolTip property specifies the tooltip being shown when the cursor hovers the level. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. The ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipWidth property to specify the width of the tooltip window. The WeekDays property retrieves or sets a value that indicates the list of names for each week day, separated by space. Use the MonthNames property to specify the name of the months in the year. The UnitScale property changes the Label, Unit and the ToolTip for a level with predefined values defined by the Label and LabelToolTip properties. The Tooltip(0, -1, , , , ,) event occurs once the level's tooltip is about to be shown (-1 if the mouse pointer hovers the levels of the chart).

The ToolTip property supports the following built-in tags:

- <%**d**%> - Day of the month in one or two numeric digits, as needed (1 to 31).
- <%**dd**%> - Day of the month in two numeric digits (01 to 31).
- <%**d1**%> - First letter of the weekday (S to S). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**d2**%> - First two letters of the weekday (Su to Sa). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**d3**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**ddd**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays property to specify the name of the days in the week). You can use the <%**loc_ddd**%> that indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%**loc_ddd**%> - Indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%**dddd**%> - Full name of the weekday (Sunday to Saturday). (Use the WeekDays

property to specify the name of the days in the week). You can use the <%**loc_dddd**%> that indicates day of week as its full name using the current user regional and language settings.

- <%**loc_dddd**%> - Indicates day of week as its full name using the current user regional and language settings.
- <%**i**%> - Displays the number instead the date. For instance, you can display numbers as 1000, 1001, 1002, 1003, instead dates. (the valid range is from -647,434 to 2,958,465)
- <%**w**%> - Day of the week (1 to 7).
- <%**ww**%> - Week of the year (1 to 53).
- <%**m**%> - Month of the year in one or two numeric digits, as needed (1 to 12).
- <%**mr**%> - Month of the year in Roman numerals, as needed (I to XII).
- <%**mm**%> - Month of the year in two numeric digits (01 to 12).
- <%**m1**%> - First letter of the month (J to D). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**m2**%> - First two letters of the month (Ja to De). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**m3**%> - First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**mmm**%> - First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmm**%> that indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%**loc_mmm**%> - Indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%**mmmm**%> - Full name of the month (January to December). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmmm**%> that indicates month as its full name using the current user regional and language settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user regional and language settings.
- <%**q**%> - Date displayed as the quarter of the year (1 to 4).
- <%**y**%> - Number of the day of the year (1 to 366).
- <%**yy**%> - Last two digits of the year (01 to 99).
- <%**yyyy**%> - Full year (0100 to 9999).
- <%**hy**%> - Date displayed as the half of the year (1 to 2).
- <%**loc_gg**%> - Indicates period/era using the current user regional and language

settings.

- <%**loc_sdate**%> - Indicates the date in the short format using the current user regional and language settings.
- <%**loc_ldate**%> - Indicates the date in the long format using the current user regional and language settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user regional and language settings (/).
- <%**h**%> - Hour in one or two digits, as needed (0 to 23).
- <%**hh**%> - Hour in two digits (00 to 23).
- <%**h12**%> - Hour in 12-hour time format, in one or two digits - [0(12),11]
- <%**hh12**%> - hour in 12-hour time format, in two digits - [00(12),11]
- <%**n**%> - Minute in one or two digits, as needed (0 to 59).
- <%**nn**%> - Minute in two digits (00 to 59).
- <%**s**%> - Second in one or two digits, as needed (0 to 59).
- <%**ss**%> - Second in two digits (00 to 59).
- <%**AM/PM**%> - Twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate. (Use the AMPM property to specify the name of the AM and PM indicators). You can use the <%**loc_AM/PM**%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%**loc_A/P**%> that indicates the one character time marker such as A or P using the current user regional and language settings
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user regional and language settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user regional and language settings.
- <%**loc_time**%> - Indicates the time using the current user regional and language settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%**loc_tsep**%> - indicates the time separator using the current user regional and language settings (:)
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

The following tags are displayed based on the user's Regional and Language Options:

- <%**loc_sdate**%> - Indicates the date in the short format using the current user settings.
- <%**loc_ldate**%> - Indicates the date in the long format using the current user settings.
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**loc_ddd**%> - Indicates day of week as a three-letters abbreviation using the current user settings.
- <%**loc_dddd**%> - Indicates day of week as its full name using the current user settings.
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**loc_mmm**%> - Indicates month as a three-letters abbreviation using the current user settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user settings.
- <%**loc_gg**%> - Indicates period/era using the current user settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user settings.
- <%**loc_time**%> - Indicates the time using the current user settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user settings.
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user settings.
- <%**loc_tsep**%> - Indicates the time separator using the current user settings
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits,

for example, "13". No additional leading zeros are displayed

The ToolTip property supports the following built-in HTML tags:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.

  The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using <a ;exp=> or <a ;e64=> anchor tags. The exp/e64 field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption.

  - exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
  - e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

  Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with

a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.

- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the

text such as: Text with ^{subscript}

- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

property Level.Unit as UnitEnum

Retrieves or sets a value that indicates the unit of the level.

Type	Description
UnitEnum	An UnitEnum expression that indicates the level's time unit.

The Unit property specifies the unit being used to display labels in the level. **The Label property may change the Unit and the Count property**. You can always use a different Unit or Count by setting the property after setting the Label property. Changing the Label property may change the Unit property. For instance, if the user calls Label = "<%d%>", the Unit property is automatically put on exDay. The UnitScale property indicates the minimum time unit from all levels. The UnitScale property changes the Label, Unit and the ToolTip for a level with predefined values defined by the Label and LabelToolTip properties. Use the LevelCount property to specify the count of levels in the chart's header. Use the UnitWidth property to specify the width of the time unit. Use the Count property to specify the number of units being displayed in the same place. Use the NextDate property to get the next date. Use the Zoom method to zoom the chart to a specified interval of dates.

The first level displays the month, the year and the number of the week in the year , the second level displays the name of the week day, and the third level displays the day of the month. The LevelCount property specifies the number of levels being displayed, in our case 3.

The following Template shows how to display your header using three levels as arranged in the picture above (just copy and paste the following script to Template page):

```
BeginUpdate()
Chart
{
   LevelCount = 3
   Level(0)
   {
      Label = "<b><%mmm%>, <%yyyy%></b> <r>Week: <%ww%>"
      Unit = 256 'exWeek
   }
   Level(1).Label = "<%d1%>"
   Level(2).Label = "<%d%>"
```

```
}
EndUpdate()
```

The following VB sample displays your header using 3 levels as shown above:

```
With G2antt1
   .BeginUpdate
   With .Chart
      .LevelCount = 3
      With .Level(0)
         .Label = "<b><%mmm%>, <%yyyy%></b> <r>Week: <%ww%>"
         .Unit = EXG2ANTTLibCtl.UnitEnum.exWeek
      End With
      .Level(1).Label = "<%d1%>"
      .Level(2).Label = "<%d%>"
   End With
   .EndUpdate
End With
```

The following VFP sample displays your header using 3 levels:

```
with thisform.g2antt1
.BeginUpdate()
with .Chart
   .LevelCount = 3
   with .Level(0)
      .Label = "<b><%mmm%>, <%yyyy%></b> <r>Week: <%ww%>"
      .Unit = 256
   endwith
   .Level(1).Label = "<%d1%>"
   .Level(2).Label = "<%d%>"
endwith
.EndUpdate()
endwith
```

The following VB.NET sample displays your header using 3 levels:

```
With AxG2antt1
   .BeginUpdate()
```

```
  With .Chart
    .LevelCount = 3
    With .Level(0)
      .Label = "<b><%mmm%>, <%yyyy%></b> <r>Week: <%ww%>"
      .Unit = EXG2ANTTLib.UnitEnum.exWeek
    End With
    .Level(1).Label = "<%d1%>"
    .Level(2).Label = "<%d%>"
  End With
  .EndUpdate()
End With
```

The following C# sample displays your header using 3 levels:

```
axG2antt1.BeginUpdate();
EXG2ANTTLib.Chart chart = axG2antt1.Chart;
chart.LevelCount = 3;
chart.get_Level(0).Label = "<b><%mmm%>, <%yyyy%></b> <r>Week: <%ww%>";
chart.get_Level(0).Unit = EXG2ANTTLib.UnitEnum.exWeek;
chart.get_Level(1).Label = "<%d1%>";
chart.get_Level(2).Label = "<%d%>";
axG2antt1.EndUpdate();
```

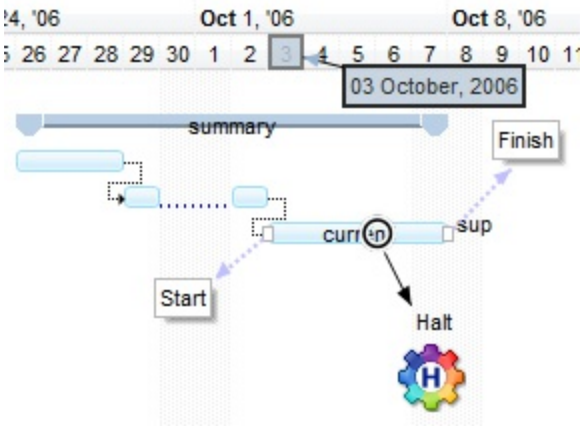
The following C++ sample displays your header using 3 levels:

```
m_g2antt.BeginUpdate();
CChart chart = m_g2antt.GetChart();
chart.SetLevelCount( 3 );
chart.GetLevel(0).SetLabel(COleVariant( "<b><%mmm%>, <%yyyy%></b> <r>Week:
<%ww%>" ));
chart.GetLevel(0).SetUnit(256);
chart.GetLevel(1).SetLabel(COleVariant( "<%d1%>" ));
chart.GetLevel(2).SetLabel(COleVariant( "<%d%>" ));
m_g2antt.EndUpdate();
```

Note object

A Note object indicates a fully customizable box that can be associated or attached with a DATE or a BAR in the chart area . Once a bar is associated with a DATE, the box is moved accordingly with the DATE in the chart, and the same if the note is attached to a bar. A Note can display HTML text, icons, pictures, borders, and so on. The Note can be positioned anywhere in the chart area relative to the DATE or BAR. A note is composed by two parts, the starting part and the ending part. Each part can display HTML text, icons, pictures, and so on. The Notes property access the chart's Notes collection. Use the Add method to add new notes in the chart. Use the exBarCaption to assign a caption to a bar, use the exBarExtraCaption to associate extra captions to a bar.

The following screen shot shows a note/box associated with a DATE (03 October, 2006), and 3 notes (Start, Finish and Halt)associated with the bar:



The Note object supports the following properties and methods:

Name	Description
ClearPartBackColor	Clears the background color for the part of the note.
ClearPartBorderColor	Clears the border of the note.
Data	Associates an user data to a note.
ID	Specifies the identifier of the note.
Item	Specifies the handle of the note's item.
Key	Specifies the key of the note.
LinkColor	Specifies the color of the link between parts of the note.
LinkStyle	Specifies the style of the link between parts of the note.
LinkWidth	Specifies the size of the link between parts of the note.
PartAlignment	Specifies the horizontal alignment of text inside the note's part.

PartBackColor	Specifies the background color to show the part of the note.
PartBorderColor	Specifies the color to show the border.
PartBorderSize	Specifies the size of the border for the note's part.
PartCanMove	Specifies whether the user can move the part of the note.
PartFixedHeight	Specifies whether the part has a fixed height.
PartFixedWidth	Specifies whether the part has a fixed width.
PartForeColor	Specifies the foreground color to show the part of the note.
PartHOffset	Specifies the horizontal offset to display the part of the note.
PartShadow	Specifies whether the part of the note shows a shadow border.
PartText	Specifies the HTML caption being shown in the part of the note.
PartToolTip	Specifies the HTML tooltip being shown when the cursor hovers the the part of the note.
PartToolTipTitle	Specifies the title tooltip being shown when the cursor hovers the the part of the note.
PartTransparency	Specifies the transparency to diaplay the part of the note.
PartVisible	Specifies whether a part of the note is visible or hidden.
PartVOffset	Specifies the vertical offset to display the part of the note.
RelativePosition	Specifies the position of the note relative to associated object.
ShowLink	Retrieves or sets a value that indicates the link between parts of the note.
Text	Specifies the HTML caption being shown in the first visible part of the note.
Visible	Specifies whether the note is visible or hidden.

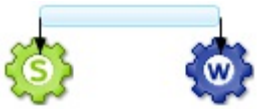
method Note.ClearPartBackColor (Part as NotePartEnum)

Clears the background color for the part of the note.

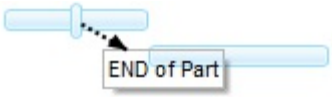
Type	Description
Part as NotePartEnum	A NotePartEnum expression that indicates whose part's background is cleared.

By default, both parts use the default window background color (white). Use the ClearPartBackColor method to erase the part's background color so you can put a transparent picture using the <img> tag in the PartText property. Use the PartBorderSize property on 0, to hide the part's borders. Use the PartShadow property to hide the shadow around the part. Use the ClearPartBackColor method to clear the part's background which means that the part shows only the borders (PartBorderSize property is greater than 0), shadows (PartShadow property is True) and the text of the part (PartText property), so the part is shown with no erasing its background. The PartBackColor property specifies the part's background color. Use the PartVisible property to show or hide the note's starting or ending part. Use the PartTransparency property to specify the transparency to display the part of the note. The PartForeColor property to specify the part's foreground color. Use the <bgcolor> HTML tag in the PartText property to specify parts of the note's caption with different background colors. The PartBorderColor property indicates the color to show the part's frame.

The following sample shows notes with pictures (PartBorderSize = 0, PartShadow = False, PartText = "<img>p1</img>") :



The following sample shows the note with the no PartBackColor property set (actually the ClearPartBackColor method is called before) :



The following sample shows the note with the PartBackColor property set on red:

The following sample shows the note with the PartBackColor property set on red, semi-transparent (PartTransparency property is 50):

The following VB sample assigns a note to the bar, by displaying a picture, when user right clicks the bar:

```
Private Sub G2antt1_RClick()
   Dim h As Long, c As Long, hit As HitTestInfoEnum
   G2antt1.BeginUpdate
      With G2antt1
         h = .ItemFromPoint(-1, -1, c, hit)
         If (h <> 0) Then
            Dim k As Variant
            k = .Chart.BarFromPoint(-1, -1)
            If (Not IsEmpty(k)) Then
               With .Chart.Notes.Add(.Chart.Notes.Count, h, k, "<img>p1</img>")
                  .ClearPartBackColor exNoteEnd
                  .PartBorderSize(exNoteEnd) = 0
                  .PartShadow(exNoteEnd) = False
               End With
            End If
         End If
      End With
   G2antt1.EndUpdate
End Sub
```

method Note.ClearPartBorderColor (Part as NotePartEnum)

Clears the border of the note.

Type	Description
Part as NotePartEnum	A NotePartEnum expression that specifies the part whose border is cleared.

If the ClearPartBorderColor method is called, the part show no border, until the PartBorderColor is set again. By default, the PartBorderSize property is 1, which means that the part draws a frame of color being indicated by the PartBorderColor property. Use the PartTransparency property to specify the transparency to display the part of the note. The PartForeColor property to specify the part's foreground color. Use the <bgcolor> HTML tag in the PartText property to specify parts of the note's caption with different background colors. Use the The PartBackColor property to specify an EBN object to show a different visual appearance for the part (borders and background).

property Note.Data as Variant

Associates an user data to a note.

Type	Description
Variant	A VARIANT expression that specifies any extra data associated with the Note.

By default, the Data property is Empty. Use the Data property to associate any extra data to the note. The Data property can store anything, from numbers, strings to objects. The ID property indicates the identifier of the Note object.

property Note.ID as Variant

Specifies the identifier of the note.

Type	Description
Variant	A VARIANT expression that specifies the identifier of the note. Could be a number, a string, a date, an object, and so on.

The ID property indicates the unique identifier to refer a note. Currently, this property is read only, so use the ID parameter of the Add method to specify the identifier of the note. The Data property can store anything, from numbers, strings to objects. Use the Item property to access a note giving its identifier. The Count property indicates the number of Note objects in the Notes collection.

The following VB sample prints the ID for each note in the control:

```
Dim n As EXG2ANTTLibCtl.Note
For Each n In G2antt1.Chart.Notes
   Debug.Print n.ID
Next
```

property Note.Item as Variant

Specifies the handle of the note's item.

Type	Description
Variant	A long expression that specifies the handle of the item that hosts the note.

The Item property returns the same value as Add's Item parameter. The Add method adds a note or a box associated with a DATE or a BAR in the chart. The Item property indicates the handle of the item that hosts the note. Once an item is move to a different position, the associated notes are moved too.

property Note.Key as Variant

Specifies the key of the note.

Type	Description
Variant	A VARIANT expression that specifies the key of the note.

The Key property returns the same value as the Add's Key parameter. The Key parameter of the Add method specifies the object to relate the note. The Key could be one of the following:

- Key parameter is of Date type, it indicates the DATE in the chart to associate the note. By default, the RelativePosition property is 0.5, which indicates the center of the unit where the DATE is (0, means the start unit, while the 1 is the end of the unit, and so on). The DateFromPoint property retrieves the date from the point. By default, If a note is associated to a DATE, the RelativePostion property is 0.5, it displays only the ending part of the note, and the ending part of the note is not movable.
- Key parameter is not of Date type, it indicates the Key of the BAR to associate the note (The Item and the Key indicates the bar to associate the note). By default, the RelativePosition property is 0, which indicates the starting point of the bar (0, means the starting point of the bar, while the 1 is the ending point of the bar, 0.5 indicates the middle of the bar, and so on). The BarFromPoint property retrieves the key of the bar from the cursor. By default, If a note is associated to a BAR, the RelativePostion property is 0, it displays only the ending part of the note, and the ending part of the note is not movable. Also, the direction from start to end part is visible.

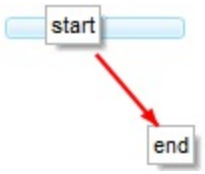
By default, the starting part of the note is not visible, so only the ending part of the note is visible.

property Note.LinkColor as Color

Specifies the color of the link between parts of the note.

Type	Description
Color	A Color expression that determines the color of the link between parts of the note.

The LinkColor property specifies the color of the link between parts of the notes. The ShowLink property specifies whether the note shows or hides the link between parts of the notes. Use the PartVisible property to specify whether the start or ending part of the note is visible or hidden. The LinkStyle property determines the style of the link between parts of the note. while the LinkWidth property determines the width of the link between parts of the notes.



The link between parts of the note is shown

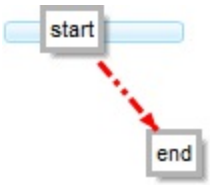
- if the ShowLink property includes the exNoteLinkVisible flag,
- LinkWidth property is greater than 0,
- the start and end part of the note do not intersect.

property Note.LinkStyle as LinkStyleEnum

Specifies the style of the link between parts of the note.

Type	Description
LinkStyleEnum	A LinkStyleEnum expression that specifies the style of the link between parts of the note.

The LinkStyle property determines the style of the link between parts of the note. The LinkColor property specifies the color of the link between parts of the notes. The ShowLink property specifies whether the note shows or hides the link between parts of the notes. Use the PartVisible property to specify whether the start or ending part of the note is visible or hidden. The LinkWidth property determines the width of the link between parts of the notes.



The link between parts of the note is shown

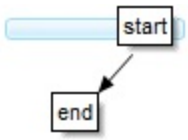
- if the ShowLink property includes the exNoteLinkVisible flag,
- LinkWidth property is greater than 0,
- the start and end part of the note do not intersect.

property Note.LinkWidth as Long

Specifies the size of the link between parts of the note.

Type	Description
Long	A long expression that specifies the size in pixels of the link between parts of the note.

By default, the LinkWidth property is 1. The link is not shown if the LinkWidth property is 0. The LinkWidth property determines the width of the link between parts of the notes. The LinkStyle property determines the style of the link between parts of the note. The LinkColor property specifies the color of the link between parts of the notes. The ShowLink property specifies whether the note shows or hides the link between parts of the notes. Use the PartVisible property to specify whether the start or ending part of the note is visible or hidden.



The link between parts of the note is shown

- if the ShowLink property includes the exNoteLinkVisible flag,
- LinkWidth property is greater than 0,
- the start and end part of the note do not intersect.

property Note.PartAlignment(Part as NotePartEnum) as AlignmentEnum

Specifies the horizontal alignment of text inside the note's part.

Type	Description
Part as NotePartEnum	A NotePartEnum expression that indicates the part to align text on
AlignmentEnum	An AlignmentEnum expression that specifies the text's alignment.

By default the PartAlignment property is CenterAlignment, so the text is being centered in the part. Use the Text or PartText property to specify the text to be displayed on the part of the note. The PartFixedWidth property specifies the fixed width to display the part. The PartFixedHeight property specifies the height to display the part of the note. If none of these properties are set, the size of the part is automatically computed based on the part's text.

The following screen shot shows the text being aligned to the left and to the right side of the note:

property Note.PartBackColor(Part as NotePartEnum) as Color

Specifies the background color to show the part of the note.

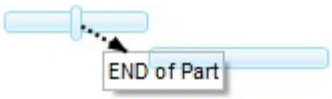
Type	Description
Part as NotePartEnum	A NotePartEnum expression that indicates whose part's background is changed.
Color	A Color expression that specifies the part's background color. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part.

By default, both parts use the default window background color (white). The PartBackColor property specifies the part's background color. Use the PartVisible property to show or hide the note's starting or ending part. Use the ClearPartBackColor method to clear the part's background which means that the part shows only the borders (PartBorderSize property is greater than 0), shadows (PartShadow property is True) and the text of the part (PartText property), so the part is shown with no erasing its background. Use the PartTransparency property to specify the transparency to display the part of the note. The PartForeColor property to specify the part's foreground color. Use the <bgcolor> HTML tag in the PartText property to specify parts of the note's caption with different background colors. The PartBorderColor property indicates the color to show the part's frame.

The following sample shows the note with the PartBackColor property set on red:

The following sample shows the note with the PartBackColor property set on red, semi-transparent (PartTransparency property is 50):

The following sample shows the note with the no PartBackColor property set (actually the ClearPartBackColor method is called before) :



property Note.PartBorderColor(Part as NotePartEnum) as Color

Specifies the color to show the border.

Type	Description
Part as NotePartEnum	A NotePartEnum expression that indicates the part being accessed.
Color	A Color expression that specifies the color for the part's border.

The PartBorderColor property indicates the color to show the part's frame. By default, the PartBorderSize property is 1, which means that the part draws a frame of color being indicated by the PartBorderColor property. Use the PartBorderSize property on 0, to hide the part's borders. Use the ClearPartBackColor method to erase the part's background color so you can put a transparent picture using the <img> tag in the PartText property. Use the PartShadow property to hide the shadow around the part. Use the ClearPartBackColor method to clear the part's background which means that the part shows only the borders (PartBorderSize property is greater than 0), shadows (PartShadow property is True) and the text of the part (PartText property), so the part is shown with no erasing its background. The PartBackColor property specifies the part's background color. Use the PartVisible property to show or hide the note's starting or ending part. Use the PartTransparency property to specify the transparency to display the part of the note. The PartForeColor property to specify the part's foreground color. Use the <bgcolor> HTML tag in the PartText property to specify parts of the note's caption with different background colors. Use the The PartBackColor property to specify an EBN object to show a different visual appearance for the part (borders and background).

property Note.PartBorderSize(Part as NotePartEnum) as Long

Specifies the size of the border for the note's part.

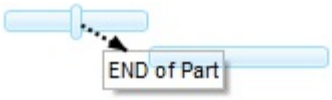
Type	Description
Part as NotePartEnum	A NotePartEnum expression that indicates whose part's border is changed.
Long	A Long expression that specifies the size of the frame around the part.

By default, the PartBorderSize property is 1, which means that the part draws a frame of color being indicated by the PartBorderColor property. Use the PartBorderSize property on 0, to hide the part's borders. Use the ClearPartBackColor method to erase the part's background color so you can put a transparent picture using the <img> tag in the PartText property. Use the PartShadow property to hide the shadow around the part. Use the ClearPartBackColor method to clear the part's background which means that the part shows only the borders (PartBorderSize property is greater than 0), shadows (PartShadow property is True) and the text of the part (PartText property), so the part is shown with no erasing its background. The PartBackColor property specifies the part's background color. Use the PartVisible property to show or hide the note's starting or ending part. Use the PartTransparency property to specify the transparency to display the part of the note. The PartForeColor property to specify the part's foreground color. Use the <bgcolor> HTML tag in the PartText property to specify parts of the note's caption with different background colors. The PartBorderColor property indicates the color to show the part's frame.

The following sample shows notes with hyperlinks and pictures:

The following sample shows notes with pictures (PartBorderSize = 0, PartShadow = False, PartText = "<img>p1</img>") :

The following sample shows the note with the no PartBackColor property set (actually the ClearPartBackColor method is called before) :



The following sample shows the note with the PartBackColor property set on red:

The following sample shows the note with the PartBackColor property set on red, semi-transparent (PartTransparency property is 50):

The following VB sample assigns a note to the bar, by displaying a picture, when user right clicks the bar:

```
Private Sub G2antt1_RClick()
   Dim h As Long, c As Long, hit As HitTestInfoEnum
   G2antt1.BeginUpdate
      With G2antt1
         h = .ItemFromPoint(-1, -1, c, hit)
         If (h <> 0) Then
            Dim k As Variant
            k = .Chart.BarFromPoint(-1, -1)
            If (Not IsEmpty(k)) Then
               With .Chart.Notes.Add(.Chart.Notes.Count, h, k, "<img>p1</img>")
                  .ClearPartBackColor exNoteEnd
                  .PartBorderSize(exNoteEnd) = 0
                  .PartShadow(exNoteEnd) = False
               End With
            End If
         End If
      End With
   G2antt1.EndUpdate
End Sub
```

property Note.PartCanMove(Part as NotePartEnum) as Boolean

Specifies whether the user can move the part of the note.

Type	Description
Part as NotePartEnum	A NotePartEnum expression that specifies the part to be movable
Boolean	A Boolean expression that specifies whether the part of the note is movable or fixed.

By default, all parts of the note are fixed, in other words the PartCanMove property is False. Once, the PartCanMove property is True, the shape of the cursor that hovers the part is changed to indicate a movement, and so the user can click and drag the part to a new position. Use the PartCanMove property to allow the user to move the note relative to the DATE or BAR, or relative to the starting part of the note. For instance, if the user moves the starting part of the note, the RelativePosition property is changed from 0 to 1, while if the ending part of the note is moved, the PartHOffset / PartVOffset properties are adjusted. Use the PartVisible property to show or hide the note's starting or ending part.

The movement of the parts can be:

- *PartCanMove(exNoteStart) = True, PartCanMove(exNoteEnd) = True*, both parts are moveable, so if the user moves the starting part of the note, the note is moved relative to the object being related such as a DATE or a BAR (RelativePosition), while if the user moves the ending part of the note, it is moved relative to the starting part of the note (PartHOffset(exNoteEnd) / PartVOffset(exNoteEnd)).
- *PartCanMove(exNoteStart) = False, PartCanMove(exNoteEnd) = True*, only the ending part is movable, so the starting part of the note is fixed, while the user can move the ending part of the note, relative to the starting part of the bar (PartHOffset(exNoteEnd) / PartVOffset(exNoteEnd)).
- *PartCanMove(exNoteStart) = True, PartCanMove(exNoteEnd) = False*, only the starting part is movable, so the user moves the note relative to the related object DATE or BAR, and it is valid for any part of the note (RelativePosition). For instance, even if the cursor hovers the ending part of the note, the note is moved relative to the object DATE or BAR.

The RelativePosition property indicates a float value between 0 and 1 relative to the object DATE or BAR associated with as follow:

- if the note is associated with a DATE (the Key parameter of the Add method is of DATE type), 0 indicates the starting of the unit where the DATE is, 1 indicates the finish of the time unit where the DATE is, and 0.5 indicates the center of the unit.
- if the note is associated with a BAR (the Key parameter of the Add method is of DATE

type), 0 indicates the starting point of the bar, 1 indicates the ending point of the bar, 0.5 indicates the middle of the bar.

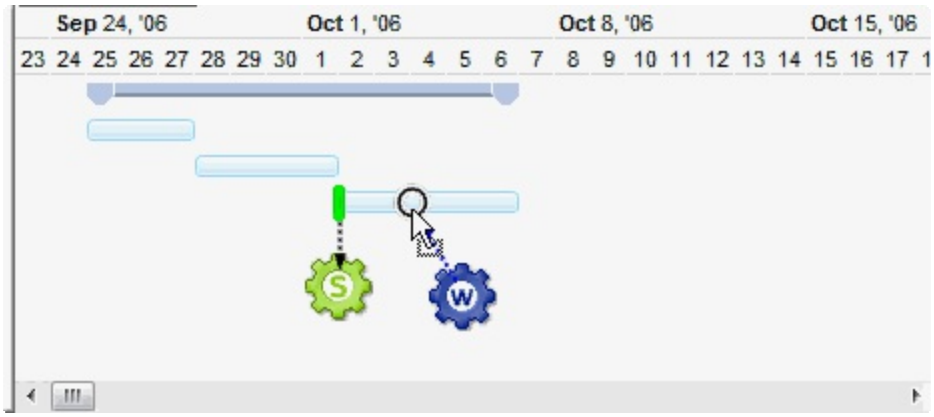
The PartHOffset / PartVOffset properties indicate the horizontal / vertical offset as follow:

- for exNoteStart, the PartHOffset / PartVOffset properties indicate the offset relative to the point referred by the RelativePosition property.
- for exNoteEnd, the PartHOffset / PartVOffset properties indicate the offset relative to the exNoteStart

The moving cursor is shown on the movable part if:

- the part is visible and it means the PartVisible property is True, and it is visible on the screen (has the PartText not empty, or has the both PartFixedWidth / PartFixedHeight properties not zero)
- the exNoteStart is movable, and it means that the PartCanMove(exNoteStart) property is True
- the exNoteEnd is movable, and it means that the PartCanMove(exNoteStart) OR PartCanMove(exNoteEnd) property is True.

The following screen shows shows how the notes can be moved by the user:



property Note.PartFixedHeight(Part as NotePartEnum) as Long

Specifies whether the part has a fixed height.

Type	Description
Part as NotePartEnum	A NotePartEnum expression that indicates the part to specify the height
Long	A Long expression that specifies the fixed height of the part.

By default, the PartFixedHeight property is 0. The property has no effect if 0. The PartFixedWidth / PartFixedHeight property is 0, the control computes the width / height of the note, else it indicates the fixed width / height. The PartText property indicates the HTML caption being shown in the note. If no text is assigned, use the PartFixedWidth / PartFixedHeight properties to specify a blank frame. Use the The PartBackColor property to specify an EBN object to show a different visual appearance for the part (borders and background). The PartBorderColor property indicates the color to show the part's frame. Use the PartBorderSize property on 0, to hide the part's borders. Use the ClearPartBackColor method to erase the part's background color so you can put a transparent picture using the <img> tag in the PartText property. Use the PartShadow property to hide the shadow around the part.

property Note.PartFixedWidth(Part as NotePartEnum) as Long

Specifies whether the part has a fixed width.

Type	Description
Part as NotePartEnum	A NotePartEnum expression that indicates the part to specify the width
Long	A Long expression that specifies the fixed width of the part.

By default, the PartFixedWidth property is 0. The property has no effect if 0. The PartText property indicates the HTML caption being shown in the note. If no text is assigned, use the PartFixedWidth / PartFixedHeight properties to specify a blank frame. Use the The PartBackColor property to specify an EBN object to show a different visual appearance for the part (borders and background). The PartBorderColor property indicates the color to show the part's frame. Use the PartBorderSize property on 0, to hide the part's borders. Use the ClearPartBackColor method to erase the part's background color so you can put a transparent picture using the <img> tag in the PartText property. Use the PartShadow property to hide the shadow around the part.

property Note.PartForeColor(Part as NotePartEnum) as Color

Specifies the foreground color to show the part of the note.

Type	Description
Part as NotePartEnum	A NotePartEnum expression that specifies the part.
Color	A Color expression that specifies the part's foreground color.

The PartForeColor property indicates the foreground color to display the PartText. Use the <fgcolor> HTML tag in the PartText property to specify parts of the note's caption with different foreground colors. Use the <bgcolor> HTML tag in the PartText property to specify parts of the note's caption with different background colors. Use the The PartBackColor property to specify an EBN / Solid Color object to show a different visual appearance for the part (borders and/or background).

property Note.PartHOffset(Part as NotePartEnum) as Long

Specifies the horizontal offset to display the part of the note.

Type	Description
Part as NotePartEnum	A NotePartEnum expression that specifies the part to add horizontal offset
Long	A long expression that specifies the horizontal offset of the part of the note.

The PartHOffset(exNoteStart) property indicates the horizontal offset relative to the point being referred by the RelativePosition property. The PartHOffset(exNoteEnd) property indicates the horizontal offset relative to the starting part of the note. The PartCanMove property specifies whether the user can move at runtime the part of the note.

The RelativePosition property indicates a float value between 0 and 1 relative to the object DATE or BAR associated with as follow:

- if the note is associated with a DATE (the Key parameter of the Add method is of DATE type), 0 indicates the starting of the unit where the DATE is, 1 indicates the finish of the time unit where the DATE is, and 0.5 indicates the center of the unit.
- if the note is associated with a BAR (the Key parameter of the Add method is of DATE type), 0 indicates the starting point of the bar, 1 indicates the ending point of the bar, 0.5 indicates the middle of the bar.

The PartHOffset / PartVOffset properties indicate the horizontal / vertical offset as follow:

- for exNoteStart, the PartHOffset / PartVOffset properties indicate the offset relative to the point referred by the RelativePosition property.
- for exNoteEnd, the PartHOffset / PartVOffset properties indicate the offset relative to the exNoteStart

The moving cursor is shown on the movable part if:

- the part is visible and it means the PartVisible property is True, and it is visible on the screen (has the PartText not empty, or has the both PartFixedWidth / PartFixedHeight properties not zero)
- the exNoteStart is movable, and it means that the PartCanMove(exNoteStart) property is True
- the exNoteEnd is movable, and it means that the PartCanMove(exNoteStart) OR PartCanMove(exNoteEnd) property is True.

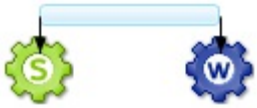
property Note.PartShadow(Part as NotePartEnum) as Boolean

Specifies whether the part of the note shows a shadow border.

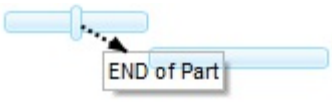
Type	Description
Part as NotePartEnum	A NotePartEnum expression that indicates whose part's shadow is changed.
Boolean	A boolean expression that specifies whether the part shows a shadow around.

By default, the PartShadow property is True. Use the PartShadow property to hide the shadow around the part. By default, the PartBorderSize property is 1, which means that the part draws a frame of color being indicated by the PartBorderColor property. Use the PartBorderSize property on 0, to hide the part's borders. Use the ClearPartBackColor method to erase the part's background color so you can put a transparent picture using the <img> tag in the PartText property. Use the ClearPartBackColor method to clear the part's background which means that the part shows only the borders (PartBorderSize property is greater than 0), shadows (PartShadow property is True) and the text of the part (PartText property), so the part is shown with no erasing its background. The PartBackColor property specifies the part's background color. Use the PartVisible property to show or hide the note's starting or ending part. Use the PartTransparency property to specify the transparency to display the part of the note. The PartForeColor property to specify the part's foreground color. Use the <bgcolor> HTML tag in the PartText property to specify parts of the note's caption with different background colors. The PartBorderColor property indicates the color to show the part's frame.

The following sample shows notes with pictures (PartBorderSize = 0, PartShadow = False, PartText = "<img>p1</img>") :



The following sample shows the note with the no PartBackColor property set (actually the ClearPartBackColor method is called before) :



The following sample shows the note with the PartBackColor property set on red:

The following sample shows the note with the PartBackColor property set on red, semi-transparent (PartTransparency property is 50):

The following VB sample assigns a note to the bar, by displaying a picture, when user right clicks the bar:

```
Private Sub G2antt1_RClick()
   Dim h As Long, c As Long, hit As HitTestInfoEnum
   G2antt1.BeginUpdate
      With G2antt1
         h = .ItemFromPoint(-1, -1, c, hit)
         If (h <> 0) Then
            Dim k As Variant
            k = .Chart.BarFromPoint(-1, -1)
            If (Not IsEmpty(k)) Then
               With .Chart.Notes.Add(.Chart.Notes.Count, h, k, "<img>p1</img>")
                  .ClearPartBackColor exNoteEnd
                  .PartBorderSize(exNoteEnd) = 0
                  .PartShadow(exNoteEnd) = False
               End With
            End If
         End If
      End With
   G2antt1.EndUpdate
End Sub
```

property Note.PartText(Part as NotePartEnum) as String

Specifies the HTML caption being shown in the part of the note.

Type	Description
Part as NotePartEnum	A NotePartEnum expression that indicates the part to put text on
String	A String expression that specifies the HTML text to be displayed on the ending part of the note. The Text parameter supports HTML tags as well as Chart Tags such as <%dd%> that displays the day in 2 digits, and so on like described bellow. Use the Images method to specify a list of icons that can be displayed in the control using the <img> tag. Use the HTMLPicture property to add custom- size pictures to be used in the HTML captions using the <img> tag.

By default, the PartText property is empty. Use the Text parameter of the Add method to specify the text on the ending part on adding the note. If no text is assigned, use the PartFixedWidth / PartFixedHeight properties to specify a blank frame. The PartForeColor property to specify the part's foreground color. Use the <bgcolor> HTML tag in the PartText property to specify parts of the note's caption with different background colors. Use the <fgcolor> HTML tag in the PartText property to specify parts of the note's caption with different foreground colors. The PartBorderColor property indicates the color to show the part's frame. Use the The PartBackColor property to specify an EBN object to show a different visual appearance for the part (borders and background). The PartAlignment property specifies the text's alignment in the part of the note.

If you need to display a picture only, you need:

- if require icons, use the Images method to specify a list of icons that can be displayed in the control
- if require custom size pictures, use the HTMLPicture property to add custom- size pictures
- set the PartVisible property to show or hide the note's starting or ending part.
- use the ClearPartBackColor method to clear the part's background
- set the PartBorderSize property on 0
- set the PartShadow property on False

The PartText property supports the following:

- <%**d**%> - Day of the month in one or two numeric digits, as needed (1 to 31).
- <%**dd**%> - Day of the month in two numeric digits (01 to 31).
- <%**d1**%> - First letter of the weekday (S to S). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**d2**%> - First two letters of the weekday (Su to Sa). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**d3**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**ddd**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays property to specify the name of the days in the week). You can use the <%**loc_ddd**%> that indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%**loc_ddd**%> - Indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%**dddd**%> - Full name of the weekday (Sunday to Saturday). (Use the WeekDays property to specify the name of the days in the week). You can use the <%**loc_dddd**%> that indicates day of week as its full name using the current user regional and language settings.
- <%**loc_dddd**%> - Indicates day of week as its full name using the current user regional and language settings.
- <%**i**%> - Displays the number instead the date. For instance, you can display numbers as 1000, 1001, 1002, 1003, instead dates. (the valid range is from -647,434 to 2,958,465)
- <%**w**%> - Day of the week (1 to 7).
- <%**ww**%> - Week of the year (1 to 53).
- <%**m**%> - Month of the year in one or two numeric digits, as needed (1 to 12).
- <%**mr**%> - Month of the year in Roman numerals, as needed (I to XII).
- <%**mm**%> - Month of the year in two numeric digits (01 to 12).
- <%**m1**%> - First letter of the month (J to D). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**m2**%> - First two letters of the month (Ja to De). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**m3**%> - First three letters of the month (Jan to Dec). (Use the MonthNames

property to specify the name of the months in the year)
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**mmm**%> - First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmm**%> that indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%**loc_mmm**%> - Indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%**mmmm**%> - Full name of the month (January to December). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmmm**%> that indicates month as its full name using the current user regional and language settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user regional and language settings.
- <%**q**%> - Date displayed as the quarter of the year (1 to 4).
- <%**y**%> - Number of the day of the year (1 to 366).
- <%**yy**%> - Last two digits of the year (01 to 99).
- <%**yyyy**%> - Full year (0100 to 9999).
- <%**hy**%> - Date displayed as the half of the year (1 to 2).
- <%**loc_gg**%> - Indicates period/era using the current user regional and language settings.
- <%**loc_sdate**%> - Indicates the date in the short format using the current user regional and language settings.
- <%**loc_ldate**%> - Indicates the date in the long format using the current user regional and language settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user regional and language settings (/).
- <%**h**%> - Hour in one or two digits, as needed (0 to 23).
- <%**hh**%> - Hour in two digits (00 to 23).
- <%**h12**%> - Hour in 12-hour time format, in one or two digits - [0(12),11]
- <%**hh12**%> - hour in 12-hour time format, in two digits - [00(12),11]
- <%**n**%> - Minute in one or two digits, as needed (0 to 59).
- <%**nn**%> - Minute in two digits (00 to 59).
- <%**s**%> - Second in one or two digits, as needed (0 to 59).
- <%**ss**%> - Second in two digits (00 to 59).
- <%**AM/PM**%> - Twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate. (Use the AMPM property to specify the name of the AM and PM indicators). You can use the <%**loc_AM/PM**%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%**loc_A/P**%> that indicates the one character time marker such as A or P using the current user regional and language settings
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current

user regional and language settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user regional and language settings.
- <%**loc_time**%> - Indicates the time using the current user regional and language settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%**loc_tsep**%> - indicates the time separator using the current user regional and language settings (:)
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

The following tags are displayed based on the user's Regional and Language Options:

- <%**loc_sdate**%> - Indicates the date in the short format using the current user settings.
- <%**loc_ldate**%> - Indicates the date in the long format using the current user settings.
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**loc_ddd**%> - Indicates day of week as a three-letters abbreviation using the current user settings.
- <%**loc_dddd**%> - Indicates day of week as its full name using the current user settings.
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**loc_mmm**%> - Indicates month as a three-letters abbreviation using the current user settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user settings.

- <%**loc_gg**%> - Indicates period/era using the current user settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user settings.
- <%**loc_time**%> - Indicates the time using the current user settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user settings.
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user settings.
- <%**loc_tsep**%> - Indicates the time separator using the current user settings
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

The PartText property supports the following built-in HTML tags:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.

  The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using <a ;exp=> or <a ;e64=> anchor tags. The exp/e64 field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption.

  - exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
  - e64, encodes in BASE64 the HTML text to be shown once the user clicks the

anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part

of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.

- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{subscript}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the

color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

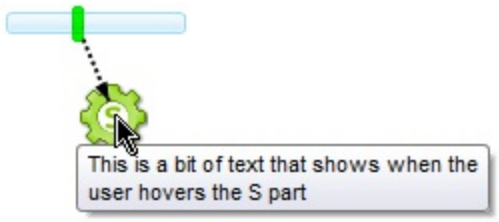
property Note.PartToolTip(Part as NotePartEnum) as String

Specifies the HTML tooltip being shown when the cursor hovers the the part of the note.

Type	Description
Part as NotePartEnum	A NotePartEnum expression that specifies the part with the tooltip.
String	A String expression that indicates the HTML text to be displayed when the cursor hovers the part.

By default, the PartToolTip property is empty. The tooltip is shown if the cursor hovers the part, if the PartToolTip or PartToolTipTitle property is not empty. Use the ToolTipWidth property to specify the width of the tooltip window. The ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the <font> element to specify a different font or size for the tooltip, or use the ToolTipFont property to specify a different font or size for all tooltips in the control. The Tooltip(0, -4, , , , ,) event occurs once the note's tooltip (Note.PartToolTip) is about to be shown (-4 if the mouse pointer hovers the notes of the chart).

The following screen shot shows the tooltip being shown when the cursor hovers the note:



The PartToolTip property supports the following HTML tags:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.

  The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using <a ;exp=> or <a ;e64=> anchor tags. The exp/e64 field of the anchor stores the HTML line/lines to show once

the user clicks/collapses/expands the caption.

  - exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
  - e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break

- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with $^{\text{subscript}}$
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor></**sha**>*" gets:

property Note.PartToolTipTitle(Part as NotePartEnum) as String

Specifies the title tooltip being shown when the cursor hovers the the part of the note.

Type	Description
Part as NotePartEnum	A NotePartEnum expression that specifies the part with the tooltip.
String	A String expression that indicates the title to be shown for the part's tooltip.

By default, the PartToolTipTitle property is empty. The tooltip is shown if the cursor hovers the part, if the PartToolTip or PartToolTipTitle property is not empty. Use the ToolTipWidth property to specify the width of the tooltip window. The ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the <font> element to specify a different font or size for the tooltip, or use the ToolTipFont property to specify a different font or size for all tooltips in the control. The Tooltip(0, -4, , , , ,) event occurs once the note's tooltip (Note.PartToolTip) is about to be shown (-4 if the mouse pointer hovers the notes of the chart).

property Note.PartTransparency(Part as NotePartEnum) as Long

Specifies the transparency to diaplay the part of the note.

Type	Description
Part as NotePartEnum	A NotePartEnum expression that specifies the part to be shown semi-transparent
Long	A Long expression between 0 and 100 which indicates the percent of transparency to show the part. 0 means opaque, 50 means semi-transparent, while 100 means fully transparent.

By default, the PartTransparency property is 0. The PartBackColor property specifies the part's background color. Use the PartVisible property to show or hide the note's starting or ending part. Use the ClearPartBackColor method to clear the part's background which means that the part shows only the borders (PartBorderSize property is greater than 0), shadows (PartShadow property is True) and the text of the part (PartText property), so the part is shown with no erasing its background. The PartForeColor property to specify the part's foreground color. Use the <bgcolor> HTML tag in the PartText property to specify parts of the note's caption with different background colors. Use the <fgcolor> HTML tag in the PartText property to specify parts of the note's caption with different foreground colors. The PartBorderColor property indicates the color to show the part's frame.

The following sample shows the note with the PartBackColor property set on red, semi-transparent (PartTransparency property is 50):

property Note.PartVisible(Part as NotePartEnum) as Boolean

Specifies whether a part of the note is visible or hidden.

Type	Description
Part as NotePartEnum	A NotePartEnum expression that specifies the part being visible or hidden.
Boolean	A Boolean expression that specifies whether the indicated part is visible or hidden.

By default, only the ending part of the note is visible. Use the PartVisible property to specify whether the start or ending part of the note is visible or hidden. The PartText property indicates the HTML caption to be displayed in the start or ending part of the note. The ShowLink property specifies whether the link between parts of the notes is visible or hidden. The PartFixedWidth property specifies whether the part is using fixed width or when the width of the part is based on the part's caption. The PartFixedHeight property specifies whether the part is using fixed height or when the height of the part is based on the part's caption.

The following VB sample adds a note associated with the DATE being double clicked:

```
Private Sub G2antt1_DblClick(Shift As Integer, X As Single, Y As Single)
  With G2antt1
    .BeginUpdate
    Dim h As Long, c As Long, hit As HitTestInfoEnum
    h = G2antt1.ItemFromPoint(-1, -1, c, hit)
    If (h <> 0) Then
      Dim d As Date
      d = .Chart.DateFromPoint(-1, -1)
      If (d <> 0) Then
        With .Chart.Notes.Add(d, h, d, "")
          .PartVisible(exNoteEnd) = False
          .PartVisible(exNoteStart) = True
          .PartText(exNoteStart) = "<%dd%><br><%mm%>"
          .PartFixedWidth(exNoteStart) = G2antt1.Chart.UnitWidth
          .PartFixedHeight(exNoteStart) = 36
          .PartShadow(exNoteStart) = True
        End With
      End If
    End If
```

```
    .EndUpdate
  End With
End Sub
```

Use the Visible property to show or hide the entire note. Use the ShowNotes property to show or hide the notes in the control. Use the Remove method to remove a note from the Notes collection.

property Note.PartVOffset(Part as NotePartEnum) as Long

Specifies the vertical offset to display the part of the note.

Type	Description
Part as NotePartEnum	A NotePartEnum expression that specifies the part to add vertical offset
Long	A long expression that specifies the vertical offset of the part of the note.

The PartVOffset(exNoteStart) property indicates the vertical offset relative to the point being referred by the RelativePosition property. The PartVOffset(exNoteEnd) property indicates the vertical offset relative to the starting part of the note. The PartCanMove property specifies whether the user can move at runtime the part of the note.

The RelativePosition property indicates a float value between 0 and 1 relative to the object DATE or BAR associated with as follow:

- if the note is associated with a DATE (the Key parameter of the Add method is of DATE type), 0 indicates the starting of the unit where the DATE is, 1 indicates the finish of the time unit where the DATE is, and 0.5 indicates the center of the unit.
- if the note is associated with a BAR (the Key parameter of the Add method is of DATE type), 0 indicates the starting point of the bar, 1 indicates the ending point of the bar, 0.5 indicates the middle of the bar.

The PartHOffset / PartVOffset properties indicate the horizontal / vertical offset as follow:

- for exNoteStart, the PartHOffset / PartVOffset properties indicate the offset relative to the point referred by the RelativePosition property.
- for exNoteEnd, the PartHOffset / PartVOffset properties indicate the offset relative to the exNoteStart

The moving cursor is shown on the movable part if:

- the part is visible and it means the PartVisible property is True, and it is visible on the screen (has the PartText not empty, or has the both PartFixedWidth / PartFixedHeight properties not zero)
- the exNoteStart is movable, and it means that the PartCanMove(exNoteStart) property is True
- the exNoteEnd is movable, and it means that the PartCanMove(exNoteStart) OR PartCanMove(exNoteEnd) property is True.

property Note.RelativePosition as Variant

Specifies the position of the note relative to associated object.

Type	Description
Variant	A **numeric** expression that specifies the relative position, or a **string** expression that starts with "S" or "E" following a numeric expression, as explained in the description.

By default, the RelativePosition property is 0.5 if a note is associated with a DATE, and 0, if the note is associated with a BAR. The Key parameter of the Add method determines whether the bar is associated with a DATE or with a BAR. The RelativePosition property always specifies the position of the starting part of the note relative to the DATE or BAR being associated. Use the PartHOffset / PartVOffset property to specify the horizontal / vertical offset relative to the start or end part. The PartCanMove property specifies whether the user can move at runtime the part of the note.

Using notes feature of the control you can associate notes or boxes to dates or bars as follow:

- associate a note or a box to a DATE in the chart control. The note is shown relative to the date in the chart area, so once the chart is scrolled the note is moved or repositioned accordingly to the date. The vertical position of the note is determined by the item that hosts the note, and the PartVOffset property. The RelativePosition is a number expression that determines whether the note is associated with the start of the date, end of the date, or a percent.
- associate a note or a box relative to a BAR in the chart control. The note is shown relative to the bar in the chart area, so once the bar is moved or resized the note is shown accordingly to the bar. The vertical position of the note is determined by the item that hosts the note, and the PartVOffset property. In this case, the type of the RelativePosition property determines how the note is related to the bar using the following rules:
  - is a numeric expression, the RelativePosition determines the position to display the note relative to the range of the bar, determined by the starting */ItemBar(exBarStart)/* and ending */ItemBar(exBarEnd)/* point of the bar. For instance, if the RelativePosition property is 0.5 the note is being displayed in the middle of the bar, while if the RelativePosition property is 1, the note is positioned at the end of the bar. If the user resizes the bar, its range is changed, so the note is moved accordingly.
  - a string expression that starts with S or E being followed by numeric expression (sample "E-1", "E-0.5", "S+2") it determines the note to be relative to the starting point of the bar if starts with **S**, or relative to the end of the bar if it starts with E. The numeric expression that follows to S or E determines the number of days to

be positioned the note. For instance, the "E-1", indicates that the note is positioned one day before bar ends. If the user resizes the ending point of the bar, the note will always be displayed one day before it ends, and so on. The "S+2" , indicates that the note will be displayed 2 days after bar starts, and if the user resizes the bar, the note will always be displayed 2 days after the starting point of the bar. The "E-0.5" indicates that the note will be displayed 12 hours (1/2 or 0.5 from a day) before bar ends.

If The RelativePosition property indicates a **numeric value** the note is displayed relative to the DATE or BAR as follow:

- if the note is associated with a DATE (the Key parameter of the Add method is of DATE type), 0 indicates the starting of the unit where the DATE is, 1 indicates the finish of the time unit where the DATE is, and 0.5 indicates the center of the unit.
- if the note is associated with a BAR (the Key parameter of the Add method is of DATE type), 0 indicates the starting point of the bar, 1 indicates the ending point of the bar, 0.5 indicates the middle of the bar.

If The RelativePosition property indicates a **string expression** that starts with S or E the note is displayed relative to the BAR (ONLY) as follow:

- starts with "S" it indicates that the note is related to starting point of the bar */ItemBar(exBarStart)/*, while the following number in the string expression determines the number of days to display the note relative to the starting point of the bar. For instance, the "S+3" indicates that the note is displayed 3 days later after the bar starts. The note is moved automatically once the starting point of the bar is updated.
- starts with "E" it indicates that the note is related to ending point of the bar */ItemBar(exBarEnd)/*, while the following number in the string expression determines the number of days to display the note relative to the ending point of the bar. For instance, the "E-1" indicates that the note is displayed 1 day before bar ends. The note is moved automatically once the ending point of the bar is updated.

The PartHOffset / PartVOffset properties indicate the horizontal / vertical offset as follow:

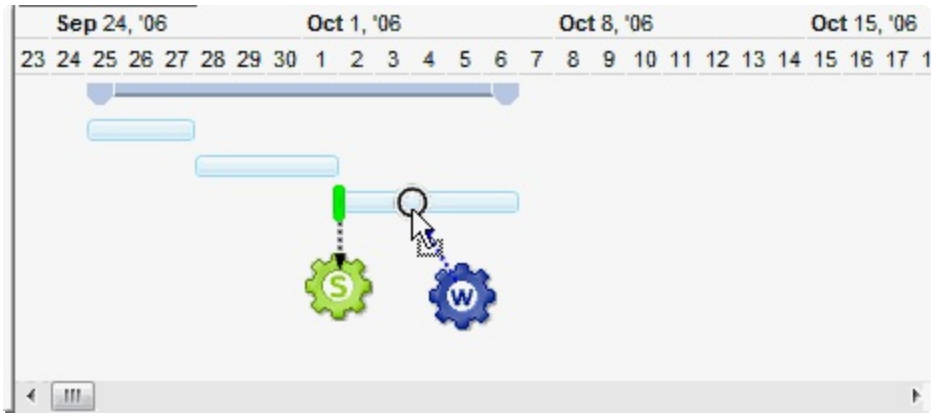
- for exNoteStart, the PartHOffset / PartVOffset properties indicate the offset relative to the point referred by the RelativePosition property.
- for exNoteEnd, the PartHOffset / PartVOffset properties indicate the offset relative to the exNoteStart

The moving cursor is shown on the movable part if:

- the part is visible and it means the PartVisible property is True, and it is visible on the screen (has the PartText not empty, or has the both PartFixedWidth / PartFixedHeight properties not zero)

- the exNoteStart is movable, and it means that the PartCanMove(exNoteStart) property is True
- the exNoteEnd is movable, and it means that the PartCanMove(exNoteStart) OR PartCanMove(exNoteEnd) property is True.

The following screen shows shows how the notes can be moved by the user:

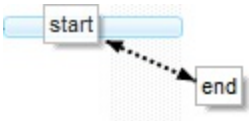


property Note.ShowLink as NoteLinkTypeEnum

Retrieves or sets a value that indicates the link between parts of the note.

Type	Description
NoteLinkTypeEnum	A NoteLinkTypeEnum combination that determines whether the link is shown or hidden, whether the start to end direction is shown, whether the end to start direction is shown, and so on/

The ShowLink property specifies whether the note shows or hides the link between parts of the notes. Use the PartVisible property to specify whether the start or ending part of the note is visible or hidden. The LinkStyle property determines the style of the link between parts of the note. The LinkColor property specifies the color of the link between parts of the notes, while the LinkWidth property determines the width of the link between parts of the notes.



The link between parts of the note is shown

- if the ShowLink property includes the exNoteLinkVisible flag,
- LinkWidth property is greater than 0,
- the start and end part of the note do not intersect.

property Note.Text as String

Specifies the HTML caption being shown in the first visible part of the note.

Type	Description
String	A String expression that specifies the HTML text to be displayed on the ending part of the note. The Text parameter supports HTML tags as well as Chart Tags such as <%dd%> that displays the day in 2 digits, and so on like described bellow. Use the Images method to specify a list of icons that can be displayed in the control using the <img> tag. Use the HTMLPicture property to add custom- size pictures to be used in the HTML captions using the <img> tag.

The Text property is just a shortcut function for PartText property. Use the Text parameter of the Add method to specify the text on the ending part on adding the note. The Text property specifies the PartText for exNoteStart part if visible, else it specifies the PartText for exNoteEnd. The PartForeColor property to specify the part's foreground color. Use the <bgcolor> HTML tag in the PartText property to specify parts of the note's caption with different background colors. The PartAlignment property specifies the text's alignment in the part of the note.

If you need to display a picture only, you need:

- if require icons, use the Images method to specify a list of icons that can be displayed in the control
- if require custom size pictures, use the HTMLPicture property to add custom- size pictures
- set the PartVisible property to show or hide the note's starting or ending part.
- use the ClearPartBackColor method to clear the part's background
- set the PartBorderSize property is on 0
- set the PartShadow property on False

The Text property supports the following:

- <%**d**%> - Day of the month in one or two numeric digits, as needed (1 to 31).
- <%**dd**%> - Day of the month in two numeric digits (01 to 31).
- <%**d1**%> - First letter of the weekday (S to S). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**d2**%> - First two letters of the weekday (Su to Sa). (Use the WeekDays property to specify the name of the days in the week)

- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**d3**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**ddd**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays property to specify the name of the days in the week). You can use the <%**loc_ddd**%> that indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%**loc_ddd**%> - Indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%**dddd**%> - Full name of the weekday (Sunday to Saturday). (Use the WeekDays property to specify the name of the days in the week). You can use the <%**loc_dddd**%> that indicates day of week as its full name using the current user regional and language settings.
- <%**loc_dddd**%> - Indicates day of week as its full name using the current user regional and language settings.
- <%**i**%> - Displays the number instead the date. For instance, you can display numbers as 1000, 1001, 1002, 1003, instead dates. (the valid range is from -647,434 to 2,958,465)
- <%**w**%> - Day of the week (1 to 7).
- <%**ww**%> - Week of the year (1 to 53).
- <%**m**%> - Month of the year in one or two numeric digits, as needed (1 to 12).
- <%**mr**%> - Month of the year in Roman numerals, as needed (I to XII).
- <%**mm**%> - Month of the year in two numeric digits (01 to 12).
- <%**m1**%> - First letter of the month (J to D). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**m2**%> - First two letters of the month (Ja to De). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**m3**%> - First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**mmm**%> - First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmm**%> that indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%**loc_mmm**%> - Indicates month as a three-letter abbreviation using the current user regional and language settings.

- <%**mmmm**%> - Full name of the month (January to December). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmmm**%> that indicates month as its full name using the current user regional and language settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user regional and language settings.
- <%**q**%> - Date displayed as the quarter of the year (1 to 4).
- <%**y**%> - Number of the day of the year (1 to 366).
- <%**yy**%> - Last two digits of the year (01 to 99).
- <%**yyyy**%> - Full year (0100 to 9999).
- <%**hy**%> - Date displayed as the half of the year (1 to 2).
- <%**loc_gg**%> - Indicates period/era using the current user regional and language settings.
- <%**loc_sdate**%> - Indicates the date in the short format using the current user regional and language settings.
- <%**loc_ldate**%> - Indicates the date in the long format using the current user regional and language settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user regional and language settings (/).
- <%**h**%> - Hour in one or two digits, as needed (0 to 23).
- <%**hh**%> - Hour in two digits (00 to 23).
- <%**h12**%> - Hour in 12-hour time format, in one or two digits - [0(12),11]
- <%**hh12**%> - hour in 12-hour time format, in two digits - [00(12),11]
- <%**n**%> - Minute in one or two digits, as needed (0 to 59).
- <%**nn**%> - Minute in two digits (00 to 59).
- <%**s**%> - Second in one or two digits, as needed (0 to 59).
- <%**ss**%> - Second in two digits (00 to 59).
- <%**AM/PM**%> - Twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate. (Use the AMPM property to specify the name of the AM and PM indicators). You can use the <%**loc_AM/PM**%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%**loc_A/P**%> that indicates the one character time marker such as A or P using the current user regional and language settings
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user regional and language settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user regional and language settings.
- <%**loc_time**%> - Indicates the time using the current user regional and language settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%**loc_tsep**%> - indicates the time separator using the current user regional and

language settings (:)
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

The following tags are displayed based on the user's Regional and Language Options:

- <%**loc_sdate**%> - Indicates the date in the short format using the current user settings.
- <%**loc_ldate**%> - Indicates the date in the long format using the current user settings.
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**loc_ddd**%> - Indicates day of week as a three-letters abbreviation using the current user settings.
- <%**loc_dddd**%> - Indicates day of week as its full name using the current user settings.
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**loc_mmm**%> - Indicates month as a three-letters abbreviation using the current user settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user settings.
- <%**loc_gg**%> - Indicates period/era using the current user settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user settings.
- <%**loc_time**%> - Indicates the time using the current user settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user settings.
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user settings.

- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user settings.
- <%**loc_tsep**%> - Indicates the time separator using the current user settings
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

The Text property supports the following built-in HTML tags:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.

  The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using <a ;exp=> or <a ;e64=> anchor tags. The exp/e64 field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption.

  - exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
  - e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.

- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{subscript}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

shadow

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:


outline anti-aliasing

property Note.Visible as Boolean

Specifies whether the note is visible or hidden.

Type	Description
Boolean	A Boolean expression that specifies whether the Note is visible or hidden.

By default, the Visible property is True. Use the Visible property to show or hide a specific note. Use the PartVisible property to specify whether the start or ending part of the note is visible or hidden. When adding new notes, only the ending part of the note is visible. The ShowLink property specifies whether the link between parts of the notes is visible or hidden. Use the ShowNotes property to show or hide the notes in the control. Use the Remove method to remove a note from the Notes collection.

Notes object

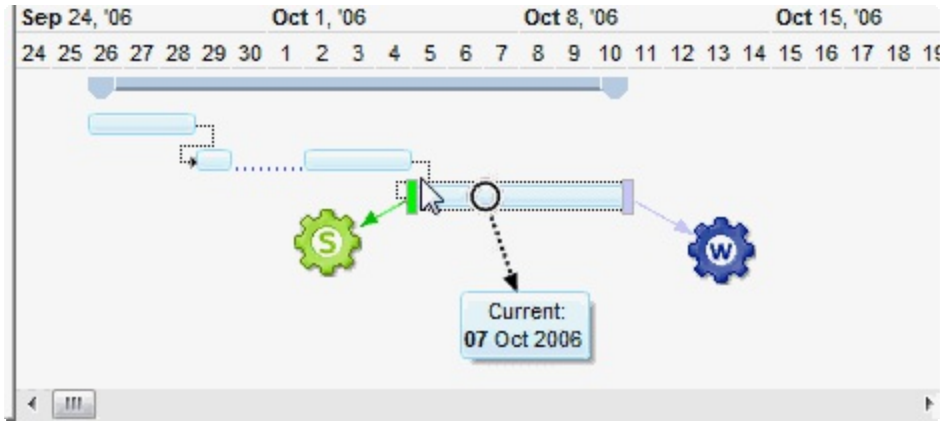
(exn2tes) The Notes collection holds a collection of Note objects. A note can be associated with a DATE in the chart or can be associated to a BAR in the chart. A note is a box that moves together with the related object. For instance, if a note is associated with the starting point of the bar (start date), and the user resizes the bar in the left side (so it changes the starting point of the bar), the related box/note is moved relatively too. The Notes object can be accessed through the Notes property of the Chart object. A Note or a Box can display HTML captions, images, icons, borders, links, and it is fully customizable. The note is composed by two parts, the starting part and end part, that can be linked together. The start part is related to the DATE or to the BAR, while the end part is related to the start part of the note, such us if the start part is moved, the end part is relatively moved. The user can move the end part around the start part, while the start part remains unchanged, or can move so the entire box is moved relatively to the object (DATE or BAR). Use the Items.ItemBar(exBarCaption) property to assign a HTML text, icons, pictures to a bar. Use the Items.ItemBar(exBarExtraCaption) property to add or associate extra captions to a bar.

Using notes feature of the control you can associate notes or boxes to dates or bars as follow:

- associate a note or a box to a DATE in the chart control. The note is shown relative to the date in the chart area, so once the chart is scrolled the note is moved or repositioned accordingly to the date. The vertical position of the note is determined by the item that hosts the note, and the PartVOffset property. The RelativePosition is a number expression that determines whether the note is associated with the start of the date, end of the date, or a percent.
- associate a note or a box relative to a BAR in the chart control. The note is shown relative to the bar in the chart area, so once the bar is moved or resized the note is shown accordingly to the bar. The vertical position of the note is determined by the item that hosts the note, and the PartVOffset property. In this case, the type of the RelativePosition property determines how the note is related to the bar using the following rules:
  - is a numeric expression, the RelativePosition determines the position to display the note relative to the range of the bar, determined by the starting */ItemBar(exBarStart)/* and ending */ItemBar(exBarEnd)/* point of the bar. For instance, if the RelativePosition property is 0.5 the note is being displayed in the middle of the bar, while if the RelativePosition property is 1, the note is positioned at the end of the bar. If the user resizes the bar, its range is changed, so the note is moved accordingly.
  - a string expression that starts with S or E being followed by numeric expression (sample "E-1", "E-0.5", "S+2") it determines the note to be relative to the starting point of the bar if starts with **S**, or relative to the end of the bar if it starts with E.

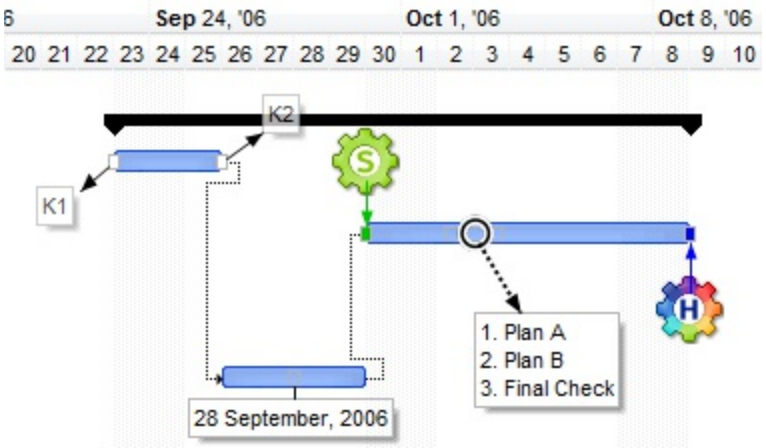
The numeric expression that follows to S or E determines the number of days to be positioned the note. For instance, the "E-1", indicates that the note is positioned one day before bar ends. If the user resizes the ending point of the bar, the note will always be displayed one day before it ends, and so on. The "S+2" , indicates that the note will be displayed 2 days after bar starts, and if the user resizes the bar, the note will always be displayed 2 days after the starting point of the bar. The "E-0.5" indicates that the note will be displayed 12 hours (1/2 or 0.5 from a day) before bar ends.

The following animated screen shows some of capabilities for notes:



The following screen shot shows notes//boxes associated to bars:

The following screen shot shows notes//boxes associated to bars:



The Notes collection supports the following properties and methods:

Name	Description
Add	Adds a new note/box to the control and returns a reference to the newly created object.
Clear	Removes all notes in the control.
ClipTo	Specifies the region of the chart to clip the notes.
Count	Returns the number of objects in a collection.
Item	Returns a specific note/box.
Remove	Removes a specific note/box from the collection.

method Notes.Add (ID as Variant, Item as Variant, Key as Variant, Text as String)

Adds a new note/box to the control and returns a reference to the newly created object.

Type	Description
ID as Variant	An Unique identifier that specifies the ID of the note being added. Use this identifier to access later the node or the box.
Item as Variant	A long expression that specifies the handle of the item where the note is assigned. The ItemFromPoint property retrieves the handle of the item from the cursor.
Key as Variant	A VARIANT expression that specifies the object to relate the note as follows: • Key parameter is of Date type, it indicates the DATE in the chart to associate the note. By default, the RelativePosition property is 0.5, which indicates the center of the unit where the DATE is (0, means the start unit, while the 1 is the end of the unit, and so on). The DateFromPoint property retrieves the date from the point. By default, If a note is associated to a DATE, the RelativePostion property is 0.5, it displays only the ending part of the note, and the ending part of the note is not movable. • Key parameter is not of Date type, it indicates the Key of the BAR to associate the note (The Item and the Key indicates the bar to associate the note). By default, the RelativePosition property is 0, which indicates the starting point of the bar (0, means the starting point of the bar, while the 1 is the ending point of the bar, 0.5 indicates the middle of the bar, and so on). The BarFromPoint property retrieves the key of the bar from the cursor. By default, If a note is associated to a BAR, the RelativePostion property is 0, it displays only the ending part of the note, and the ending part of the note is not movable. Also, the direction from start to end part is visible. By default, the starting part of the note is not visible, so only the ending part of the note is visible.

Text as String	A String expression that specifies the HTML text to be displayed on the ending part of the note. The Text parameter supports HTML tags as well as Chart Tags such as <%dd%> that displays the day in 2 digits, and so on like described bellow. Use the Images method to specify a list of icons that can be displayed in the control using the <img> tag. Use the HTMLPicture property to add custom- size pictures to be used in the HTML captions using the <img> tag.

Return	Description
Note	A Note object being created. Use the NoteFromPoint property to access the Note from the cursor.

The Add method adds a note or a box associated with a DATE or a BAR in the chart. The type of the Key parameter specifies when a DATE or a BAR is being associated with the note. By default, the starting part of the note is not visible, so use the PartVisible property to show or hide any part of the note. The PartCanMove property specifies whether the use can move the part. The PartText property indicates the HTML text to display in the part. The RelativePosition property always specifies the position of the starting part of the note relative to the DATE or BAR being associated. Use the PartHOffset / PartVOffset property to specify the horizontal / vertical offset relative to the start or end part. Use the exBarCaption property to specify the caption for a bar. Use the exBarExtraCaption property to assign more extra captions to a bar.

The Text parameter / PartText property supports the following:

- <%**d**%> - Day of the month in one or two numeric digits, as needed (1 to 31).
- <%**dd**%> - Day of the month in two numeric digits (01 to 31).
- <%**d1**%> - First letter of the weekday (S to S). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**d2**%> - First two letters of the weekday (Su to Sa). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**d3**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays property to specify the name of the days in the week)
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**ddd**%> - First three letters of the weekday (Sun to Sat). (Use the WeekDays property to specify the name of the days in the week). You can use the <%**loc_ddd**%> that indicates the day of week as a three-letter abbreviation using the

current user regional and language settings.

- <%**loc_ddd**%> - Indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%**dddd**%> - Full name of the weekday (Sunday to Saturday). (Use the WeekDays property to specify the name of the days in the week). You can use the <%**loc_dddd**%> that indicates day of week as its full name using the current user regional and language settings.
- <%**loc_dddd**%> - Indicates day of week as its full name using the current user regional and language settings.
- <%**i**%> - Displays the number instead the date. For instance, you can display numbers as 1000, 1001, 1002, 1003, instead dates. (the valid range is from -647,434 to 2,958,465)
- <%**w**%> - Day of the week (1 to 7).
- <%**ww**%> - Week of the year (1 to 53).
- <%**m**%> - Month of the year in one or two numeric digits, as needed (1 to 12).
- <%**mr**%> - Month of the year in Roman numerals, as needed (I to XII).
- <%**mm**%> - Month of the year in two numeric digits (01 to 12).
- <%**m1**%> - First letter of the month (J to D). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**m2**%> - First two letters of the month (Ja to De). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**m3**%> - First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year)
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**mmm**%> - First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmm**%> that indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%**loc_mmm**%> - Indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%**mmmm**%> - Full name of the month (January to December). (Use the MonthNames property to specify the name of the months in the year). You can use the <%**loc_mmmm**%> that indicates month as its full name using the current user regional and language settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user regional and language settings.
- <%**q**%> - Date displayed as the quarter of the year (1 to 4).
- <%**y**%> - Number of the day of the year (1 to 366).

- <%**yy**%> - Last two digits of the year (01 to 99).
- <%**yyyy**%> - Full year (0100 to 9999).
- <%**hy**%> - Date displayed as the half of the year (1 to 2).
- <%**loc_gg**%> - Indicates period/era using the current user regional and language settings.
- <%**loc_sdate**%> - Indicates the date in the short format using the current user regional and language settings.
- <%**loc_ldate**%> - Indicates the date in the long format using the current user regional and language settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user regional and language settings (/).
- <%**h**%> - Hour in one or two digits, as needed (0 to 23).
- <%**hh**%> - Hour in two digits (00 to 23).
- <%**h12**%> - Hour in 12-hour time format, in one or two digits - [0(12),11]
- <%**hh12**%> - hour in 12-hour time format, in two digits - [00(12),11]
- <%**n**%> - Minute in one or two digits, as needed (0 to 59).
- <%**nn**%> - Minute in two digits (00 to 59).
- <%**s**%> - Second in one or two digits, as needed (0 to 59).
- <%**ss**%> - Second in two digits (00 to 59).
- <%**AM/PM**%> - Twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate. (Use the AMPM property to specify the name of the AM and PM indicators). You can use the <%**loc_AM/PM**%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%**loc_A/P**%> that indicates the one character time marker such as A or P using the current user regional and language settings
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user regional and language settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user regional and language settings.
- <%**loc_time**%> - Indicates the time using the current user regional and language settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%**loc_tsep**%> - indicates the time separator using the current user regional and language settings (:)
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported

calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

The following tags are displayed based on the user's Regional and Language Options:

- <%**loc_sdate**%> - Indicates the date in the short format using the current user settings.
- <%**loc_ldate**%> - Indicates the date in the long format using the current user settings.
- <%**loc_d1**%> - Indicates day of week as a one-letter abbreviation using the current user settings.
- <%**loc_d2**%> - Indicates day of week as a two-letters abbreviation using the current user settings.
- <%**loc_d3**%> equivalent with <%**loc_ddd**%>
- <%**loc_ddd**%> - Indicates day of week as a three-letters abbreviation using the current user settings.
- <%**loc_dddd**%> - Indicates day of week as its full name using the current user settings.
- <%**loc_m1**%> - Indicates month as a one-letter abbreviation using the current user settings.
- <%**loc_m2**%> - Indicates month as a two-letters abbreviation using the current user settings.
- <%**loc_m3**%> - equivalent with <%**loc_mmm**%>
- <%**loc_mmm**%> - Indicates month as a three-letters abbreviation using the current user settings.
- <%**loc_mmmm**%> - Indicates month as its full name using the current user settings.
- <%**loc_gg**%> - Indicates period/era using the current user settings.
- <%**loc_dsep**%> - Indicates the date separator using the current user settings.
- <%**loc_time**%> - Indicates the time using the current user settings.
- <%**loc_time24**%> - Indicates the time in 24 hours format without a time marker using the current user settings.
- <%**loc_AM/PM**%> - Indicates the time marker such as AM or PM using the current user settings.
- <%**loc_A/P**%> - Indicates the one character time marker such as A or P using the current user settings.
- <%**loc_tsep**%> - Indicates the time separator using the current user settings
- <%**loc_y**%> - Represents the Year only by the last digit, using current regional settings.
- <%**loc_yy**%> - Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%**loc_yyyy**%> - Represents the Year by a full four or five digits, depending on the

calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

The Text parameter / PartText property supports the following built-in HTML tags:

- ** ... ** displays the text in **bold**
- ** ... ** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The *FormatAnchor* property customizes the visual effect for anchor elements.

  The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using <a ;exp=> or <a ;e64=> anchor tags. The exp/e64 field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption.

  - exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
  - e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

  Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using

the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.

- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the

offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{subscript}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}

- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**out** 000000> <fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

shadow

or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

outline anti-aliasing

The following VB sample adds a note associated with the **DATE** from the cursor, when the user double clicks the chart area:

```
Private Sub G2antt1_DblClick(Shift As Integer, X As Single, Y As Single)
  With G2antt1
    .BeginUpdate
    Dim h As Long, c As Long, hit As HitTestInfoEnum
    h = G2antt1.ItemFromPoint(-1, -1, c, hit)
    If (h <> 0) Then
      Dim d As Date
      d = .Chart.DateFromPoint(-1, -1)
      If (d <> 0) Then
        .Chart.Notes.Add d, h, d, "<b><%dd%></b>/<%mm%>/<%yyyy%>"
      End If
    End If
    .EndUpdate
  End With
End Sub
```

The following VB sample adds a note associated with the **BAR** from the cursor, when the user double clicks the chart area:

```
Private Sub G2antt1_DblClick(Shift As Integer, X As Single, Y As Single)
  With G2antt1
    .BeginUpdate
    Dim h As Long, c As Long, hit As HitTestInfoEnum
    h = G2antt1.ItemFromPoint(-1, -1, c, hit)
    If (h <> 0) Then
      Dim k As Variant
      k = .Chart.BarFromPoint(-1, -1)
      If (Not IsEmpty(k)) Then
        .Chart.Notes.Add k, h, k, "start"
      End If
    End If
    .EndUpdate
  End With
End Sub
```

method Notes.Clear ()

Removes all notes in the control.

Type	Description

The Clear method removes all notes in the control. Use the Remove method to remove a specific Note in the collection. Use the Add method to add new notes to the chart area. Use the NoteFromPoint property to access the note from the cursor. Use the ShowNotes property to show or hide the notes in the control. Use the Visible property to show or hide a specific note, or use the PartVisible property to specify whether the start or ending part of the note is visible or hidden.

property Notes.ClipTo as NotesClipToEnum

Specifies the region of the chart to clip the notes.

Type	Description
NotesClipToEnum	A NotesClipToEnum expression that specifies the chart's Notes limits.

By default, the ClipTo property is exNotesClipNone, which indicates that no clipping is performed when displaying the notes. For instance, you can clip the notes to the items section (exNotesClipToItems), when smoothing scroll the control's content using the AutoDrag property on exAutoDragScroll or ScrollBySingleLine property on False. The Add method adds a new note to the chart associated to the bar/task or date. The ShowNotes property indicates whether the chart displays or hides the notes. Use the Visible property to show or hide a specific note, or use the PartVisible property to specify whether the start or ending part of the note is visible or hidden.

property Notes.Count as Long

Returns the number of objects in a collection.

Type	Description
Long	A long expression that specifies the number of notes in the collection.

The Count property counts the number of notes in the control. Use the Add method to add new notes to the chart area. Use the Item property to access a specific member of the notes collection. Use the Remove method to remove a specific Note in the collection. The Clear method removes all notes in the control. Use the NoteFromPoint property to access the note from the cursor.

The following VB sample prints the ID for each note in the control:

```
Dim n As EXG2ANTTLibCtl.Note
For Each n In G2antt1.Chart.Notes
   Debug.Print n.ID
Next
```

property Notes.Item (ID as Variant) as Note

Returns a specific expression.

Type	Description
ID as Variant	A VARIANT expression that specifies the ID of the note being requested. The ID parameter of the Add method indicates the unique identifier of the note being added. If the ID parameter is a long expression between 0 and Count - 1, the Item property gets the Note by its index.
Note	A Note object being accessed.

Use the Item property to access a specific member of the notes collection. The Count property counts the number of notes in the control. Use the Add method to add new notes to the chart area. Use the Remove method to remove a specific Note in the collection. The Clear method removes all notes in the control. Use the NoteFromPoint property to access the note from the cursor.

The following VB sample prints the ID for each note in the control:

```
Dim n As EXG2ANTTLibCtl.Note
For Each n In G2antt1.Chart.Notes
   Debug.Print n.ID
Next
```

method Notes.Remove (ID as Variant)

Removes a specific note/box from the collection.

Type	Description
ID as Variant	A VARIANT expression that specifies the identifier of the Note to be removed.

Use the Remove method to remove a specific Note in the collection. Use the Add method to add new notes to the chart area. The Clear method removes all notes in the control. Use the ShowNotes property to show or hide the notes in the control. Use the Visible property to show or hide a specific note, or use the PartVisible property to specify whether the start or ending part of the note is visible or hidden. The ShowLink property specifies whether the link between parts of the notes is visible or hidden. Use the NoteFromPoint property to access the note from the cursor.

The following VB sample removes the note being double clicked:

```
Private Sub G2antt1_DblClick(Shift As Integer, X As Single, Y As Single)
  With G2antt1
    .BeginUpdate
    Dim n As EXG2ANTTLibCtl.Note
    Set n = .Chart.NoteFromPoint(-1, -1)
    If Not n Is Nothing Then
      .Chart.Notes.Remove n.ID
    End If
    .EndUpdate
  End With
End Sub
```

OleEvent object

The OleEvent object holds information about an event fired by an ActiveX control hosted by in item that was created using the InsertControlItem method.

Name	Description
CountParam	Retrieves the count of the OLE event's arguments.
ID	Retrieves a long expression that specifies the identifier of the event.
Name	Retrieves the original name of the fired event.
Param	Retrieves an OleEventParam object given either the index of the parameter, or its name.
ToString	Retrieves information about the event.

property OleEvent.CountParam as Long

Retrieves the count of the OLE event's arguments.

Type	Description
Long	A long value that indicates the count of the arguments.

The following sample enumerates the arguments of an OLE event when ItemOLEEvent is fired.

```
Private Sub G2antt1_ItemOleEvent(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal Ev As
EXG2ANTTLibCtl.IOleEvent)
  Debug.Print "Event name:" & Ev.Name
  If (Ev.CountParam = 0) Then
    Debug.Print "The event has no arguments."
  Else
    Debug.Print "The event has the following arguments:"
    Dim i As Long
    For i = 0 To Ev.CountParam - 1
      Debug.Print Ev(i).Name; " = " & Ev(i).Value
    Next
  End If
End Sub
```

The following VC sample displays the events that an ActiveX control is firing while it is hosted by an item:

```
#import <exg2antt.dll> rename( "GetItems", "exGetItems" )

static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
   if ( pv )
   {
      if ( pv->vt == VT_ERROR )
         return szDefault;

      COleVariant vt;
      vt.ChangeType( VT_BSTR, pv );
      return V_BSTR( &vt );
```

```
    }
    return szDefault;
}

void OnItemOleEventG2antt1(long Item, LPDISPATCH Ev)
{
    EXG2ANTTLib::IOleEventPtr spEvent( Ev );
    CString strOutput;
    strOutput.Format( "Event's name: %s\n", spEvent->Name.operator const char *() );
    OutputDebugString( strOutput );
    if ( spEvent->CountParam == 0 )
        OutputDebugString( "The event has no parameters." );
    else
    {
        for ( long i = 0; i < spEvent->CountParam; i++ )
        {
            EXG2ANTTLib::IOleEventParamPtr spParam = spEvent->GetParam( COleVariant( i ) );
            strOutput.Format( "Name: %s, Value: %s\n", spParam->Name.operator const char *
(), V2S( &spParam->Value ) );
            OutputDebugString( strOutput );
        }
    }
    OutputDebugString( "" );
}
```

The #import clause is required to get the wrapper classes for IOleEvent and IOleEventParam objects, that are not defined by the MFC class wizard. The same #import statement defines the EXG2ANTTLib namespace that include all objects and types of the control's TypeLibrary. In case your exg2antt.dll library is located to another place than the system folder or well known path, the path to the library should be provided, in order to let the VC finds the type library.

The following VB.NET sample displays the events that an ActiveX control is firing while it is hosted by an item:

```
Private Sub AxG2antt1_ItemOleEvent(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_ItemOleEventEvent) Handles AxG2antt1.ItemOleEvent
    Debug.WriteLine("Event's name: " & e.ev.Name)
    Dim i As Long
```

```
    For i = 0 To e.ev.CountParam - 1
        Dim eP As EXG2ANTTLib.OleEventParam
        eP = e.ev(i)
        Debug.WriteLine("Name: " & e.ev.Name & " Value: " & eP.Value)
    Next
End Sub
```

The following C# sample displays the events that an ActiveX control is firing while it is hosted by an item:

```
private void axG2antt1_ItemOleEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_ItemOleEventEvent e)
{
    System.Diagnostics.Debug.WriteLine( "Event's name: " + e.ev.Name.ToString() );
    for ( int i= 0; i < e.ev.CountParam ; i++ )
    {
        EXG2ANTTLib.IOleEventParam evP = e.ev[i];
        System.Diagnostics.Debug.WriteLine( "Name: " + evP.Name.ToString() + ", Value: " +
evP.Value.ToString() );
    }
}
```

The following VFP sample displays the events that an ActiveX control fires when it is hosted by an item:

```
*** ActiveX Control Event ***
LPARAMETERS item, ev

local s
s = "Event's name: " + ev.Name
for i = 0 to ev.CountParam - 1
        s = s +  "Name: " + ev.Param(i).Name + " ,Value: " + Str(ev.Param(i).Value)
endfor
wait window nowait s
```

property OleEvent.ID as Long

Retrieves a long expression that specifies the identifier of the event.

Type	Description
Long	A Long expression that defines the identifier of the OLE event.

The identifier of the event could be used to identify a specified OLE event. Use the Name property of the OLE Event to get the name of the OLE Event. Use the ToString property to display information about an OLE event. The ToString property displays the idenfier of the event after the name of the event in two [] brackets. For instance, the ToString property gets the "KeyDown[-602](KeyCode/Short* = 9,Shift/Short = 0)" when TAB key is pressed, so the identifier of the KeyDown event being fired by the inside User editor is -602. For instance, tThe following VB sample closes the editor and focus a new column when user presses the TAB key inside an User editor:

```
Private Sub G2antt1_UserEditorOleEvent(ByVal Object As Object, ByVal Ev As
EXG2ANTTLibCtl.IOleEvent, CloseEditor As Boolean, ByVal Item As
EXG2ANTTLibCtl.HITEM, ByVal ColIndex As Long)
  If (Ev.ID = -602) Then  ' KeyDown
    Dim iKey As Long
    iKey = Ev(0).Value
    If iKey = vbKeyTab Then
      With G2antt1
        CloseEditor = True
        .FocusColumnIndex = .FocusColumnIndex + 1
        .SearchColumnIndex = .FocusColumnIndex
      End With
    End If
  End If
```

property OleEvent.Name as String

Retrieves the original name of the fired event.

Type	Description
String	A string expression that indicates the event's name.

Use the ID property to specify a specified even by its identifier. Use the ToString property to display information about fired event such us name, parameters, types and values. Use the CountParam property to count the parameters of an OLE event. Use the Param property to get the event's parameter. Use the Value property to specify the value of the parameter.

The following VB sample enumerates the arguments of an OLE event when ItemOLEEvent is fired.

```
Private Sub G2antt1_ItemOleEvent(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal Ev As
EXG2ANTTLibCtl.IOleEvent)
  Debug.Print "Event name:" & Ev.Name
  If (Ev.CountParam = 0) Then
    Debug.Print "The event has no arguments."
  Else
    Debug.Print "The event has the following arguments:"
    Dim i As Long
    For i = 0 To Ev.CountParam - 1
      Debug.Print Ev(i).Name; " = " & Ev(i).Value
    Next
  End If
End Sub
```

The following VC sample displays the events that an ActiveX control is firing while it is hosted by an item:

```
#import <exg2antt.dll> rename( "GetItems", "exGetItems" )

static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
   if ( pv )
   {
      if ( pv->vt == VT_ERROR )
         return szDefault;
```

```
        COleVariant vt;
        vt.ChangeType( VT_BSTR, pv );
        return V_BSTR( &vt );
    }
    return szDefault;
}

void OnItemOleEventG2antt1(long Item, LPDISPATCH Ev)
{
    EXG2ANTTLib::IOleEventPtr spEvent( Ev );
    CString strOutput;
    strOutput.Format( "Event's name: %s\n", spEvent->Name.operator const char *() );
    OutputDebugString( strOutput );
    if ( spEvent->CountParam == 0 )
        OutputDebugString( "The event has no parameters." );
    else
    {
        for ( long i = 0; i < spEvent->CountParam; i++ )
        {
            EXG2ANTTLib::IOleEventParamPtr spParam = spEvent->GetParam( COleVariant( i ) );
            strOutput.Format( "Name: %s, Value: %s\n", spParam->Name.operator const char *
(), V2S( &spParam->Value ) );
            OutputDebugString( strOutput );
        }
    }
    OutputDebugString( "" );
}
```

The #import clause is required to get the wrapper classes for IOleEvent and IOleEventParam objects, that are not defined by the MFC class wizard. The same #import statement defines the EXG2ANTTLib namespace that include all objects and types of the control's TypeLibrary. In case your exg2antt.dll library is located to another place than the system folder or well known path, the path to the library should be provided, in order to let the VC finds the type library.

The following VB.NET sample displays the events that an ActiveX control is firing while it is hosted by an item:

```
Private Sub AxG2antt1_ItemOleEvent(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_ItemOleEventEvent) Handles AxG2antt1.ItemOleEvent
    Debug.WriteLine("Event's name: " & e.ev.Name)
    Dim i As Long
    For i = 0 To e.ev.CountParam - 1
        Dim eP As EXG2ANTTLib.OleEventParam
        eP = e.ev(i)
        Debug.WriteLine("Name: " & e.ev.Name & " Value: " & eP.Value)
    Next
End Sub
```

The following C# sample displays the events that an ActiveX control is firing while it is hosted by an item:

```
private void axG2antt1_ItemOleEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_ItemOleEventEvent e)
{
    System.Diagnostics.Debug.WriteLine( "Event's name: " + e.ev.Name.ToString() );
    for ( int i= 0; i < e.ev.CountParam ; i++ )
    {
        EXG2ANTTLib.IOleEventParam evP = e.ev[i];
        System.Diagnostics.Debug.WriteLine( "Name: " + evP.Name.ToString() + ", Value: " +
evP.Value.ToString() );
    }
}
```

The following VFP sample displays the events that an ActiveX control fires when it is hosted by an item:

```
*** ActiveX Control Event ***
LPARAMETERS item, ev

local s
s = "Event's name: " + ev.Name
for i = 0 to ev.CountParam - 1
        s = s +  "Name: " + ev.Param(i).Name + " ,Value: " + Str(ev.Param(i).Value)
endfor
wait window nowait s
```

property OleEvent.Param (Item as Variant) as OleEventParam

Retrieves an OleEventParam object given either the index of the parameter, or its name.

Type	Description
Item as Variant	A long expression that indicates the argument's index or a string expression that indicates the argument's name.
OleEventParam	An OleEventParam object that contains the name and the value for the argument.

The following VB sample enumerates the arguments of an OLE event when ItemOLEEvent is fired.

```
Private Sub G2antt1_ItemOleEvent(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal Ev As
EXG2ANTTLibCtl.IOleEvent)
  Debug.Print "Event name:" & Ev.Name
  If (Ev.CountParam = 0) Then
    Debug.Print "The event has no arguments."
  Else
    Debug.Print "The event has the following arguments:"
    Dim i As Long
    For i = 0 To Ev.CountParam - 1
      Debug.Print Ev(i).Name; " = " & Ev(i).Value
    Next
  End If
End Sub
```

The following VC sample displays the events that an ActiveX control is firing while it is hosted by an item:

```
#import <exg2antt.dll> rename( "GetItems", "exGetItems" )

static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
   if ( pv )
   {
      if ( pv->vt == VT_ERROR )
         return szDefault;
```

```
        COleVariant vt;
        vt.ChangeType( VT_BSTR, pv );
        return V_BSTR( &vt );
    }
    return szDefault;
}

void OnItemOleEventG2antt1(long Item, LPDISPATCH Ev)
{
    EXG2ANTTLib::IOleEventPtr spEvent( Ev );
    CString strOutput;
    strOutput.Format( "Event's name: %s\n", spEvent->Name.operator const char *() );
    OutputDebugString( strOutput );
    if ( spEvent->CountParam == 0 )
        OutputDebugString( "The event has no parameters." );
    else
    {
        for ( long i = 0; i < spEvent->CountParam; i++ )
        {
            EXG2ANTTLib::IOleEventParamPtr spParam = spEvent->GetParam( COleVariant( i ) );
            strOutput.Format( "Name: %s, Value: %s\n", spParam->Name.operator const char *
(), V2S( &spParam->Value ) );
            OutputDebugString( strOutput );
        }
    }
    OutputDebugString( "" );
}
```

The #import clause is required to get the wrapper classes for IOleEvent and IOleEventParam objects, that are not defined by the MFC class wizard. The same #import statement defines the EXG2ANTTLib namespace that include all objects and types of the control's TypeLibrary. In case your exg2antt.dll library is located to another place than the system folder or well known path, the path to the library should be provided, in order to let the VC finds the type library.

The following VB.NET sample displays the events that an ActiveX control is firing while it is hosted by an item:

```
Private Sub AxG2antt1_ItemOleEvent(ByVal sender As Object, ByVal e As
```

```
AxEXG2ANTTLib._IG2anttEvents_ItemOleEventEvent) Handles AxG2antt1.ItemOleEvent
    Debug.WriteLine("Event's name: " & e.ev.Name)
    Dim i As Long
    For i = 0 To e.ev.CountParam - 1
        Dim eP As EXG2ANTTLib.OleEventParam
        eP = e.ev(i)
        Debug.WriteLine("Name: " & e.ev.Name & " Value: " & eP.Value)
    Next
End Sub
```

The following C# sample displays the events that an ActiveX control is firing while it is hosted by an item:

```
private void axG2antt1_ItemOleEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_ItemOleEventEvent e)
{
    System.Diagnostics.Debug.WriteLine( "Event's name: " + e.ev.Name.ToString() );
    for ( int i= 0; i < e.ev.CountParam ; i++ )
    {
        EXG2ANTTLib.IOleEventParam evP = e.ev[i];
        System.Diagnostics.Debug.WriteLine( "Name: " + evP.Name.ToString() + ", Value: " +
evP.Value.ToString() );
    }
}
```

The following VFP sample displays the events that an ActiveX control fires when it is hosted by an item:

```
*** ActiveX Control Event ***
LPARAMETERS item, ev

local s
s = "Event's name: " + ev.Name
for i = 0 to ev.CountParam - 1
        s = s +  "Name: " + ev.Param(i).Name + " ,Value: " + Str(ev.Param(i).Value)
endfor
wait window nowait s
```

property OleEvent.ToString as String

Retrieves information about the event.

Type	Description
String	A String expression that shows information about an OLE event. The ToString property gets the information as follows: Name[ID] (Param/Type = Value, Param/Type = Value, ...). For instance, "KeyDown[-602] (KeyCode/Short* = 9,Shift/Short = 0)" indicates that the KeyDown event is fired, with the identifier -602 with two parameters KeyCode as a reference to a short type with the value 8, and Shift parameter as Short type with the value 0.

Use the ToString property to display information about fired event such us name, parameters, types and values. Using the ToString property you can quickly identifies the event that you should handle in your application. Use the ID property to specify a specified even by its identifier. Use the Name property to get the name of the event. Use the Param property to access a specified parameter using its index or its name.

Displaying ToString property during the OLE Event event may show data like follows:

```
MouseMove[-606](Button/Short = 0,Shift/Short = 0,X/Long = 46,Y/Long = 15)
MouseDown[-605](Button/Short = 1,Shift/Short = 0,X/Long = 46,Y/Long = 15)
KeyDown[-602](KeyCode/Short* = 83,Shift/Short = 0)
KeyPress[-603](KeyAscii/Short* = 115)
Change[2]()
KeyUp[-604](KeyCode/Short* = 83,Shift/Short = 0)
MouseUp[-607](Button/Short = 1,Shift/Short = 0,X/Long = 46,Y/Long = 15)
MouseMove[-606](Button/Short = 0,Shift/Short = 0,X/Long = 46,Y/Long = 15)
```

OleEventParam object

The OleEventParam holds the name and the value for an event's argument.

Name	Description
Name	Retrieves the name of the event's parameter.
Value	Retrieves or sets the value of the event's parameter.

property OleEventParam.Name as String

Retrieves the name of the event's parameter.

Type	Description
String	A string expression that indicates the name of the event's parameter.

The following sample enumerates the arguments of an OLE event when ItemOLEEvent is fired.

```
Private Sub G2antt1_ItemOleEvent(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal Ev As
EXG2ANTTLibCtl.IOleEvent)
  Debug.Print "Event name:" & Ev.Name
  If (Ev.CountParam = 0) Then
    Debug.Print "The event has no arguments."
  Else
    Debug.Print "The event has the following arguments:"
    Dim i As Long
    For i = 0 To Ev.CountParam - 1
      Debug.Print Ev(i).Name; " = " & Ev(i).Value
    Next
  End If
End Sub
```

The following VC sample displays the events that an ActiveX control is firing while it is hosted by an item:

```
#import <exg2antt.dll> rename( "GetItems", "exGetItems" )

static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
   if ( pv )
   {
      if ( pv->vt == VT_ERROR )
         return szDefault;

      COleVariant vt;
      vt.ChangeType( VT_BSTR, pv );
```

```
        return V_BSTR( &vt );
    }
    return szDefault;
}

void OnItemOleEventG2antt1(long Item, LPDISPATCH Ev)
{
    EXG2ANTTLib::IOleEventPtr spEvent( Ev );
    CString strOutput;
    strOutput.Format( "Event's name: %s\n", spEvent->Name.operator const char *() );
    OutputDebugString( strOutput );
    if ( spEvent->CountParam == 0 )
        OutputDebugString( "The event has no parameters." );
    else
    {
        for ( long i = 0; i < spEvent->CountParam; i++ )
        {
            EXG2ANTTLib::IOleEventParamPtr spParam = spEvent->GetParam( COleVariant( i ) );
            strOutput.Format( "Name: %s, Value: %s\n", spParam->Name.operator const char *
(), V2S( &spParam->Value ) );
            OutputDebugString( strOutput );
        }
    }
    OutputDebugString( "" );
}
```

The #import clause is required to get the wrapper classes for IOleEvent and IOleEventParam objects, that are not defined by the MFC class wizard. The same #import statement defines the EXG2ANTTLib namespace that include all objects and types of the control's TypeLibrary. In case your exg2antt.dll library is located to another place than the system folder or well known path, the path to the library should be provided, in order to let the VC finds the type library.

The following VB.NET sample displays the events that an ActiveX control is firing while it is hosted by an item:

```
Private Sub AxG2antt1_ItemOleEvent(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_ItemOleEventEvent) Handles AxG2antt1.ItemOleEvent
    Debug.WriteLine("Event's name: " & e.ev.Name)
```

```
    Dim i As Long
    For i = 0 To e.ev.CountParam - 1
        Dim eP As EXG2ANTTLib.OleEventParam
        eP = e.ev(i)
        Debug.WriteLine("Name: " & e.ev.Name & " Value: " & eP.Value)
    Next
End Sub
```

The following C# sample displays the events that an ActiveX control is firing while it is hosted by an item:

```
private void axG2antt1_ItemOleEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_ItemOleEventEvent e)
{
    System.Diagnostics.Debug.WriteLine( "Event's name: " + e.ev.Name.ToString() );
    for ( int i= 0; i < e.ev.CountParam ; i++ )
    {
        EXG2ANTTLib.IOleEventParam evP = e.ev[i];
        System.Diagnostics.Debug.WriteLine( "Name: " + evP.Name.ToString() + ", Value: " +
evP.Value.ToString() );
    }
}
```

The following VFP sample displays the events that an ActiveX control fires when it is hosted by an item:

```
*** ActiveX Control Event ***
LPARAMETERS item, ev

local s
s = "Event's name: " + ev.Name
for i = 0 to ev.CountParam - 1
        s = s +  "Name: " + ev.Param(i).Name + " ,Value: " + Str(ev.Param(i).Value)
endfor
wait window nowait s
```

property OleEventParam.Value as Variant

Retrieves or sets the value of the event's parameter.

Type	Description
Variant	A variant value that indicates the value of the event's parameter.

The following sample enumerates the arguments of an OLE event when ItemOLEEvent is fired.

```
Private Sub G2antt1_ItemOleEvent(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal Ev As
EXG2ANTTLibCtl.IOleEvent)
  Debug.Print "Event name:" & Ev.Name
  If (Ev.CountParam = 0) Then
    Debug.Print "The event has no arguments."
  Else
    Debug.Print "The event has the following arguments:"
    Dim i As Long
    For i = 0 To Ev.CountParam - 1
      Debug.Print Ev(i).Name; " = " & Ev(i).Value
    Next
  End If
End Sub
```

The following VC sample displays the events that an ActiveX control is firing while it is hosted by an item:

```
#import <exg2antt.dll> rename( "GetItems", "exGetItems" )

static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
   if ( pv )
   {
      if ( pv->vt == VT_ERROR )
         return szDefault;

      COleVariant vt;
      vt.ChangeType( VT_BSTR, pv );
```

```
        return V_BSTR( &vt );
    }
    return szDefault;
}

void OnItemOleEventG2antt1(long Item, LPDISPATCH Ev)
{
    EXG2ANTTLib::IOleEventPtr spEvent( Ev );
    CString strOutput;
    strOutput.Format( "Event's name: %s\n", spEvent->Name.operator const char *() );
    OutputDebugString( strOutput );
    if ( spEvent->CountParam == 0 )
        OutputDebugString( "The event has no parameters." );
    else
    {
        for ( long i = 0; i < spEvent->CountParam; i++ )
        {
            EXG2ANTTLib::IOleEventParamPtr spParam = spEvent->GetParam( COleVariant( i ) );
            strOutput.Format( "Name: %s, Value: %s\n", spParam->Name.operator const char *
(), V2S( &spParam->Value ) );
            OutputDebugString( strOutput );
        }
    }
    OutputDebugString( "" );
}
```

The #import clause is required to get the wrapper classes for IOleEvent and IOleEventParam objects, that are not defined by the MFC class wizard. The same #import statement defines the EXG2ANTTLib namespace that include all objects and types of the control's TypeLibrary. In case your exg2antt.dll library is located to another place than the system folder or well known path, the path to the library should be provided, in order to let the VC finds the type library.

The following VB.NET sample displays the events that an ActiveX control is firing while it is hosted by an item:

```
Private Sub AxG2antt1_ItemOleEvent(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_ItemOleEventEvent) Handles AxG2antt1.ItemOleEvent
    Debug.WriteLine("Event's name: " & e.ev.Name)
```

```
    Dim i As Long
    For i = 0 To e.ev.CountParam - 1
        Dim eP As EXG2ANTTLib.OleEventParam
        eP = e.ev(i)
        Debug.WriteLine("Name: " & e.ev.Name & " Value: " & eP.Value)
    Next
End Sub
```

The following C# sample displays the events that an ActiveX control is firing while it is hosted by an item:

```
private void axG2antt1_ItemOleEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_ItemOleEventEvent e)
{
    System.Diagnostics.Debug.WriteLine( "Event's name: " + e.ev.Name.ToString() );
    for ( int i= 0; i < e.ev.CountParam ; i++ )
    {
        EXG2ANTTLib.IOleEventParam evP = e.ev[i];
        System.Diagnostics.Debug.WriteLine( "Name: " + evP.Name.ToString() + ", Value: " +
evP.Value.ToString() );
    }
}
```

The following VFP sample displays the events that an ActiveX control fires when it is hosted by an item:

```
*** ActiveX Control Event ***
LPARAMETERS item, ev

local s
s = "Event's name: " + ev.Name
for i = 0 to ev.CountParam - 1
        s = s +  "Name: " + ev.Param(i).Name + " ,Value: " + Str(ev.Param(i).Value)
endfor
wait window nowait s
```

ExG2antt events

The Exontrol's ExG2antt component supports the following events:

Name	Description
AddColumn	Fired after a new column has been added.
AddGroupItem	Occurs after a new Group Item has been inserted to Items collection.
AddItem	Occurs after a new Item has been inserted to Items collection.
AddLink	Occurs when the user links two bars using the mouse.
AfterDrawPart	Occurs right after drawing the part of the control.
AfterExpandItem	Fired after an item is expanded (collapsed).
AllowAutoDrag	Occurs when the user drags the item between InsertA and InsertB as child of NewParent.
AllowLink	Notifies at runtime when a link between two bars is possible.
AnchorClick	Occurs when an anchor element is clicked.
BarParentChange	Occurs just before moving a bar from current item to another item.
BarResize	Occurs when a bar is moved or resized.
BarResizing	Occurs when a bar is moving or resizing.
BeforeDrawPart	Occurs just before drawing a part of the control.
BeforeExpandItem	Fired before an item is about to be expanded (collapsed).
ButtonClick	Occurs when user clicks on the cell's button.
CellImageClick	Fired after the user clicks on the image's cell area.
CellStateChanged	Fired after cell's state has been changed.
CellStateChanging	Fired before cell's state is about to be changed.
Change	Occurs when the user changes the cell's content.
ChartEndChanging	Occurs after the chart has been changed.
ChartSelectionChanged	Occurs when the user selects objects in the chart area.
ChartStartChanging	Occurs when the chart is about to be changed.
Click	Occurs when the user presses and then releases the left mouse button over the tree control.

ColumnClick	Fired after the user clicks on column's header.
CreateBar	Fired when the user creates a new bar.
DateChange	Occurs when the first visible date is changed.
DateTimeChanged	Notifies your application that the current time is changed.
DblClick	Occurs when the user dblclk the left mouse button over an object.
Edit	Occurs just before editing the focused cell.
EditClose	Occurs when the edit operation ends.
EditOpen	Occurs when the edit operation starts.
Error	Fired when an internal error occurs.
Event	Notifies the application once the control fires an event.
FilterChange	Occurs when the filter was changed.
FilterChanging	Notifies your application that the filter is about to change.
FocusChanged	Occurs when a cell gets the focus.
FormatColumn	Fired when a cell requires to format its caption.
HistogramBoundsChanged	Occurs when the location and the size of the histogram is changed.
HyperLinkClick	Occurs when the user clicks on a hyperlink cell.
InsideZoom	Notifies your application that a date is about to be magnified.
ItemOleEvent	Fired when an ActiveX control hosted by an item has fired an event.
KeyDown	Occurs when the user presses a key while an object has the focus.
KeyPress	Occurs when the user presses and releases an ANSI key.
KeyUp	Occurs when the user releases a key while an object has the focus.
LayoutChanged	Occurs when column's position or column's size is changed.
MouseDown	Occurs when the user presses a mouse button.
MouseMove	Occurs when the user moves the mouse.
MouseUp	Occurs when the user releases a mouse button.
OffsetChanged	Occurs when the scroll position has been changed.

OLECompleteDrag	Occurs when a source component is dropped onto a target component, informing the source component that a drag action was either performed or canceled
OLEDragDrop	Occurs when a source component is dropped onto a target component when the source component determines that a drop can occur.
OLEDragOver	Occurs when one component is dragged over another.
OLEGiveFeedback	Allows the drag source to specify the type of OLE drag-and-drop operation and the visual feedback.
OLESetData	Occurs on a drag source when a drop target calls the GetData method and there is no data in a specified format in the OLE drag-and-drop DataObject.
OLEStartDrag	Occurs when the OLEDrag method is called.
OversizeChanged	Occurs when the right range of the scroll has been changed.
OverviewZoom	Occurs once the user selects a new time scale unit in the overview zoom area.
RClick	Fired when right mouse button is clicked
RemoveColumn	Fired before deleting a Column.
RemoveItem	Occurs before deleting an Item.
ScrollButtonClick	Occurs when the user clicks a button in the scrollbar.
SelectionChanged	Fired after a new item has been selected.
Sort	Fired when the control sorts a column.
ToolTip	Fired when the control prepares the object's tooltip.
UserEditorClose	Fired the user editor is about to be opened.
UserEditorOleEvent	Occurs when an user editor fires an event.
UserEditorOpen	Occurs when an user editor is about to be opened.
ValidateValue	Occurs before user changes the cell's value.

event AddColumn (Column as Column)

Fired after a new column has been added.

Type	Description
Column as Column	A Column object that's added to the Columns collection.

The AddColumn event is fired after a new column has been inserted to Columns collection. Use the AddColumn event to associate extra data to a new column. Use the Add method to add new columns to Columns collection. Use the ColumnAutoSize property to fit all visible columns in the control's client area.

Syntax for AddColumn event, **/NET** version, on:

C#
```
private void AddColumn(object sender,exontrol.EXG2ANTTLib.Column Column)
{
}
```

VB
```
Private Sub AddColumn(ByVal sender As System.Object,ByVal Column As
exontrol.EXG2ANTTLib.Column) Handles AddColumn
End Sub
```

Syntax for AddColumn event, **/COM** version, on:

C#
```
private void AddColumn(object sender,
AxEXG2ANTTLib._IG2anttEvents_AddColumnEvent e)
{
}
```

C++
```
void OnAddColumn(LPDISPATCH Column)
{
}
```

C++ Builder
```
void __fastcall AddColumn(TObject *Sender,Exg2anttlib_tlb::IColumn *Column)
{
}
```

Delphi
```
procedure AddColumn(ASender: TObject; Column : IColumn);
begin
end;
```

Delphi 8 (.NET only)

```
procedure AddColumn(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_AddColumnEvent);
begin
end;
```

Powe…

```
begin event AddColumn(oleobject Column)
end event AddColumn
```

VB.NET

```
Private Sub AddColumn(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_AddColumnEvent) Handles AddColumn
End Sub
```

VB6

```
Private Sub AddColumn(ByVal Column As EXG2ANTTLibCtl.IColumn)
End Sub
```

VBA

```
Private Sub AddColumn(ByVal Column As Object)
End Sub
```

VFP

```
LPARAMETERS Column
```

Xbas…

```
PROCEDURE OnAddColumn(oG2antt,Column)
RETURN
```

Syntax for AddColumn event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="AddColumn(Column)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function AddColumn(Column)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComAddColumn Variant llColumn
    Forward Send OnComAddColumn llColumn
End_Procedure
```

Visual Objects

```
METHOD OCX_AddColumn(Column) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_AddColumn(COM _Column)
{
}
```

XBasic

```
function AddColumn as v (Column as OLE::Exontrol.G2antt.1::IColumn)
end function
```

dBASE

```
function nativeObject_AddColumn(Column)
return
```

The following VB sample shows how to set the width for all columns:

```
Private Sub G2antt1_AddColumn(ByVal Column As EXG2ANTTLibCtl.IColumn)
    Column.Width = 128
End Sub
```

The following VB.NET sample changes the column's width when a new column is added:

```
Private Sub AxG2antt1_AddColumn(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_AddColumnEvent) Handles AxG2antt1.AddColumn
    e.column.Width = 128
End Sub
```

The following C# sample changes the column's width when a new column is added:

```
private void axG2antt1_AddColumn(object sender,
AxEXG2ANTTLib._IG2anttEvents_AddColumnEvent e)
{
    e.column.Width = 128;
}
```

The following C++ sample changes the column's width when a new column is added:

```
#include "Column.h"
```

```
#include "Columns.h"
void OnAddColumnG2antt1(LPDISPATCH Column)
{
    CColumn column( Column );column.m_bAutoRelease = FALSE;
    column.SetWidth( 128 );
}
```

The following VFP sample changes the column's width when a new column is added:

```
*** ActiveX Control Event ***
LPARAMETERS column

with column
    .Width = 128
endwith
```

event AddGroupItem (Item as HITEM)

Occurs after a new Group Item has been inserted to Items collection.

Type	Description
Item as HITEM	A Long expression that indicates the handle of the grouping items being inserted.

The AddGroupItem event is fired for each new item to be inserted in the Items collection during the grouping. The GroupItem method determines the index of the column that indicates the column being grouped. In other words, the CellCaption(Item,GroupItem(Item)) gets the default caption to be displayed for the grouping item. The Ungroup method removes all grouping items. For instance, when a column gets grouped by, the control sorts by that column, collects the unique values being found, and add a new item for each value found, by adding the items of the same value as children.

During the AddGroupItem event, you can use:

- Items.CellValue(Item, Items.GroupItem(Item)) to update the cell's content or what the grouping item is displaying
- ItemDivider/ItemDividerLine/ItemDividerLineAlignment property to show the grouping item as a divider or as a regular item
- AddBar method to add a new bar to the grouping item
- DefineSummaryBar method to define the bars belonging to a summary bar

Of course, these are just a few of properties you may use, you can use any property related to the item/cell or bar and link.

The AddGroupItem event can be used in any of the followings:

- customize the visual appearance for any grouping item,
- adding new headers or footers for grouping items

Syntax for AddGroupItem event, **/NET** version, on:

C#

```
private void AddGroupItem(object sender,int Item)
{
}
```

VB

```
Private Sub AddGroupItem(ByVal sender As System.Object,ByVal Item As Integer)
Handles AddGroupItem
End Sub
```

Syntax for AddGroupItem event, **/COM** version, on:

C#

```
private void AddGroupItem(object sender,
AxEXG2ANTTLib._IG2anttEvents_AddGroupItemEvent e)
{
}
```

C++

```
void OnAddGroupItem(long Item)
{
}
```

C++ Builder

```
void __fastcall AddGroupItem(TObject *Sender,Exg2anttlib_tlb::HITEM Item)
{
}
```

Delphi

```
procedure AddGroupItem(ASender: TObject; Item : HITEM);
begin
end;
```

Delphi 8 (.NET only)

```
procedure AddGroupItem(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_AddGroupItemEvent);
begin
end;
```

Powe…

```
begin event AddGroupItem(long Item)
end event AddGroupItem
```

VB.NET

```
Private Sub AddGroupItem(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_AddGroupItemEvent) Handles AddGroupItem
End Sub
```

VB6

```
Private Sub AddGroupItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
End Sub
```

VBA

```
Private Sub AddGroupItem(ByVal Item As Long)
End Sub
```

VFP

```
LPARAMETERS Item
```

Xbas…
```
PROCEDURE OnAddGroupItem(oG2antt,Item)
RETURN
```

Syntax for AddGroupItem event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="AddGroupItem(Item)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function AddGroupItem(Item)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComAddGroupItem HITEM llItem
   Forward Send OnComAddGroupItem llItem
End_Procedure
```

Visual Objects
```
METHOD OCX_AddGroupItem(Item) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_AddGroupItem(int _Item)
{
}
```

XBasic
```
function AddGroupItem as v (Item as OLE::Exontrol.G2antt.1::HITEM)
end function
```

dBASE
```
function nativeObject_AddGroupItem(Item)
return
```

event AddItem (Item as HITEM)

Occurs after a new Item has been inserted to Items collection.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item that's inserted to the Items collection.

The AddItem event notifies your application that a new items is inserted. Use the AddItem and InsertItem methods to insert new items to Items collection. Use the InsertControlItem method to add a new item that hosts an ActiveX control. Use the Add method to add new columns to Columns Collection. Use the Def property to specify a common value for all cells in the same column. Use the the AddBar method to add new bars to the newly added item.

If the control's DataSource property is set, the AddItem event occurs as soon as a new record is loaded from the giving recrodset. Also, the AddItem event occurs if the AddNew (method of the ADO.RecordSet object) is performed, if the control's DetectAddNew property is True. If using the CellValue properties during the AddItem event, you must be sure that they are available, or they have the proper values or expected values.

Syntax for AddItem event, **/NET** version, on:

C#
```
private void AddItem(object sender,int Item)
{
}
```

VB
```
Private Sub AddItem(ByVal sender As System.Object,ByVal Item As Integer)
Handles AddItem
End Sub
```

Syntax for AddItem event, **/COM** version, on:

C#
```
private void AddItem(object sender,
AxEXG2ANTTLib._IG2anttEvents_AddItemEvent e)
{
}
```

C++
```
void OnAddItem(long Item)
{
}
```

C++ Builder

```
void __fastcall AddItem(TObject *Sender,Exg2anttlib_tlb::HITEM Item)
{
}
```

Delphi

```
procedure AddItem(ASender: TObject; Item : HITEM);
begin
end;
```

Delphi 8 (.NET only)

```
procedure AddItem(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_AddItemEvent);
begin
end;
```

Powe…

```
begin event AddItem(long Item)
end event AddItem
```

VB.NET

```
Private Sub AddItem(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_AddItemEvent) Handles AddItem
End Sub
```

VB6

```
Private Sub AddItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
End Sub
```

VBA

```
Private Sub AddItem(ByVal Item As Long)
End Sub
```

VFP

```
LPARAMETERS Item
```

Xbas…

```
PROCEDURE OnAddItem(oG2antt,Item)
RETURN
```

Syntax for AddItem event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="AddItem(Item)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function AddItem(Item)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComAddItem HITEM llItem
    Forward Send OnComAddItem llItem
End_Procedure
```

Visual Objects
```
METHOD OCX_AddItem(Item) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_AddItem(int _Item)
{
}
```

XBasic
```
function AddItem as v (Item as OLE::Exontrol.G2antt.1::HITEM)
end function
```

dBASE
```
function nativeObject_AddItem(Item)
return
```

For instance, let's say that we defined the AddItem event such as:

```
Private Sub G2antt1_AddItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
    With G2antt1.Items
        .AddBar Item, "Task", .CellValue(Item, 1), .CellValue(Item, 2)
    End With
End Sub
```

If using the r.AddNew method we MUST use the values to be added as parameters of the AddNew method as in the following sample:

```
r.AddNew Array(0, 1, 2), Array("Task", #1/3/2001#, #1/4/2001#)
```

instead using the following code:

```
r.AddNew
    r(0) = "Task"
```

```
r(1) = #1/1/2001#
r(2) = #1/2/2001#
r.Update
```

which is wrong as the AddItem event is called when the r.AddNew method is performed, and so during the AddItem event, the values for the cells are NOT yet available, as the r(0), r(1), r(2) are filled later then r.AddNew call.

The following VB sample shows how to change the item's foreground color:

```
Private Sub G2antt1_AddItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
   G2antt1.Items.ItemForeColor(Item) = vbBlue
End Sub
```

The following VB sample changes the background color for all cells in the first column:

```
G2antt1.Columns(0).Def(exCellBackColor) = RGB(240, 240, 240)
```

The following C++ sample changes the item's foreground color when a new items is inserted:

```
#include "Items.h"
void OnAddItemG2antt1(long Item)
{
   if ( ::IsWindow( m_g2antt.m_hWnd ) )
   {
      CItems items = m_g2antt.GetItems();
      items.SetItemForeColor( Item, RGB(0,0,255) );
   }
}
```

The following C++ sample changes the background color for all cells in the first column:

```
COleVariant vtBackColor( (long)RGB(240, 240, 240) );
m_g2antt.GetColumns().GetItem( COleVariant( (long) 0 ) ).SetDef( /*exCellBackColor*/ 4,
vtBackColor );
```

The following VB.NET sample changes the item's foreground color when a new items is inserted:

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
```

```
    Dim i As Long
    i = c.R
    i = i + 256 * c.G
    i = i + 256 * 256 * c.B
    ToUInt32 = Convert.ToUInt32(i)
End Function

Private Sub AxG2antt1_AddItem(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_AddItemEvent) Handles AxG2antt1.AddItem
    AxG2antt1.Items.ItemForeColor(e.item) = ToUInt32(Color.Blue)
End Sub
```

The following VB.NET sample changes the background color for all cells in the first column:

```
With AxG2antt1.Columns(0)
    .Def(EXG2ANTTLib.DefColumnEnum.exCellBackColor) = ToUInt32(Color.WhiteSmoke)
End With
```

The following C# sample changes the item's foreground color when a new items is inserted:

```
private UInt32 ToUInt32(Color c)
{
    long i;
    i = c.R;
    i = i + 256 * c.G;
    i = i + 256 * 256 * c.B;
    return Convert.ToUInt32(i);
}

private void axG2antt1_AddItem(object sender,
AxEXG2ANTTLib._IG2anttEvents_AddItemEvent e)
{
    axG2antt1.Items.set_ItemForeColor( e.item, ToUInt32(Color.Blue) );
}
```

The following C# sample changes the background color for all cells in the first column:

```
axG2antt1.Columns[0].set_Def(EXG2ANTTLib.DefColumnEnum.exCellBackColor,
ToUInt32(Color.WhiteSmoke));
```

The following VFP sample changes the item's foreground color when a new items is inserted:

```
*** ActiveX Control Event ***
LPARAMETERS item

with thisform.G2antt1.Items
   .DefaultItem = item
   .ItemForeColor( 0 ) = RGB(0,0,255 )
endwith
```

The following VFP sample changes the background color for all cells in the first column:

```
with thisform.G2antt1.Columns(0)
   .Def( 4 ) = RGB(240,240,240)
endwith
```

For instance, the following VB sample loads an ADO recordset.

```
Dim rs As Object

Private Sub Form_Load()

   Set rs = CreateObject("ADODB.Recordset")
   rs.Open "Orders", "Provider=Microsoft.Jet.OLEDB.3.51;Data Source= D:\Program
Files\Microsoft Visual Studio\VB98\NWIND.MDB", 3 ' Opens the table using static mode

   G2antt1.BeginUpdate
   ' Add the columns
   With G2antt1.Columns
   For Each f In rs.Fields
      .Add f.Name
   Next
   End With

   ' Add the items
   With G2antt1.Items
   rs.MoveFirst
```

```
   While Not rs.EOF
      .InsertItem , rs.Bookmark
      rs.MoveNext
   Wend
End With

   G2antt1.EndUpdate
End Sub

Private Sub G2antt1_AddItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
   Dim i As Integer
   Dim n As Integer
   n = G2antt1.Columns.Count
   With G2antt1.Items
   For i = 0 To n - 1
      .CellValue(Item, i) = rs(i).Value
   Next
   End With
End Sub
```

The following VB sample use the PutItems method to load items to the control:

```
Dim rs As Object

Private Sub Form_Load()

   Set rs = CreateObject("ADODB.Recordset")
   rs.Open "Orders", "Provider=Microsoft.Jet.OLEDB.3.51;Data Source= D:\Program
Files\Microsoft Visual Studio\VB98\NWIND.MDB", 3 ' Opens the table using static mode

   G2antt1.BeginUpdate
   ' Add the columns
   With G2antt1.Columns
   For Each f In rs.Fields
      .Add f.Name
   Next
   End With
```

```
    G2antt1.PutItems rs.getRows()

    G2antt1.EndUpdate
End Sub
```

event AddLink (LinkKey as String)

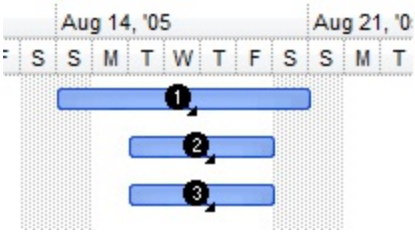
Occurs when the user links two bars using the mouse.

Type	Description
LinkKey as String	A String expression that indicates the key of the link being added.

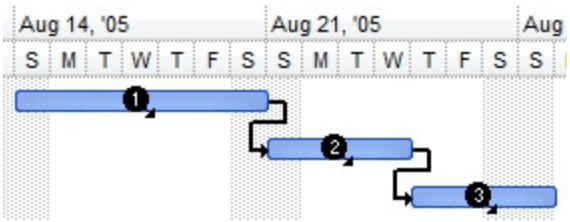
The AddLink event notifies your application that the user links two bars using the mouse. The AllowLink event is called before adding the link, so you can prevent adding new links between specified type of bars. The AddLink event occurs right after the control calls the AddLink method that adds a link between the bars that the user selected. By default, the control provides keys as "Link1", "Link2", ... You can use the Link(exLinkKey) property to change the default key of the link. If the control supports Undo/Redo, you can use the UndoRemoveAction method to remove a specific action from the undo queue. For instance, if you do not need to record RemoveLink during the AddLink event, you can call after RemoveLink the UndoRemoveAction(exChartUndoRedoRemoveLink,1) method to remove the last exChartUndoRedoRemoveLink action from the Undo queue. Use the AddLink method to add links programmatically. ***Use the Link property of the Items object to access the properties and options of the link.*** *Call the Link(exLinkGroupBars) to group two linked bars. This way they move together when a bar is changed.*

The AddLink event may occur only if the AllowLinkBars property is True. The LinkKey parameter indicates the key of the newly added link. Use the RemoveLink method to remove the link, if you don't need certain links to be added. The FirstLink property retrieves the key of the first link in the chart. Use the NextLink property to retrieve the key of the next link, in the chart. Use these properties to enumerate the link in the control. Use the CellValue property to access the cell's value.

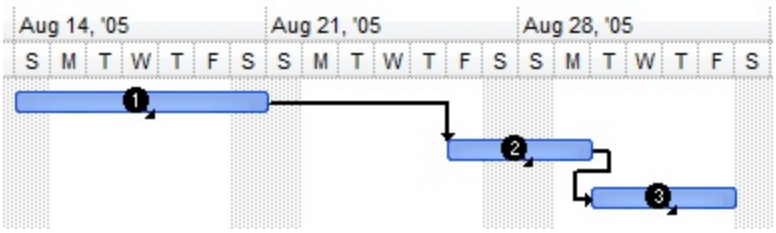
In the following screen shot shows the bars before linking and grouping:



In the following screen shot shows the bars after linking and grouping, as the bar 1 is linked to bar 2, and bar 2 to 3.



In the following screen shot shows the bars once the bar 2 is moved to the right:



Syntax for AddLink event, **/NET** version, on:

C#

```
private void AddLink(object sender,string LinkKey)
{
}
```

VB

```
Private Sub AddLink(ByVal sender As System.Object,ByVal LinkKey As String)
Handles AddLink
End Sub
```

Syntax for AddLink event, **/COM** version, on:

C#

```
private void AddLink(object sender,
AxEXG2ANTTLib._IG2anttEvents_AddLinkEvent e)
{
}
```

C++

```
void OnAddLink(LPCTSTR LinkKey)
{
}
```

C++ Builder

```
void __fastcall AddLink(TObject *Sender,BSTR LinkKey)
{
}
```

Delphi

```
procedure AddLink(ASender: TObject; LinkKey : WideString);
begin
```

```
end;
```

Delphi 8 (.NET only)

```
procedure AddLink(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_AddLinkEvent);
begin
end;
```

Powe…

```
begin event AddLink(string LinkKey)
end event AddLink
```

VB.NET

```
Private Sub AddLink(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_AddLinkEvent) Handles AddLink
End Sub
```

VB6

```
Private Sub AddLink(ByVal LinkKey As String)
End Sub
```

VBA

```
Private Sub AddLink(ByVal LinkKey As String)
End Sub
```

VFP

```
LPARAMETERS LinkKey

```

Xbas…

```
PROCEDURE OnAddLink(oG2antt,LinkKey)
RETURN
```

Syntax for AddLink event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="AddLink(LinkKey)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function AddLink(LinkKey)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComAddLink String llLinkKey
    Forward Send OnComAddLink llLinkKey
```

```
End_Procedure
```

Visual Objects

```
METHOD OCX_AddLink(LinkKey) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_AddLink(str _LinkKey)
{
}
```

XBasic

```
function AddLink as v (LinkKey as C)
end function
```

dBASE

```
function nativeObject_AddLink(LinkKey)
return
```

The following VB sample groups the linked bars:

```
Private Sub G2antt1_AddLink(ByVal LinkKey As String)
   With G2antt1.Items
      .Link(LinkKey, exLinkGroupBars) = GroupBarsOptionsEnum.exPreserveBarLength +
GroupBarsOptionsEnum.exFlexibleInterval +
GroupBarsOptionsEnum.exIgnoreOriginalInterval
   End With
End Sub
```

The following VB sample changes the style of the link if certain condition is met (in this sample, the cell on the first column is called "Root"):

```
Private Sub G2antt1_AddLink(ByVal LinkKey As String)
   With G2antt1.Items
      If .CellValue(.Link(LinkKey, exLinkStartItem), 0) = "Root" Then
         .Link(LinkKey, exLinkStyle) = exLinkSolid
         .Link(LinkKey, exLinkWidth) = 2
         .Link(LinkKey, exLinkColor) = RGB(255, 0, 0)
      End If
   End With
End Sub
```

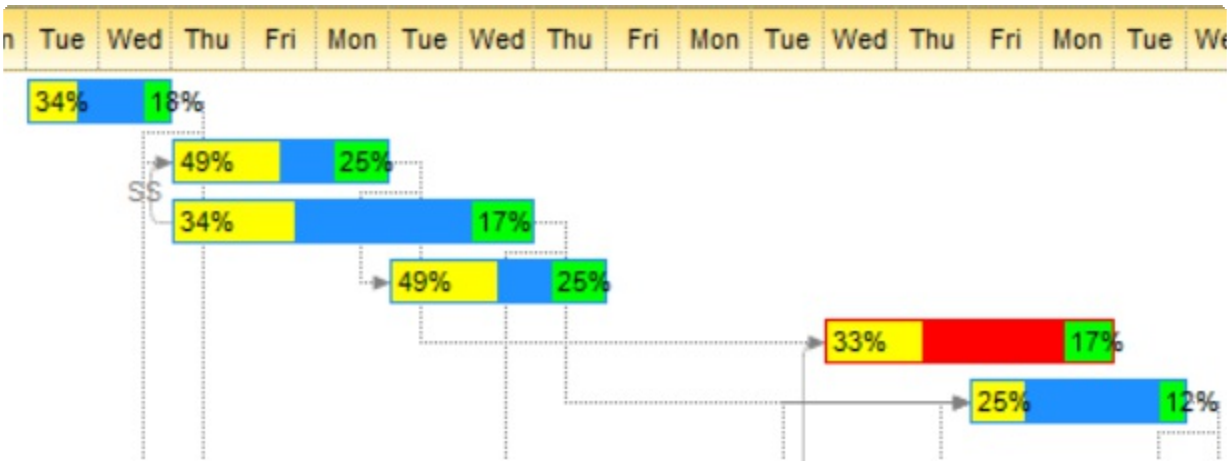
event AfterDrawPart (Part as DrawPartEnum, hDC as Long, X as Long, Y as Long, Width as Long, Height as Long)

Occurs right after drawing the part of the control.

Type	Description
Part as DrawPartEnum	A Part being painted. If the Part parameter is exOwnerDrawBar, the DrawPartItem property specifies the handle of the item that hosts the "OwnerDraw" bar, while the DrawPartKey property specifies the key of the bar to be painted. Use the Add or Copy method to add an "OwnerDraw" bar
hDC as Long	A long expression that specifies the handle of the device context where you can draw. The /NET or /WPF assembly provides a System.Drawing.Graphics object instead hDC parameter
X as Long	A long expression that specifies the left coordinate of the rectangle where the paint should occur. The /NET or /WPF assembly provides a System.Drawing.Rectangle instead (X, Y, Width, Height).
Y as Long	A long expression that specifies the top coordinate of the rectangle where the paint should occur. The /NET or /WPF assembly provides a System.Drawing.Rectangle instead (X, Y, Width, Height).
Width as Long	A long expression that specifies the width of the rectangle where the paint should occur. The /NET or /WPF assembly provides a System.Drawing.Rectangle instead (X, Y, Width, Height).
Height as Long	A long expression that specifies the height of the rectangle where the paint should occur. The /NET or /WPF assembly provides a System.Drawing.Rectangle instead (X, Y, Width, Height).

The BeforeDrawPart and AfterDrawPart events occur when different parts of the control requires to be drawn. Use the BeforeDrawPart and AfterDrawPart events to add your custom drawing to be shown in the component. Use the BeforeDrawPart event to perform your own drawing before the default drawing, canceling the default drawing, or changing the area being assigned to the part part when painting. Use the AfterDrawPart event to perform your own drawing after default painting occurs. The /NET Assembly provides instead hDC and (X,Y,Width,Height) parameters a Graphics object and a Rectangle object, the last being passed by reference. Use the HistogramBoundsChanged event to notify your

application when the left part of the histogram is resized, so inside controls must be re-positioned.



Currently, the control's owner-draw feature allows you to customize the visual-appearance for bars/tasks or control's histogram part.

The control provides the following events:

- BeforeDrawPart, occurs just before drawing a part of the control. For instance, you need to customize the part's background
- AfterDrawPart, occurs right after drawing the part of the control. For instance, you need to customize the part's foreground

Shortly, you can customize the drawing of control's part before and after default-drawing.

For instance, let's say that you need to specify a different background color, for "task" bars, while still keeping the bar's pattern color, which is specified by the bar's exBarColor property. In order to perform owner-draw for bars you need:

- defines a new type of bar called "OwnerDraw", which can be a new bar or a copy of any existing bars (only bars of "OwnerDraw" type fires BeforeDrawPart and AfterDrawPart events when they require to be painted)
- overrides the BeforeDrawPart and / or AfterDrawPart events

During the BeforeDrawPart and AfterDrawPart events, while Part event parameter is exOwnerDrawBar, the control's DrawPartItem and DrawPartKey provides the handle of the item and the key of the bar being drawn. You can use the Items.ItemBar(DrawPartItem, DrawPartKey, BarProperty) property to access any property related to the bar being painted. In the same manner you can use the Items.CellValue/Items.CellCaption properties to access any value/caption from the list/columns part of the control.

Syntax for AfterDrawPart event, **/NET** version, on:

C#
```
private void AfterDrawPart(object sender,exontrol.EXG2ANTTLib.DrawPartEnum
```

```
Part,int hDC,int X,int Y,int Width,int Height)
{
}
```

VB

```
Private Sub AfterDrawPart(ByVal sender As System.Object,ByVal Part As
exontrol.EXG2ANTTLib.DrawPartEnum,ByVal hDC As Integer,ByVal X As
Integer,ByVal Y As Integer,ByVal Width As Integer,ByVal Height As Integer) Handles
AfterDrawPart
End Sub
```

Syntax for AfterDrawPart event, **/COM** version, on:

C#

```
private void AfterDrawPart(object sender,
AxEXG2ANTTLib._IG2anttEvents_AfterDrawPartEvent e)
{
}
```

C++

```
void OnAfterDrawPart(long Part,long hDC,long X,long Y,long Width,long Height)
{
}
```

C++ Builder

```
void __fastcall AfterDrawPart(TObject *Sender,Exg2anttlib_tlb::DrawPartEnum
Part,long hDC,long X,long Y,long Width,long Height)
{
}
```

Delphi

```
procedure AfterDrawPart(ASender: TObject; Part : DrawPartEnum;hDC : Integer;X :
Integer;Y : Integer;Width : Integer;Height : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure AfterDrawPart(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_AfterDrawPartEvent);
begin
end;
```

Powe…

```
begin event AfterDrawPart(long Part,long hDC,long X,long Y,long Width,long
Height)
```

```
end event AfterDrawPart
```

VB.NET

```
Private Sub AfterDrawPart(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_AfterDrawPartEvent) Handles AfterDrawPart
End Sub
```

VB6

```
Private Sub AfterDrawPart(ByVal Part As EXG2ANTTLibCtl.DrawPartEnum,ByVal
hDC As Long,ByVal X As Long,ByVal Y As Long,ByVal Width As Long,ByVal Height
As Long)
End Sub
```

VBA

```
Private Sub AfterDrawPart(ByVal Part As Long,ByVal hDC As Long,ByVal X As
Long,ByVal Y As Long,ByVal Width As Long,ByVal Height As Long)
End Sub
```

VFP

```
LPARAMETERS Part,hDC,X,Y,Width,Height
```

Xbas…

```
PROCEDURE OnAfterDrawPart(oG2antt,Part,hDC,X,Y,Width,Height)
RETURN
```

Syntax for AfterDrawPart event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="AfterDrawPart(Part,hDC,X,Y,Width,Height)"
LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function AfterDrawPart(Part,hDC,X,Y,Width,Height)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComAfterDrawPart OLEDrawPartEnum llPart Integer llhDC Integer
llX Integer llY Integer llWidth Integer llHeight
    Forward Send OnComAfterDrawPart llPart llhDC llX llY llWidth llHeight
End_Procedure
```

Visual Objects

```
METHOD OCX_AfterDrawPart(Part,hDC,X,Y,Width,Height) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_AfterDrawPart(int _Part,int _hDC,int _X,int _Y,int _Width,int _Height)
{
}
```

XBasic

```
function AfterDrawPart as v (Part as OLE::Exontrol.G2antt.1::DrawPartEnum,hDC as N,X as N,Y as N,Width as N,Height as N)
end function
```

dBASE

```
function nativeObject_AfterDrawPart(Part,hDC,X,Y,Width,Height)
return
```

The following VB/NET sample (/NET Assembly) shows how you can divide a bar in three colors with dynamic percentages (percentage of colors can be different for each bar):

```
Private Sub exg2antt1_AfterDrawPart(ByVal sender As Object, ByVal Part As exontrol.EXG2ANTTLib.DrawPartEnum, ByVal G As System.Drawing.Graphics, ByVal Rect As System.Drawing.Rectangle) Handles exg2antt1.AfterDrawPart
    Rect.Inflate(-1, -1)
    Dim left As Rectangle = New Rectangle(Rect.Location, Rect.Size), right As Rectangle = New Rectangle(Rect.Location, Rect.Size)
    left.Width = left.Width / (2 + exg2antt1.DrawPartItem Mod 3)
    right.Width = left.Width / 2
    right.X = Rect.Right - right.Width
    Using fmt As StringFormat = New StringFormat(StringFormatFlags.NoClip Or StringFormatFlags.NoWrap)
        fmt.Trimming = StringTrimming.None
        fmt.LineAlignment = StringAlignment.Center
        G.FillRectangle(Brushes.Yellow, left)
        G.DrawString(String.Format("{0:0}", 100 * (left.Width / CDbl(Rect.Width))) + "%", exg2antt1.Font, Brushes.Black, left, fmt)
        G.FillRectangle(Brushes.Lime, right)
        G.DrawString(String.Format("{0:0}", 100 * (right.Width / CDbl(Rect.Width))) + "%",
```

```
exg2antt1.Font, Brushes.Black, right, fmt)
   End Using
End Sub
```

Prior to this code, you can call the following:

```
.Chart.Bars.Add("OwnerDraw").Color = Color.DodgerBlue
.Items.set_ItemBar(0, "<*>", exontrol.EXG2ANTTLib.ItemBarPropertyEnum.exBarName,
"OwnerDraw")
```

that adds the "OwnerDraw" type of bar, and converts all existing bars to "OwnerDraw" type

The following VB6 sample paints over the default-visual appearance of the bar:

```
Private Sub G2antt1_AfterDrawPart(ByVal Part As EXG2ANTTLibCtl.DrawPartEnum, ByVal
hdc As Long, ByVal x As Long, ByVal y As Long, ByVal Width As Long, ByVal Height As
Long)
   If (Part = exOwnerDrawBar) Then
      Dim nBkMode, nTextAlign, nColor, hBrush As Long
      Height = Height - 1
      hBrush = SelectObject(hdc, GetStockObject(4))
      Ellipse hdc, x + (Width - Height) / 2, y, x + (Width - Height) / 2 + Height, y + Height
      SelectObject hdc, hBrush
      nBkMode = SetBkMode(hdc, 1)
      nTextAlign = SetTextAlign(hdc, 6)
      nColor = SetTextColor(hdc, RGB(255, 255, 255))
      Dim s As String
      s = G2antt1.Items.CellCaption(G2antt1.DrawPartItem, 1)
      TextOut hdc, x + (Width) / 2 - 1, y + Height / 8, s, Len(s)
      SetTextAlign hdc, nTextAlign
      SetTextColor hdc, nColor
      SetBkMode hdc, nBkMode
   End If
End Sub
```

and previously you have to declare the API functions as:

```
Private Declare Function Ellipse Lib "gdi32" (ByVal hdc As Long, ByVal X1 As Long, ByVal Y1
As Long, ByVal X2 As Long, ByVal Y2 As Long) As Long
```

```
Private Declare Function SetBkMode Lib "gdi32" (ByVal hdc As Long, ByVal nBkMode As Long) As Long
Private Declare Function SetTextAlign Lib "gdi32" (ByVal hdc As Long, ByVal wFlags As Long) As Long
Private Declare Function TextOut Lib "gdi32" Alias "TextOutA" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long, ByVal lpString As String, ByVal nCount As Long) As Long
Private Declare Function SetTextColor Lib "gdi32" (ByVal hdc As Long, ByVal crColor As Long) As Long
Private Declare Function SelectObject Lib "gdi32" (ByVal hdc As Long, ByVal hObject As Long) As Long
Private Declare Function GetStockObject Lib "gdi32" (ByVal nIndex As Long) As Long
```

The following VB/NET sample (/NET Assembly) shows how you can paint over the default-visual appearance of the bar:

```
Private Sub Exg2antt1_AfterDrawPart(ByVal sender As System.Object, ByVal Part As exontrol.EXG2ANTTLib.DrawPartEnum, ByVal G As System.Drawing.Graphics, ByVal Rect As System.Drawing.Rectangle) Handles Exg2antt1.AfterDrawPart
    If (Part = exontrol.EXG2ANTTLib.DrawPartEnum.exOwnerDrawBar) Then
        Dim rText As Rectangle = New Rectangle(Rect.Left + (Rect.Width - Rect.Height) / 2, Rect.Top, Rect.Height, Rect.Height)
        G.FillEllipse(Brushes.Black, rText)
        If Not (Exg2antt1.DrawPartItem = 0) Then
            Dim s As String = Exg2antt1.Items.get_CellCaption(Exg2antt1.DrawPartItem, 1)
            G.DrawString(s, sfFont, Brushes.White, rText, sfCenter)
        End If
    End If
End Sub
```

The following C# sample (/NET Assembly) shows how you can paint over the default-visual appearance of the bar:

```
private void Exg2antt1_AfterDrawPart(object sender, exontrol.EXG2ANTTLib.DrawPartEnum Part, Graphics G, Rectangle Rect)
{
    if ( Part == exontrol.EXG2ANTTLib.DrawPartEnum.exOwnerDrawBar )
    {
        Rectangle rText = new Rectangle(Rect.Left + (Rect.Width - Rect.Height) / 2, Rect.Top,
```

```
Rect.Height, Rect.Height);
      G.FillEllipse(Brushes.Black, rText);
      if (Exg2antt1.DrawPartItem != 0)
      {
         String s = Exg2antt1.Items.get_CellCaption(Exg2antt1.DrawPartItem, 1);
         G.DrawString(s, sfFont, Brushes.White, rText, sfCenter);
      }
   }
}
```

The following C++ sample shows how you can paint over the default-visual appearance of the bar:

```
void CSchedulingDlg::AfterDrawPartG2antt1(long Part, long hDC, long X, long Y, long
Width, long Height)
{
   if ( m_spG2antt != NULL )
      if ( Part == 0 )
      {
         HDC hDCDraw = (HDC)hDC;
         Height = Height - 1;
         HBRUSH hBrush = (HBRUSH)::SelectObject( hDCDraw, GetStockObject( 4
/*BLACK_BRUSH*/ ) );
         Ellipse( hDCDraw, X + (Width - Height)/2, Y, X + (Width - Height)/2 + Height, Y +
Height );
         ::SelectObject( hDCDraw, hBrush );
         int nBkMode = SetBkMode( hDCDraw, 1 /*TRANSPARENT*/ );
         int nColor = SetTextColor( hDCDraw, RGB(255,255,255) );
         int nTextAlign = SetTextAlign( hDCDraw, 6 /*TA_CENTER*/ );
         CString s = m_spG2antt->Items->CellCaption[m_spG2antt->DrawPartItem, 1];
         TextOut( hDCDraw, X + (Width)/2 - 1, Y + Height/8, s, s.GetLength() );
         SetTextAlign( hDCDraw, nTextAlign );
         SetTextColor( hDCDraw, nColor );
         SetBkMode( hDCDraw, nBkMode );
      }
}
```

event AfterExpandItem (Item as HITEM)

Fired after an item is expanded (collapsed).

Type	Description
Item as HITEM	A long expression that indicates the item's handle that indicates the item expanded or collapsed.

The AfterExapndItem event notifies your application that an item is collapsed or expanded. Use the ExpandItem method to programmatically expand or collapse an item. The ExpandItem property also specifies whether an item is expand or collapsed. The ItemChild property retrieves the first child item. Use the BeforeExpandItem event to cancel expanding or collapsing items.

Syntax for AfterExpandItem event, **/NET** version, on:

C#
```
private void AfterExpandItem(object sender,int Item)
{
}
```

VB
```
Private Sub AfterExpandItem(ByVal sender As System.Object,ByVal Item As
Integer) Handles AfterExpandItem
End Sub
```

Syntax for AfterExpandItem event, **/COM** version, on:

C#
```
private void AfterExpandItem(object sender,
AxEXG2ANTTLib._IG2anttEvents_AfterExpandItemEvent e)
{
}
```

C++
```
void OnAfterExpandItem(long Item)
{
}
```

C++ Builder
```
void __fastcall AfterExpandItem(TObject *Sender,Exg2anttlib_tlb::HITEM Item)
{
}
```

Delphi
```
procedure AfterExpandItem(ASender: TObject; Item : HITEM);
```

```
begin
end;
```

Delphi 8 (.NET only)

```
procedure AfterExpandItem(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_AfterExpandItemEvent);
begin
end;
```

Powe…

```
begin event AfterExpandItem(long Item)
end event AfterExpandItem
```

VB.NET

```
Private Sub AfterExpandItem(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_AfterExpandItemEvent) Handles AfterExpandItem
End Sub
```

VB6

```
Private Sub AfterExpandItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
End Sub
```

VBA

```
Private Sub AfterExpandItem(ByVal Item As Long)
End Sub
```

VFP

```
LPARAMETERS Item

```

Xbas…

```
PROCEDURE OnAfterExpandItem(oG2antt,Item)
RETURN
```

Syntax for AfterExpandItem event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="AfterExpandItem(Item)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function AfterExpandItem(Item)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComAfterExpandItem HITEM llItem
    Forward Send OnComAfterExpandItem llItem
End_Procedure
```

Visual Objects

```
METHOD OCX_AfterExpandItem(Item) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_AfterExpandItem(int _Item)
{
}
```

XBasic

```
function AfterExpandItem as v (Item as OLE::Exontrol.G2antt.1::HITEM)
end function
```

dBASE

```
function nativeObject_AfterExpandItem(Item)
return
```

The following VB sample prints the item's state when it is expanded or collapsed:

```
Private Sub G2antt1_AfterExpandItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
    Debug.Print "The " & Item & " item was " & Iif(G2antt1.Items.ExpandItem(Item),
"expanded", "collapsed")
End Sub
```

The following C# sample prints the item's state when it is expanded or collapsed:

```
private void axG2antt1_AfterExpandItem(object sender,
AxEXG2ANTTLib._IG2anttEvents_AfterExpandItemEvent e)
{
    System.Diagnostics.Debug.WriteLine( axG2antt1.Items.get_ExpandItem( e.item) ?
"expanded" : "collapsed" );
}
```

The following VB.NET sample prints the item's state when it is expanded or collapsed:

```
Private Sub AxG2antt1_AfterExpandItem(ByVal sender As Object, ByVal e As
```

```
AxEXG2ANTTLib._IG2anttEvents_AfterExpandItemEvent) Handles
AxG2antt1.AfterExpandItem
    Debug.WriteLine(IIf(AxG2antt1.Items.ExpandItem(e.item), "expanded", "collapsed"))
End Sub
```

The following C++ sample prints the item's state when it is expanded or collapsed:

```
void OnAfterExpandItemG2antt1(long Item)
{
    CItems items = m_g2antt.GetItems();
    CString strFormat;
    strFormat.Format( "%s", items.GetExpandItem( Item ) ? "expanded" : "collapsed" );
    OutputDebugString( strFormat );
}
```

The following VFP sample sample prints the item's state when it is expanded or collapsed:

```
*** ActiveX Control Event ***
LPARAMETERS item

with thisform.G2antt1.Items
    if ( .ExpandItem(item) )
        wait window "expanded" nowait
    else
        wait window "collapsed" nowait
    endif
endwith
```

event AllowAutoDrag (Item as HITEM, NewParent as HITEM, InsertA as HITEM, InsertB as HITEM, Cancel as Boolean)

Occurs when the user drags the item between InsertA and InsertB as child of NewParent.

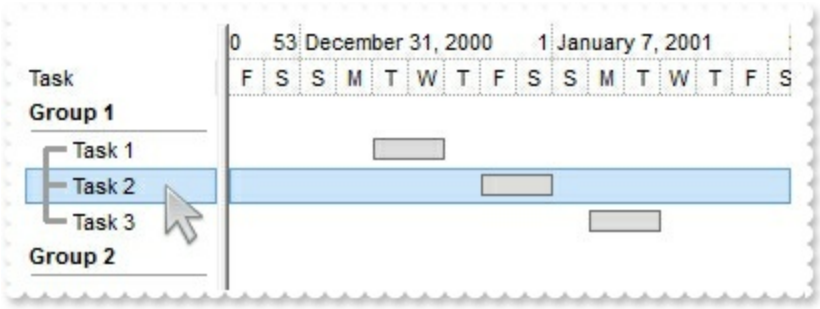
Type	Description
Item as HITEM	A long expression that specifies the handle of the item being dragged.
NewParent as HITEM	A long expression that specifies the handle of the newly parent, to insert the dragging Item. If 0, it indicates root items. The ItemParent property indicates the currently parent of the item.
InsertA as HITEM	A long expression that specifies the handle of the item to insert the dragging Item after. If 0, it indicates that no item after.
InsertB as HITEM	A long expression that specifies the handle of the item to insert the dragging Item before. If 0, it indicates that no item before.
Cancel as Boolean	A Boolean expression that specifies whether the operation can continue (this parameter is by reference)

The AllowAutoDrag event occurs when the user drags the item between InsertA and InsertB as child of NewParent, using the AutoDrag property. The AutoDrag feature indicates what the control does when the user clicks an item and starts dragging it. For instance, using the AutoDrag feature you can let the user arrange the items in the control, or can drop the selection to a any OLE compliant applications like Microsoft Word, Excel and so on... The AllowAutoDrag event may fire when the AutoDrag property is any of the following values:

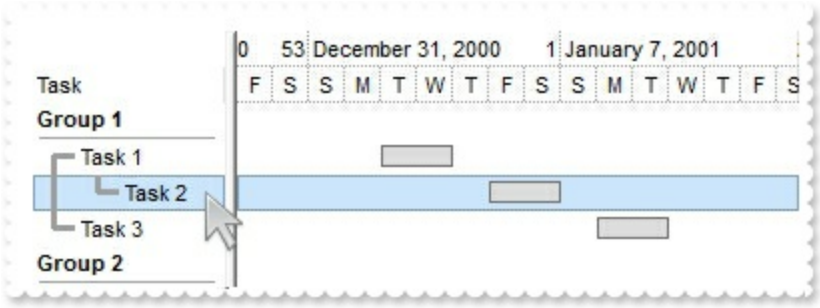
- exAutoDragPosition... (the item can be dragged from a position to another, but not outside of its group)
- exAutoDragPositionKeepIndent... (the item can be dragged to any position or to any parent, while the dragging object keeps its indentation)
- exAutoDragPositionAny... (the item can be dragged to any position or to any parent, with no restriction)

You can use the AllowAutoDrag event to cancel or continue drag and drop operation using the AutoDrag property.

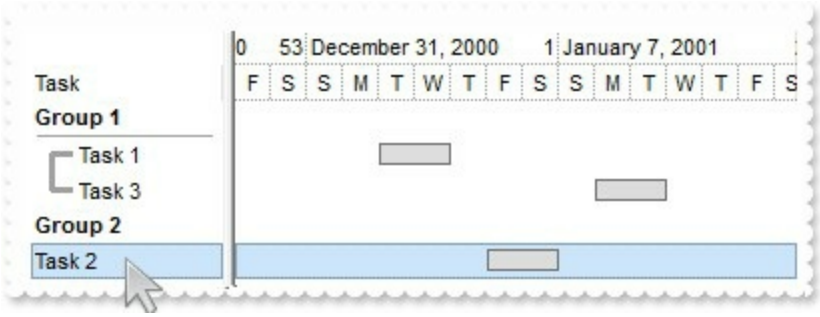
The following screen shot shows the NewParent, InsertA and InsertB parameters, when "Task 2" is dragging to a new position:



- NewParent is "Group 1"
- InsertA is "Task 1"
- InsertB is "Task 3"



- NewParent is "Task 1"
- InsertA is 0
- InsertB is 0



- NewParent is 0
- InsertA is "Group 2"
- InsertB is 0

Syntax for AllowAutoDrag event, **/NET** version, on:

C#

```
private void AllowAutoDrag(object sender,int  Item,int  NewParent,int
InsertA,int  InsertB,ref bool  Cancel)
{
}
```

VB

```
Private Sub AllowAutoDrag(ByVal sender As System.Object,ByVal Item As
Integer,ByVal NewParent As Integer,ByVal InsertA As Integer,ByVal InsertB As
```

```
Integer,ByRef Cancel As Boolean) Handles AllowAutoDrag
End Sub
```

Syntax for AllowAutoDrag event, **/COM** version, on:

C#

```
private void AllowAutoDrag(object sender,
AxEXG2ANTTLib._IG2anttEvents_AllowAutoDragEvent e)
{
}
```

C++

```
void OnAllowAutoDrag(long  Item,long  NewParent,long  InsertA,long
InsertB,BOOL FAR*  Cancel)
{
}
```

C++ Builder

```
void __fastcall AllowAutoDrag(TObject *Sender,Exg2anttlib_tlb::HITEM
Item,Exg2anttlib_tlb::HITEM  NewParent,Exg2anttlib_tlb::HITEM
InsertA,Exg2anttlib_tlb::HITEM  InsertB,VARIANT_BOOL *  Cancel)
{
}
```

Delphi

```
procedure AllowAutoDrag(ASender: TObject; Item : HITEM;NewParent :
HITEM;InsertA : HITEM;InsertB : HITEM;var Cancel : WordBool);
begin
end;
```

Delphi 8 (.NET only)

```
procedure AllowAutoDrag(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_AllowAutoDragEvent);
begin
end;
```

Powe…

```
begin event AllowAutoDrag(long  Item,long  NewParent,long  InsertA,long
InsertB,boolean  Cancel)

end event AllowAutoDrag
```

VB.NET

```
Private Sub AllowAutoDrag(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_AllowAutoDragEvent) Handles AllowAutoDrag
```

```
End Sub
```

VB6

```
Private Sub AllowAutoDrag(ByVal Item As EXG2ANTTLibCtl.HITEM,ByVal
NewParent As EXG2ANTTLibCtl.HITEM,ByVal InsertA As
EXG2ANTTLibCtl.HITEM,ByVal InsertB As EXG2ANTTLibCtl.HITEM,Cancel As
Boolean)
End Sub
```

VBA

```
Private Sub AllowAutoDrag(ByVal Item As Long,ByVal NewParent As Long,ByVal
InsertA As Long,ByVal InsertB As Long,Cancel As Boolean)
End Sub
```

VFP

```
LPARAMETERS Item,NewParent,InsertA,InsertB,Cancel
```

Xbas…

```
PROCEDURE OnAllowAutoDrag(oG2antt,Item,NewParent,InsertA,InsertB,Cancel)

RETURN
```

Syntax for AllowAutoDrag event, ***/COM*** version (others), on:

Java…

```
<SCRIPT EVENT="AllowAutoDrag(Item,NewParent,InsertA,InsertB,Cancel)"
LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function AllowAutoDrag(Item,NewParent,InsertA,InsertB,Cancel)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComAllowAutoDrag HITEM  llItem HITEM  llNewParent HITEM
llInsertA HITEM  llInsertB Boolean  llCancel
   Forward Send OnComAllowAutoDrag llItem llNewParent llInsertA llInsertB
llCancel
End_Procedure
```

Visual Objects

```
METHOD OCX_AllowAutoDrag(Item,NewParent,InsertA,InsertB,Cancel) CLASS
MainDialog
```

```
RETURN NIL
```

X++

```
void onEvent_AllowAutoDrag(int _Item,int _NewParent,int _InsertA,int
_InsertB,COMVariant /*bool*/ _Cancel)
{
}
```

XBasic

```
function AllowAutoDrag as v (Item as OLE::Exontrol.G2antt.1::HITEM,NewParent
as OLE::Exontrol.G2antt.1::HITEM,InsertA as
OLE::Exontrol.G2antt.1::HITEM,InsertB as OLE::Exontrol.G2antt.1::HITEM,Cancel as
L)
end function
```

dBASE

```
function nativeObject_AllowAutoDrag(Item,NewParent,InsertA,InsertB,Cancel)
return
```

The AllowDragDrop event triggers contiguously while the user drags / hovers the focus/selection of items over the control. The GetAsyncKeyState API method can be used to detect whether the mouse button has been released, and so the drop action occurs.

The following VB sample displays "Drag" while user dragging the items, and displays "Drop", when drop operation starts.

```
Private Sub G2antt1_AllowAutoDrag(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal
NewParent As EXG2ANTTLibCtl.HITEM, ByVal InsertA As EXG2ANTTLibCtl.HITEM, ByVal
InsertB As EXG2ANTTLibCtl.HITEM, Cancel As Boolean)
    With G2antt1
        Debug.Print "Drag"
        If (GetAsyncKeyState(VK_LBUTTON) = 0) Then
            Debug.Print "Drop"
        End If
    End With
End Sub
```

where declarations for GetAsyncKeyState API used is:

```
Private Const VK_LBUTTON = &H1
Private Declare Function GetAsyncKeyState Lib "user32" (ByVal vKey As Long) As Integer
```

Once you run the code, you will notice that the AllowAutoDrag event "Drop" may be fired multiple times, so we suggest to postpone any of your actions (like displaying a message box), by posting a window message or use a timer event, to let the control handles / completes the event as in the following sample:

```
Private Sub G2antt1_AllowAutoDrag(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal
NewParent As EXG2ANTTLibCtl.HITEM, ByVal InsertA As EXG2ANTTLibCtl.HITEM, ByVal
InsertB As EXG2ANTTLibCtl.HITEM, Cancel As Boolean)
    With G2antt1
        Debug.Print "Drag"
        If (GetAsyncKeyState(VK_LBUTTON) = 0) Then
            mctlTimerDrop.Enabled = True
        End If
    End With
End Sub
```

where mctlTimerDrop is defined as follows:

```
Dim WithEvents mctlTimerDrop As VB.Timer

Private Sub mctlTimerDrop_Timer()
    mctlTimerDrop.Enabled = False
    MsgBox "Drop."
End Sub



Private Sub Form_Load()
    Set mctlTimerDrop = Me.Controls.Add("VB.Timer", "DropTimer1")
    With mctlTimerDrop
        .Enabled = False
        .Interval = 100
    End With
End Sub
```

event AllowLink (StartItem as HITEM, StartBarKey as Variant, EndItem as HITEM, EndBarKey as Variant, LinkKey as Variant, Cancel as Boolean)

Notifies at runtime when a link between two bars is possible.

Type	Description
StartItem as HITEM	A Long expression that specifies the handle of the item that hosts the bar where the link starts.
StartBarKey as Variant	A VARIANT expression that specifies the key of the bar where the link starts.
EndItem as HITEM	A Long expression that specifies the handle of the item that hosts the bar where the link ends.
EndBarKey as Variant	A VARIANT expression that specifies the key of the bar where the link ends.
LinkKey as Variant	A String expression that specifies the next available key for the link. Use the LinkKey parameter to change the default key for the newly added link at runtime.
Cancel as Boolean	A Boolean expression that specifies whether the operation can continue. By default, the Cancel parameter is False. If The Cancel parameter is True, the specified two bars can not be linked, so the link operation is cancelled.

The AllowLink event occurs when the user links two bars. You can disable or enable linking two bars using the AllowLink event. For instance, you can call Cancel parameter on True, anytime you need to cancel linking two specified bars.

At runtime, you can control linking two bars using one of the followings:

- Handling the AllowLink event, and change the Cancel parameter whenever two bars can't be linked.
- Use the ItemBar(exBarCanBeLinked) property to specify whether a bar can participate into a link.
- Use the ItemBar(exBarCanStartLink) property to specify whether a link can start from specified bar.
- Use the ItemBar(exBarCanEndLink) property to specify whether a link can end to the specified bar.

Syntax for AllowLink event, **/NET** version, on:

C#

```
private void AllowLink(object sender,int StartItem,object StartBarKey,int
EndItem,object EndBarKey,ref object LinkKey,ref bool Cancel)
{
```

```
}
```

VB

```
Private Sub AllowLink(ByVal sender As System.Object,ByVal StartItem As
Integer,ByVal StartBarKey As Object,ByVal EndItem As Integer,ByVal EndBarKey As
Object,ByRef LinkKey As Object,ByRef Cancel As Boolean) Handles AllowLink
End Sub
```

Syntax for AllowLink event, **/COM** version, on:

C#

```
private void AllowLink(object sender,
AxEXG2ANTTLib._IG2anttEvents_AllowLinkEvent e)
{
}
```

C++

```
void OnAllowLink(long StartItem,VARIANT StartBarKey,long EndItem,VARIANT
EndBarKey,VARIANT FAR* LinkKey,BOOL FAR* Cancel)
{
}
```

C++ Builder

```
void __fastcall AllowLink(TObject *Sender,Exg2anttlib_tlb::HITEM StartItem,Variant
StartBarKey,Exg2anttlib_tlb::HITEM EndItem,Variant EndBarKey,Variant *
LinkKey,VARIANT_BOOL * Cancel)
{
}
```

Delphi

```
procedure AllowLink(ASender: TObject; StartItem : HITEM;StartBarKey :
OleVariant;EndItem : HITEM;EndBarKey : OleVariant;var LinkKey : OleVariant;var
Cancel : WordBool);
begin
end;
```

Delphi 8 (.NET only)

```
procedure AllowLink(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_AllowLinkEvent);
begin
end;
```

Powe…

```
begin event AllowLink(long StartItem,any StartBarKey,long EndItem,any
EndBarKey,any LinkKey,boolean Cancel)
```

```
end event AllowLink
```

VB.NET

```
Private Sub AllowLink(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_AllowLinkEvent) Handles AllowLink
End Sub
```

VB6

```
Private Sub AllowLink(ByVal StartItem As EXG2ANTTLibCtl.HITEM,ByVal
StartBarKey As Variant,ByVal EndItem As EXG2ANTTLibCtl.HITEM,ByVal EndBarKey
As Variant,LinkKey As Variant,Cancel As Boolean)
End Sub
```

VBA

```
Private Sub AllowLink(ByVal StartItem As Long,ByVal StartBarKey As Variant,ByVal
EndItem As Long,ByVal EndBarKey As Variant,LinkKey As Variant,Cancel As
Boolean)
End Sub
```

VFP

```
LPARAMETERS StartItem,StartBarKey,EndItem,EndBarKey,LinkKey,Cancel
```

Xbas…

```
PROCEDURE
OnAllowLink(oG2antt,StartItem,StartBarKey,EndItem,EndBarKey,LinkKey,Cancel)
RETURN
```

Syntax for AllowLink event, ***/COM*** version (others), on:

Java…

```
<SCRIPT
EVENT="AllowLink(StartItem,StartBarKey,EndItem,EndBarKey,LinkKey,Cancel)"
LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function AllowLink(StartItem,StartBarKey,EndItem,EndBarKey,LinkKey,Cancel)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComAllowLink HITEM llStartItem Variant llStartBarKey HITEM
llEndItem Variant llEndBarKey Variant llLinkKey Boolean llCancel
    Forward Send OnComAllowLink llStartItem llStartBarKey llEndItem llEndBarKey
```

```
llLinkKey llCancel
End_Procedure
```

Visual Objects

```
METHOD
OCX_AllowLink(StartItem,StartBarKey,EndItem,EndBarKey,LinkKey,Cancel) CLASS
MainDialog
RETURN NIL
```

X++

```
void onEvent_AllowLink(int _StartItem,COMVariant _StartBarKey,int
_EndItem,COMVariant _EndBarKey,COMVariant /*variant*/ _LinkKey,COMVariant
/*bool*/ _Cancel)
{
}
```

XBasic

```
function AllowLink as v (StartItem as OLE::Exontrol.G2antt.1::HITEM,StartBarKey as
A,EndItem as OLE::Exontrol.G2antt.1::HITEM,EndBarKey as A,LinkKey as A,Cancel as
L)
end function
```

dBASE

```
function
nativeObject_AllowLink(StartItem,StartBarKey,EndItem,EndBarKey,LinkKey,Cancel)
return
```

The following VB sample disable linking bars to any "Summary" bars:

```
Private Sub G2antt1_AllowLink(ByVal StartItem As EXG2ANTTLibCtl.HITEM, ByVal
StartBarKey As Variant, ByVal EndItem As EXG2ANTTLibCtl.HITEM, ByVal EndBarKey As
Variant, LinkKey As Variant, Cancel As Boolean)
    With G2antt1.Items
        If (.ItemBar(StartItem, StartBarKey, exBarName) = "Summary") Then
            Cancel = True
        Else
            If (.ItemBar(EndItem, EndBarKey, exBarName) = "Summary") Then
                Cancel = True
            End If
        End If
    End With
```

```
End Sub
```

Use the AddLink method to create a link between two bars. Use the Link property to access properties of a specified link. The Link(exLinksCount) property retrieves the number of links within the chart.

event AnchorClick (AnchorID as String, Options as String)

Occurs when an anchor element is clicked.

Type	Description
AnchorID as String	A string expression that indicates the identifier of the anchor.
Options as String	A string expression that specifies options of the anchor element.

The control fires the AnchorClick event to notify that the user clicks an anchor element. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link. The **<a>** element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The AnchorClick event is fired only if prior clicking the control it shows the hand cursor. For instance, if the cell is disabled, the hand cursor is not shown when hovers the anchor element, and so the AnchorClick event is not fired. Use the FormatAnchor property to specify the visual effect for anchor elements. For instance, if the user clicks the anchor *<a1>anchor*, the control fires the AnchorClick event, where the AnchorID parameter is 1, and the Options parameter is empty. Also, if the user clicks the anchor *<a 1;yourextradata>anchor*, the AnchorID parameter of the AnchorClick event is 1, and the Options parameter is "yourextradata".

Syntax for AnchorClick event, **/NET** version, on:

C#
```
private void AnchorClick(object sender,string AnchorID,string Options)
{
}
```

VB
```
Private Sub AnchorClick(ByVal sender As System.Object,ByVal AnchorID As
String,ByVal Options As String) Handles AnchorClick
End Sub
```

Syntax for AnchorClick event, **/COM** version, on:

C#
```
private void AnchorClick(object sender,
AxEXG2ANTTLib._IG2anttEvents_AnchorClickEvent e)
{
}
```

C++
```
void OnAnchorClick(LPCTSTR AnchorID,LPCTSTR Options)
```

```
{
}
```

C++ Builder

```
void __fastcall AnchorClick(TObject *Sender,BSTR AnchorID,BSTR Options)
{
}
```

Delphi

```
procedure AnchorClick(ASender: TObject; AnchorID : WideString;Options :
WideString);
begin
end;
```

Delphi 8 (.NET only)

```
procedure AnchorClick(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_AnchorClickEvent);
begin
end;
```

Powe…

```
begin event AnchorClick(string AnchorID,string Options)
end event AnchorClick
```

VB.NET

```
Private Sub AnchorClick(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_AnchorClickEvent) Handles AnchorClick
End Sub
```

VB6

```
Private Sub AnchorClick(ByVal AnchorID As String,ByVal Options As String)
End Sub
```

VBA

```
Private Sub AnchorClick(ByVal AnchorID As String,ByVal Options As String)
End Sub
```

VFP

```
LPARAMETERS AnchorID,Options

```

Xbas…

```
PROCEDURE OnAnchorClick(oG2antt,AnchorID,Options)
RETURN
```

Syntax for AnchorClick event, ***/COM*** version (others), on:

Java…

```
<SCRIPT EVENT="AnchorClick(AnchorID,Options)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function AnchorClick(AnchorID,Options)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComAnchorClick String llAnchorID String llOptions
    Forward Send OnComAnchorClick llAnchorID llOptions
End_Procedure
```

Visual Objects

```
METHOD OCX_AnchorClick(AnchorID,Options) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_AnchorClick(str _AnchorID,str _Options)
{
}
```

XBasic

```
function AnchorClick as v (AnchorID as C,Options as C)
end function
```

dBASE

```
function nativeObject_AnchorClick(AnchorID,Options)
return
```

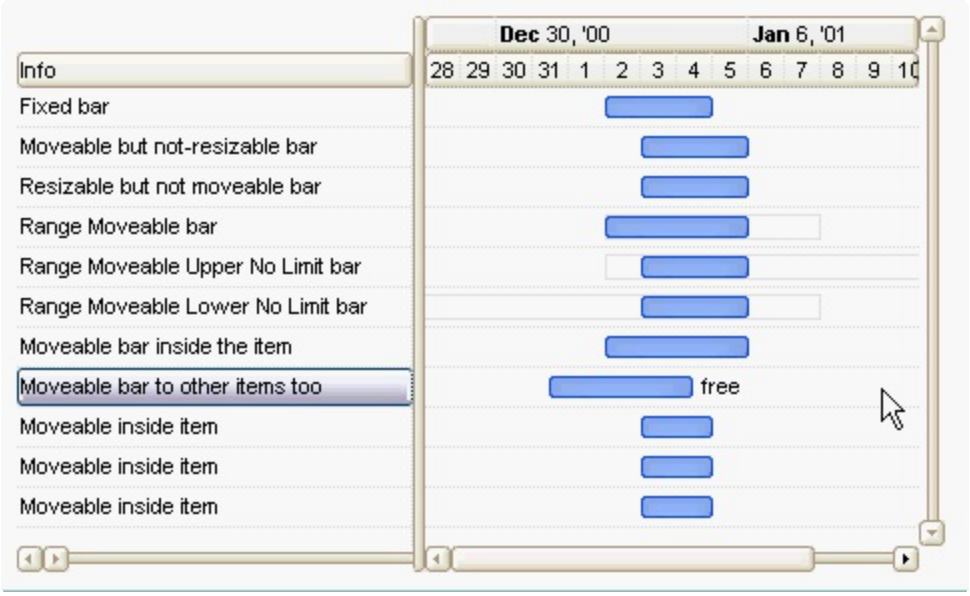
event BarParentChange (Item as HITEM, Key as Variant, NewItem as HITEM, Cancel as Boolean)

Occurs just before moving a bar from current item to another item.

Type	Description
Item as HITEM	A long expression that specifies the owner item (the handle) that displays the bar being moved.
Key as Variant	A string/variant expression that specifies the key of the bar being moved.
NewItem as HITEM	A long expression that specifies the handle of the item where the bar is going to be moved.
Cancel as Boolean	A boolean expression that indicates where the bar can be moved from Item to NewItem. By default, the Cancel parameter is False, so the bar can be moved to any item that does not contain another bar with the same key. *Handle the event, and change the Cancel parameter to False, if you need to prevent moving a bar to NewItem item.*

The BarParentChange event notifies your application when a bar is about to be moved from an item to another item. Use the BarParentChange event to control the items where the bar can be moved through the items. The ItemBar(exBarCanMoveToAnother) property specifies whether the user can drag and drop a bar from an item to another item. The ItemBar(exBarCanMove) property specifies whether the user can moves the bar to a new position inside the bar. The ItemBar(exBarParent) property specifies the handle of the item that displays the bar. Use the ItemBar(exBarParent) property to change programmatically the parent of the specified bar. The control fires the BarResize event when the bar is resized or moved to a new position inside the item. The BarParentChange event is fired during the drag and drop operation as soon as the user moves the bar's parent, but it is fired also one more time once the user releases the left mouse button. You can use the GetAsyncKeyState API function to determine whether the left mouse button is pressed or released as in the following VB sample. The ChartEndChanging(exBarMoveBar) event occurs once the user ends the UI operation like moving a bar.

The following screen shot shows few options to move, limit or resize bars:



The screen show was generate using the following x-script template (the visual appearance code was excluded):

```
BeginUpdate

ScrollBySingleLine = True
DrawGridLines = -1
DefaultItemHeight = 19
GridLineColor = RGB(220, 220, 220)
Chart
{
   FirstVisibleDate = #1/1/2001#
   ScrollRange(0) = #12/28/2000#
   ScrollRange(1) = #1/12/2001#
   DrawDateTicker = False
   NonworkingDays = 0
   DrawGridLines = -1
   ResizeUnitScale = 65536 ' exHour
   AllowCreateBar = False
   PaneWidth(0) = 128
   LevelCount = 2
   Level(0).DrawGridLines = False
   AllowLinkBars = False
   Bars("Task").OverlaidType = 515' exOverlaidBarsStack +
exOverlaidBarsStackAutoArrange
}
```

```
Columns.Add("Info")
Items
{
   Dim h
   h = AddItem("Fixed bar")
   AddBar( h, "Task", #1/2/2001#, #1/5/2001#, "F" )
   ItemBar(h,"F", 10) = False' exBarCanResize
   ItemBar(h,"F", 11) = False' exBarCanMove
   ItemBar(h,"F",6) = "This bar is fixed, so the uer can move or resize it"   'exBarToolTip
   h = AddItem("Moveable but not-resizable bar")
   AddBar( h, "Task", #1/3/2001#, #1/6/2001#, "F" )
   ItemBar(h,"F",6) = "This bar is moveable inside the item, but the user can't resize it."
'exBarToolTip
   ItemBar(h,"F", 10) = False' exBarCanResize
   h = AddItem("Resizable but not moveable bar")
   AddBar( h, "Task", #1/3/2001#, #1/6/2001#, "F" )
   ItemBar(h,"F",6) = "This bar is resizable but the user can't move it."  'exBarToolTip
   ItemBar(h,"F", 11) = False' exBarCanMove
   h = AddItem("Range Moveable bar")
   AddBar( h, "Task", #1/2/2001#, #1/6/2001#, "F" )
   ItemBar(h,"F",6) = "This bar can be moved inside the displayed range."   'exBarToolTip
   ItemBar(h,"F",22) = #1/2/2001# ' exBarMinStart
   ItemBar(h,"F",25) = #1/8/2001# ' exBarMaxEnd
   ItemBar(h,"F",26) = 32     ' exBarShowRange
   ItemBar(h,"F",27) = 90     ' exBarShowRangeTransparent
   h = AddItem("Range Moveable Upper No Limit bar")
   AddBar( h, "Task", #1/3/2001#, #1/6/2001#, "F" )
   ItemBar(h,"F",6) = "This bar can be moved inside the displayed range."   'exBarToolTip
   ItemBar(h,"F",22) = #1/2/2001# ' exBarMinStart
   ItemBar(h,"F",26) = 32     ' exBarShowRange
   ItemBar(h,"F",27) = 90     ' exBarShowRangeTransparent
   h = AddItem("Range Moveable Lower No Limit bar")
   AddBar( h, "Task", #1/3/2001#, #1/6/2001#, "F" )
   ItemBar(h,"F",6) = "This bar can be moved inside the displayed range."   'exBarToolTip
   ItemBar(h,"F",25) = #1/8/2001# ' exBarMaxEnd
   ItemBar(h,"F",26) = 32     ' exBarShowRange
   ItemBar(h,"F",27) = 90     ' exBarShowRangeTransparent
```

```
    h = AddItem("Moveable bar inside the item")
    AddBar( h, "Task", #1/2/2001#, #1/6/2001#, "F" )
    ItemBar(h,"F",6) = "This bar can be moved/resized anywhere inside the item."
'exBarToolTip
    h = AddItem("Moveable bar to other items too")
    AddBar( h, "Task", #1/2/2001#, #1/6/2001#, "FA" )
    ItemBar(h,"FA",6) = "This bar can be moved to other items too. Click the bar and move
it to other items too."   'exBarToolTip
    ItemBar(h,"FA",3) = "free"    'exBarCaption
    ItemBar(h,"FA",4) = 18 'exBarHAlignCaption
    ItemBar(h,"FA",28) = True    'exBarCanMoveToAnother
    h = AddItem("Moveable inside item")
    AddBar( h, "Task", #1/3/2001#, #1/5/2001#, "F1" )
    h = AddItem("Moveable inside item")
    AddBar( h, "Task", #1/3/2001#, #1/5/2001#, "F1" )
    h = AddItem("Moveable inside item")
    AddBar( h, "Task", #1/3/2001#, #1/5/2001#, "F1" )
}
EndUpdate()
```

Syntax for BarParentChange event, **/NET** version, on:

C#

```
private void BarParentChange(object sender,int Item,object Key,int NewItem,ref
bool Cancel)
{
}
```

VB

```
Private Sub BarParentChange(ByVal sender As System.Object,ByVal Item As
Integer,ByVal Key As Object,ByVal NewItem As Integer,ByRef Cancel As Boolean)
Handles BarParentChange
End Sub
```

Syntax for BarParentChange event, **/COM** version, on:

C#

```
private void BarParentChange(object sender,
AxEXG2ANTTLib._IG2anttEvents_BarParentChangeEvent e)
{
}
```

C++

```
void OnBarParentChange(long Item,VARIANT Key,long NewItem,BOOL FAR* Cancel)
{
}
```

C++ Builder

```
void __fastcall BarParentChange(TObject *Sender,Exg2anttlib_tlb::HITEM Item,Variant Key,Exg2anttlib_tlb::HITEM NewItem,VARIANT_BOOL * Cancel)
{
}
```

Delphi

```
procedure BarParentChange(ASender: TObject; Item : HITEM;Key : OleVariant;NewItem : HITEM;var Cancel : WordBool);
begin
end;
```

Delphi 8 (.NET only)

```
procedure BarParentChange(sender: System.Object; e: AxEXG2ANTTLib._IG2anttEvents_BarParentChangeEvent);
begin
end;
```

Powe…

```
begin event BarParentChange(long Item,any Key,long NewItem,boolean Cancel)
end event BarParentChange
```

VB.NET

```
Private Sub BarParentChange(ByVal sender As System.Object, ByVal e As AxEXG2ANTTLib._IG2anttEvents_BarParentChangeEvent) Handles BarParentChange
End Sub
```

VB6

```
Private Sub BarParentChange(ByVal Item As EXG2ANTTLibCtl.HITEM,ByVal Key As Variant,ByVal NewItem As EXG2ANTTLibCtl.HITEM,Cancel As Boolean)
End Sub
```

VBA

```
Private Sub BarParentChange(ByVal Item As Long,ByVal Key As Variant,ByVal NewItem As Long,Cancel As Boolean)
End Sub
```

VFP

```
LPARAMETERS Item,Key,NewItem,Cancel
```

Xbas…
```
PROCEDURE OnBarParentChange(oG2antt,Item,Key,NewItem,Cancel)
RETURN
```

Syntax for BarParentChange event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="BarParentChange(Item,Key,NewItem,Cancel)"
LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function BarParentChange(Item,Key,NewItem,Cancel)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComBarParentChange HITEM llItem Variant llKey HITEM llNewItem
Boolean llCancel
    Forward Send OnComBarParentChange llItem llKey llNewItem llCancel
End_Procedure
```

Visual Objects
```
METHOD OCX_BarParentChange(Item,Key,NewItem,Cancel) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_BarParentChange(int _Item,COMVariant _Key,int
_NewItem,COMVariant /*bool*/ _Cancel)
{
}
```

XBasic
```
function BarParentChange as v (Item as OLE::Exontrol.G2antt.1::HITEM,Key as
A,NewItem as OLE::Exontrol.G2antt.1::HITEM,Cancel as L)
end function
```

dBASE
```
function nativeObject_BarParentChange(Item,Key,NewItem,Cancel)
return
```

The following VB/NET sample shows how to simulate a drop event, in other words how you can get notification once the user drops a bar to another parent (the sample displays the caption of the new parent, once the user drops the bar to a new parent):

```
Dim iMoving As Long = 0
Dim bMoving As Object = Nothing

Private Sub Exg2antt1_BarParentChange(ByVal sender As Object, ByVal Item As Integer,
ByVal Key As Object, ByVal NewItem As Integer, ByRef Cancel As Boolean) Handles
Exg2antt1.BarParentChange
   iMoving = NewItem
   bMoving = Key
End Sub

Private Sub Exg2antt1_ChartEndChanging(ByVal sender As System.Object, ByVal
Operation As exontrol.EXG2ANTTLib.BarOperationEnum) Handles
Exg2antt1.ChartEndChanging
   If (Operation = exontrol.EXG2ANTTLib.BarOperationEnum.exMoveBar) Then
      If (iMoving <> 0) Then
         If Not (bMoving Is Nothing) Then

MessageBox.Show(Exg2antt1.Items.get_CellCaption(Exg2antt1.Items.get_BarParent(iMoving
 bMoving), 0)).ToString()
         End If
      End If
      iMoving = 0
      bMoving = Nothing
   End If
End Sub
```

The following VB sample prevents moving the bar to selectable items only (SelectableItem property):

```
Private Sub G2antt1_BarParentChange(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal Key
As Variant, ByVal NewItem As EXG2ANTTLibCtl.HITEM, Cancel As Boolean)
   With G2antt1.Items
      Cancel = Not .SelectableItem(NewItem)
   End With
End Sub
```

The following VB.NET sample prevents moving the bar to selectable items only (SelectableItem property):

```
Private Sub AxG2antt1_BarParentChange(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_BarParentChangeEvent) Handles
AxG2antt1.BarParentChange
   With AxG2antt1.Items
      e.cancel = Not .SelectableItem(e.newItem)
   End With
End Sub
```

The following C# sample prevents moving the bar to selectable items only (SelectableItem property):

```
private void axG2antt1_BarParentChange(object sender,
AxEXG2ANTTLib._IG2anttEvents_BarParentChangeEvent e)
{
   e.cancel = !axG2antt1.Items.get_SelectableItem(e.newItem);
}
```

The following C++ sample prevents moving the bar to selectable items only (SelectableItem property):

```
#include "Items.h"
void OnBarParentChangeG2antt1(long Item, const VARIANT FAR& Key, long NewItem,
BOOL FAR* Cancel)
{
   *Cancel = !m_g2antt.GetItems().GetSelectableItem( NewItem );
}
```

The following VFP sample prevents moving the bar to selectable items only (SelectableItem property):

```
*** ActiveX Control Event ***
LPARAMETERS item, key, newitem, cancel

cancel = !thisform.G2antt1.Items.SelectableItem(newitem)
```

event BarResize (Item as HITEM, Key as Variant)

Occurs when the bar is moved or resized.

Type	Description
Item as HITEM	A HITEM expression that indicates the handle of the item where the bar is resized.
Key as Variant	A VARIANT expression that indicates the key of the bar being resized. The Key parameter of the AddBar property specifies the key of the bar being added.

The BarResize event notifies your application that the user resizes or moves a bar. The BarResize event is fired when the user changes the percent value. Use the ItemBar property to retrieve the exBarStart and exBarEnd properties of the bar being changed. The exBarPercent value specifies the value of the percent. Use the CellValue property to change the cell's value. The BarParentChange event notifies your application when a bar is about to be moved from an item to another item. You can distingue moving or resizing a specified bar by comparing the ItemBar(exBarDuration) and ItemBar(exBarDurationPrev) values. The BarResizing event notifies the application once the bar is moving or resizing. Use the ItemBar(exBarDuration) and ItemBar(exBarDurationPrev) properties to determine the duration after resizing, and before the bar being resized, so you can determine whether the user resizes or moves a bar. The ChartStartChaning(exMoveBar) event notifies the application once the user starts moving a bar, while the ChartEndChaning(exMoveBar) notifies the application once the user moved the bar. The ChartStartChaning(exResizeStartBar) or ChartStartChaning(exResizeEndBar) event notifies the application once the user starts resizing a bar, while the ChartEndChaning(exResizeStartBar) or ChartEndChaning(exResizeEndBar) notifies the application once the user resized the bar.

Syntax for BarResize event, **/NET** version, on:

C#

```
private void BarResize(object sender,int Item,object Key)
{
}
```

VB

```
Private Sub BarResize(ByVal sender As System.Object,ByVal Item As Integer,ByVal
Key As Object) Handles BarResize
End Sub
```

Syntax for BarResize event, **/COM** version, on:

C#

```
private void BarResize(object sender,
```

```
AxEXG2ANTTLib._IG2anttEvents_BarResizeEvent e)
{
}
```

C++

```
void OnBarResize(long Item,VARIANT Key)
{
}
```

C++ Builder

```
void __fastcall BarResize(TObject *Sender,Exg2anttlib_tlb::HITEM Item,Variant Key)
{
}
```

Delphi

```
procedure BarResize(ASender: TObject; Item : HITEM;Key : OleVariant);
begin
end;
```

Delphi 8 (.NET only)

```
procedure BarResize(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_BarResizeEvent);
begin
end;
```

Powe…

```
begin event BarResize(long Item,any Key)
end event BarResize
```

VB.NET

```
Private Sub BarResize(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_BarResizeEvent) Handles BarResize
End Sub
```

VB6

```
Private Sub BarResize(ByVal Item As EXG2ANTTLibCtl.HITEM,ByVal Key As Variant)
End Sub
```

VBA

```
Private Sub BarResize(ByVal Item As Long,ByVal Key As Variant)
End Sub
```

VFP

```
LPARAMETERS Item,Key

```

Xbas…

```
PROCEDURE OnBarResize(oG2antt,Item,Key)
```

```
RETURN
```

Syntax for BarResize event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="BarResize(Item,Key)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function BarResize(Item,Key)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComBarResize HITEM llItem Variant llKey
   Forward Send OnComBarResize llItem llKey
End_Procedure
```

Visual Objects
```
METHOD OCX_BarResize(Item,Key) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_BarResize(int _Item,COMVariant _Key)
{
}
```

XBasic
```
function BarResize as v (Item as OLE::Exontrol.G2antt.1::HITEM,Key as A)
end function
```

dBASE
```
function nativeObject_BarResize(Item,Key)
return
```

Calling SchedulePDM method invokes the BarResize event for all affected bars. In conclusion, if using the SchedulePDM method during a BarResize event, you can use a counter to prevent calling the SchedulePDM multiple times, like in the following sample VB:

```
Dim iSchedulePDM As Long
Private Sub G2antt1_BarResize(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal Key As
Variant)
   If (iSchedulePDM = 0) Then
      iSchedulePDM = iSchedulePDM + 1
```

```
        G2antt1.Items.SchedulePDM Item, Key
      iSchedulePDM = iSchedulePDM - 1
    End If
End Sub
```

The following VB sample displays the BarMove message once a a bar is moved (not sizing):

```
Dim iMoving As Long

Private Sub Gantt1_BarResize(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal Key As
Variant)
    If Not (iMoving = 0) Then
        Debug.Print "BarMove"
    End If
End Sub

Private Sub Gantt1_ChartStartChanging(ByVal Operation As
EXG2ANTTLibCtl.BarOperationEnum)
    If (Operation = exMoveBar) Then
        iMoving = iMoving + 1
    End If
End Sub

Private Sub Gantt1_ChartEndChanging(ByVal Operation As
EXG2ANTTLibCtl.BarOperationEnum)
    If (Operation = exMoveBar) Then
        iMoving = iMoving - 1
    End If
End Sub
```

The following VB sample displays the operation performed as if it was a moving or a resizing the bar:

```
Private Sub G2antt1_BarResize(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal Key As
Variant)
    With G2antt1.Items
        If .ItemBar(Item, Key, exBarDurationPrev) <> .ItemBar(Item, Key, exBarDuration) Then
```

```
        Debug.Print "The item has been resized."
    Else
        Debug.Print "The item has been moved."
    End If
  End With
End Sub
```

The following VB sample displays the new start and end data for the bar being moved or resized:

```
Private Sub G2antt1_BarResize(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal Key As
Variant)
  With G2antt1.Items
    Debug.Print "NewStart: " & .ItemBar(Item, Key, exBarStart)
    Debug.Print "NewEnd: " & .ItemBar(Item, Key, exBarEnd)
  End With
End Sub
```

The following VB sample changes the background color of the bar being moved or renamed:

```
Private Sub G2antt1_BarResize(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal Key As
Variant)
  G2antt1.BeginUpdate
  With G2antt1.Items
    .ItemBar(Item, Key, exBarBackColor) = RGB(255, 0, 0)
  End With
  G2antt1.EndUpdate
End Sub
```

The following C++ sample displays the new start and end data for the bar being moved or resized:

```
void OnBarResizeG2antt1(long Item, const VARIANT FAR& Key)
{
  CItems items = m_g2antt.GetItems();
  COleVariant vtStartDate = items.GetItemBar( Item, Key, /*exBarStart*/1 );
  COleVariant vtEndDate = items.GetItemBar( Item, Key, /*exBarEnd*/2 );
  OutputDebugString( "newStartDate: " );
```

```
    OutputDebugString( V2S( &vtStartDate ) );
    OutputDebugString( "\n" );
    OutputDebugString( "newEndDate: " );
    OutputDebugString( V2S( &vtEndDate ) );
    OutputDebugString( "\n" );
}
```

The following VB.NET sample displays the new start and end data for the bar being moved or resized:

```
Private Sub AxG2antt1_BarResize(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_BarResizeEvent) Handles AxG2antt1.BarResize
    With AxG2antt1.Items
        System.Diagnostics.Debug.Print("newStartDate: " + .ItemBar(e.item, e.key,
EXG2ANTTLib.ItemBarPropertyEnum.exBarStart))
        System.Diagnostics.Debug.Print("newEndDate: " + .ItemBar(e.item, e.key,
EXG2ANTTLib.ItemBarPropertyEnum.exBarEnd))
    End With
End Sub
```

The following C# sample displays the new start and end data for the bar being moved or resized:

```
private void axG2antt1_BarResize(object sender,
AxEXG2ANTTLib._IG2anttEvents_BarResizeEvent e)
{
    System.Diagnostics.Debug.Print("newStartDate: " + axG2antt1.Items.get_ItemBar(e.item,
e.key, EXG2ANTTLib.ItemBarPropertyEnum.exBarStart).ToString());
    System.Diagnostics.Debug.Print("newStartDate: " + axG2antt1.Items.get_ItemBar(e.item,
e.key, EXG2ANTTLib.ItemBarPropertyEnum.exBarEnd).ToString());
}
```

The following VFP sample displays the new start and end data for the bar being moved or resized:

```
*** ActiveX Control Event ***
LPARAMETERS item, key

with thisform.G2antt1.Items
```

```
    ? .ItemBar(item,key,1)
    ? .ItemBar(item,key,2)
endwith
```

event BarResizing (Item as HITEM, Key as Variant)

Occurs when a bar is moving or resizing.

Type	Description
Item as HITEM	A long expression that specifies the item that hosts the bar being moved or resized.
Key as Variant	A VARIANT expression that specifies the bar being moved or resized.

The BarResizing event is fired continually while the bar is resizing or moving. The BarResize event notifies the application once the bar is moved or resized. The ChartStartChaning(exMoveBar) event notifies the application once the user starts moving a bar, while the ChartEndChaning(exMoveBar) notifies the application once the user moved the bar. The ChartStartChaning(exResizeStartBar) or ChartStartChaning(exResizeEndBar) event notifies the application once the user starts resizing a bar, while the ChartEndChaning(exResizeStartBar) or ChartEndChaning(exResizeEndBar) notifies the application once the user resized the bar.

Use the ItemBar(exBarStart) and ItemBar(exBarEnd)/ItemBar(exBarEndInclusive) properties to determine the start and end point of the bar being moved or resized. Use the ItemBar(exBarDuration) and ItemBar(exBarDurationPrev) properties to determine the duration after resizing, and before the bar being resized, so you can determine whether the user resizes or moves a bar.

Syntax for BarResizing event, **/NET** version, on:

C#
```
private void BarResizing(object sender,int Item,object Key)
{
}
```

VB
```
Private Sub BarResizing(ByVal sender As System.Object,ByVal Item As Integer,ByVal
Key As Object) Handles BarResizing
End Sub
```

Syntax for BarResizing event, **/COM** version, on:

C#
```
private void BarResizing(object sender,
AxEXG2ANTTLib._IG2anttEvents_BarResizingEvent e)
{
}
```

C++

```
void OnBarResizing(long Item,VARIANT Key)
{
}
```

C++ Builder

```
void __fastcall BarResizing(TObject *Sender,Exg2anttlib_tlb::HITEM Item,Variant Key)
{
}
```

Delphi

```
procedure BarResizing(ASender: TObject; Item : HITEM;Key : OleVariant);
begin
end;
```

Delphi 8 (.NET only)

```
procedure BarResizing(sender: System.Object; e: AxEXG2ANTTLib._IG2anttEvents_BarResizingEvent);
begin
end;
```

Powe…

```
begin event BarResizing(long Item,any Key)
end event BarResizing
```

VB.NET

```
Private Sub BarResizing(ByVal sender As System.Object, ByVal e As AxEXG2ANTTLib._IG2anttEvents_BarResizingEvent) Handles BarResizing
End Sub
```

VB6

```
Private Sub BarResizing(ByVal Item As EXG2ANTTLibCtl.HITEM,ByVal Key As Variant)
End Sub
```

VBA

```
Private Sub BarResizing(ByVal Item As Long,ByVal Key As Variant)
End Sub
```

VFP

```
LPARAMETERS Item,Key

```

Xbas…

```
PROCEDURE OnBarResizing(oG2antt,Item,Key)
RETURN
```

Syntax for BarResizing event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="BarResizing(Item,Key)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function BarResizing(Item,Key)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComBarResizing HITEM llItem Variant llKey
    Forward Send OnComBarResizing llItem llKey
End_Procedure
```

Visual Objects
```
METHOD OCX_BarResizing(Item,Key) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_BarResizing(int _Item,COMVariant _Key)
{
}
```

XBasic
```
function BarResizing as v (Item as OLE::Exontrol.G2antt.1::HITEM,Key as A)
end function
```

dBASE
```
function nativeObject_BarResizing(Item,Key)
return
```

The following VB sample moves the bar in a second gantt control once the user resizes or moves a bar in the first gantt control:

```
Private Sub G2antt1_BarResizing(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal Key As
Variant)
    With G2antt2
        .BeginUpdate
        .Items.AddBar .Items.ItemByIndex(G2antt1.Items.ItemToIndex(Item)),
G2antt1.Items.ItemBar(Item, Key, exBarName), G2antt1.Items.ItemBar(Item, Key,
exBarStart), G2antt1.Items.ItemBar(Item, Key, exBarEnd)
```

```
    .EndUpdate
  End With
End Sub
```

The sample uses the AddBar method instead ItemBar, so the start and end points of the bar are updated once.

If using the *SchedulePDM method during a BarResizing* event, you can see the order of the events in the following VB sample:

```
Private Sub G2antt1_BarResize(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal Key As
Variant)
  Debug.Print "BarResize invoked"
End Sub

Private Sub G2antt1_BarResizing(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal Key As
Variant)
  Debug.Print "BarResizing invoked"
  G2antt1.Items.SchedulePDM Item, Key
End Sub

Private Sub G2antt1_ChartStartChanging(ByVal Operation As
EXG2ANTTLibCtl.BarOperationEnum)
  If (Operation = exPDM) Then
    Debug.Print "SchedulePDM starts"
  End If
End Sub

Private Sub G2antt1_ChartEndChanging(ByVal Operation As
EXG2ANTTLibCtl.BarOperationEnum)
  If (Operation = exPDM) Then
    Debug.Print "SchedulePDM ends"
  End If
End Sub
```

The output shows as follows:

```
BarResizing invoked
SchedulePDM starts
```

```
    BarResize invoked
    BarResize invoked
SchedulePDM ends
BarResize invoked
```

event BeforeDrawPart (Part as DrawPartEnum, hDC as Long, X as Long, Y as Long, Width as Long, Height as Long, Cancel as Boolean)

Occurs just before drawing a part of the control.

Type	Description
Part as DrawPartEnum	A Part being painted. If the Part parameter is exOwnerDrawBar, the DrawPartItem property specifies the handle of the item that hosts the "OwnerDraw" bar, while the DrawPartKey property specifies the key of the bar to be painted. Use the Add or Copy method to add an "OwnerDraw" bar
hDC as Long	A long expression that specifies the handle of the device context where you can perform your own draw (available for /COM only). The /NET or /WPF assembly provides a System.Drawing.Graphics object instead hDC parameter
X as Long	(by reference) A long expression that specifies the left coordinate of the rectangle where the paint should occur. *You can change the X parameter during the handler, to define the new left coordinate for the default painting.* The /NET or /WPF assembly provides a System.Drawing.Rectangle instead (X, Y, Width, Height).
Y as Long	(by reference) A long expression that specifies the top coordinate of the rectangle where the paint should occur. *You can change the Y parameter during the handler, to define the new top coordinate for the default painting.* The /NET or /WPF assembly provides a System.Drawing.Rectangle instead (X, Y, Width, Height).
Width as Long	(by reference) A long expression that specifies the width of the rectangle where the paint should occur. *You can change the Width parameter during the handler, to define the new width for the default painting.* The /NET or /WPF assembly provides a System.Drawing.Rectangle instead (X, Y, Width, Height).
Height as Long	(by reference) A long expression that specifies the height of the rectangle where the paint should occur. *You can change the Height parameter during the handler, to define the new width for the default painting.* The /NET or /WPF assembly provides a System.Drawing.Rectangle instead (X, Y, Width, Height).

Cancel as Boolean	(by reference) A Boolean expression that specifies whether the default painting is canceled or not.

The BeforeDrawPart and AfterDrawPart events occur when different parts of the control requires to be drawn. Use the BeforeDrawPart and AfterDrawPart events to add your custom drawing to be shown in the component. Use the BeforeDrawPart event to perform your own drawing before the default drawing, canceling the default drawing, or changing the area being assigned to the part part when painting. Use the AfterDrawPart event to perform your own drawing after default painting occurs. The /NET Assembly provides instead hDC and (X,Y,Width,Height) parameters a Graphics object and a Rectangle object, the last being passed by reference. Use the HistogramBoundsChanged event to notify your application when the left part of the histogram is resized, so inside controls must be re-positioned.

Syntax for BeforeDrawPart event, **/NET** version, on:

C#
```
private void BeforeDrawPart(object sender,exontrol.EXG2ANTTLib.DrawPartEnum Part,int hDC,ref int X,ref int Y,ref int Width,ref int Height,ref bool Cancel)
{
}
```

VB
```
Private Sub BeforeDrawPart(ByVal sender As System.Object,ByVal Part As exontrol.EXG2ANTTLib.DrawPartEnum,ByVal hDC As Integer,ByRef X As Integer,ByRef Y As Integer,ByRef Width As Integer,ByRef Height As Integer,ByRef Cancel As Boolean) Handles BeforeDrawPart
End Sub
```

Syntax for BeforeDrawPart event, **/COM** version, on:

C#
```
private void BeforeDrawPart(object sender, AxEXG2ANTTLib._IG2anttEvents_BeforeDrawPartEvent e)
{
}
```

C++
```
void OnBeforeDrawPart(long Part,long hDC,long FAR* X,long FAR* Y,long FAR* Width,long FAR* Height,BOOL FAR* Cancel)
{
}
```

C++ Builder
```
void __fastcall BeforeDrawPart(TObject *Sender,Exg2anttlib_tlb::DrawPartEnum
```

```
Part,long hDC,long * X,long * Y,long * Width,long * Height,VARIANT_BOOL *
Cancel)
{
}
```

Delphi

```
procedure BeforeDrawPart(ASender: TObject; Part : DrawPartEnum;hDC :
Integer;var X : Integer;var Y : Integer;var Width : Integer;var Height : Integer;var
Cancel : WordBool);
begin
end;
```

Delphi 8 (.NET only)

```
procedure BeforeDrawPart(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_BeforeDrawPartEvent);
begin
end;
```

Powe…

```
begin event BeforeDrawPart(long Part,long hDC,long X,long Y,long Width,long
Height,boolean Cancel)
end event BeforeDrawPart
```

VB.NET

```
Private Sub BeforeDrawPart(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_BeforeDrawPartEvent) Handles BeforeDrawPart
End Sub
```

VB6

```
Private Sub BeforeDrawPart(ByVal Part As EXG2ANTTLibCtl.DrawPartEnum,ByVal
hDC As Long,X As Long,Y As Long,Width As Long,Height As Long,Cancel As
Boolean)
End Sub
```

VBA

```
Private Sub BeforeDrawPart(ByVal Part As Long,ByVal hDC As Long,X As Long,Y As
Long,Width As Long,Height As Long,Cancel As Boolean)
End Sub
```

VFP

```
LPARAMETERS Part,hDC,X,Y,Width,Height,Cancel

```

Xbas…

```
PROCEDURE OnBeforeDrawPart(oG2antt,Part,hDC,X,Y,Width,Height,Cancel)
RETURN
```

Syntax for BeforeDrawPart event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="BeforeDrawPart(Part,hDC,X,Y,Width,Height,Cancel)"
LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function BeforeDrawPart(Part,hDC,X,Y,Width,Height,Cancel)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComBeforeDrawPart OLEDrawPartEnum llPart Integer llhDC Integer
llX Integer llY Integer llWidth Integer llHeight Boolean llCancel
    Forward Send OnComBeforeDrawPart llPart llhDC llX llY llWidth llHeight
llCancel
End_Procedure
```

Visual Objects
```
METHOD OCX_BeforeDrawPart(Part,hDC,X,Y,Width,Height,Cancel) CLASS
MainDialog
RETURN NIL
```

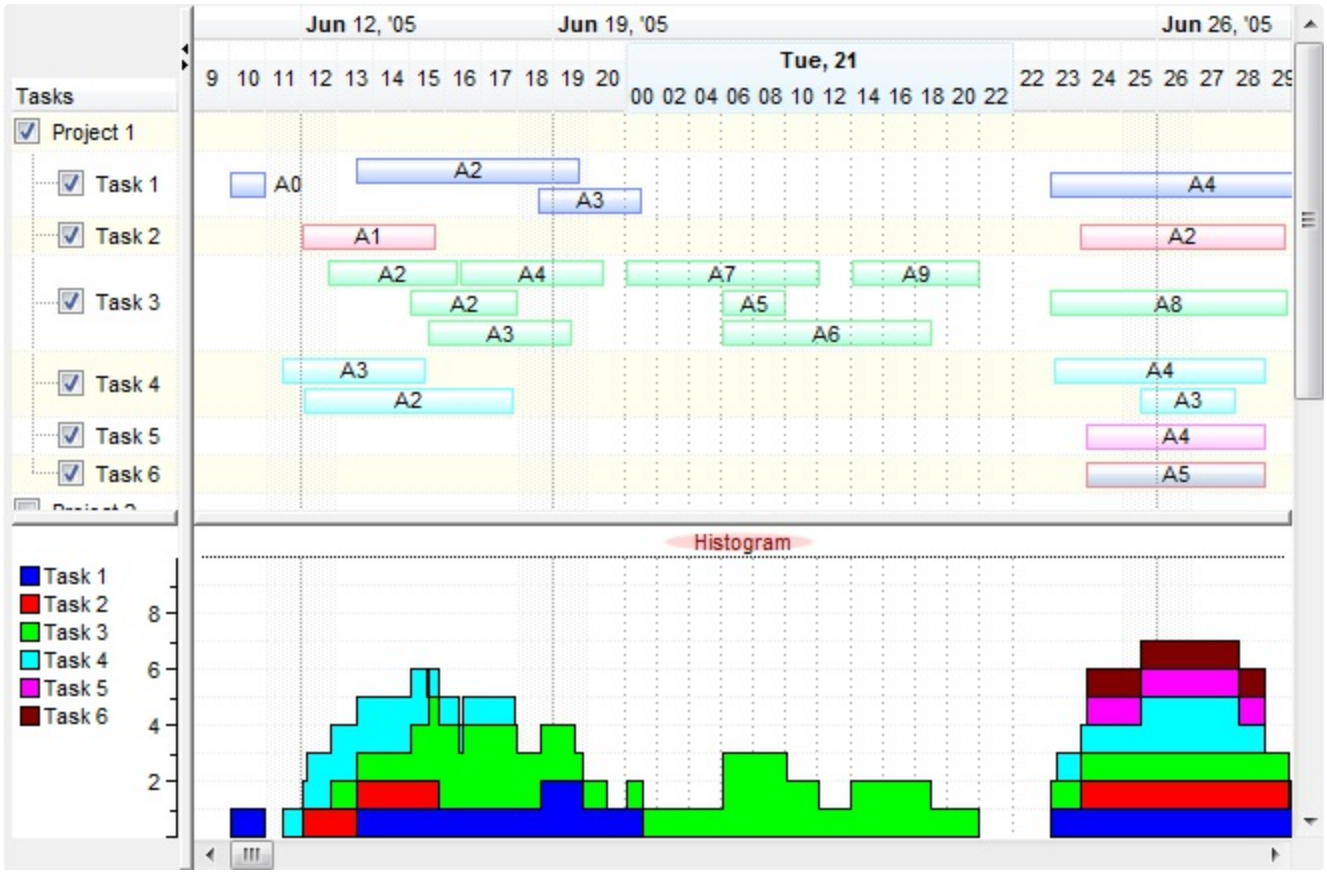
X++
```
void onEvent_BeforeDrawPart(int _Part,int _hDC,COMVariant /*long*/
_X,COMVariant /*long*/ _Y,COMVariant /*long*/ _Width,COMVariant /*long*/
_Height,COMVariant /*bool*/ _Cancel)
{
}
```

XBasic
```
function BeforeDrawPart as v (Part as OLE::Exontrol.G2antt.1::DrawPartEnum,hDC
as N,X as N,Y as N,Width as N,Height as N,Cancel as L)
end function
```

dBASE
```
function nativeObject_BeforeDrawPart(Part,hDC,X,Y,Width,Height,Cancel)
return
```

The following VB sample changes the Y and the Height parameters, and paints the "Histogram" text inside the histogram as shown bellow in the screen shot:

```
Private Type RECT
        Left As Long
        Top As Long
        Right As Long
        Bottom As Long
End Type
Private Const DT_SINGLELINE = &H20
Private Const DT_CENTER = &H1
Private Declare Function DrawText Lib "user32" Alias "DrawTextA" (ByVal hdc As Long,
ByVal lpStr As String, ByVal nCount As Long, lpRect As RECT, ByVal wFormat As Long) As
Long



Private Sub G2antt1_BeforeDrawPart(ByVal Part As EXG2ANTTLibCtl.DrawPartEnum, ByVal
hdc As Long, X As Long, Y As Long, Width As Long, Height As Long, Cancel As Boolean)
    If (Part = exDrawLeftHistogram) Or (Part = exDrawRightHistogram) Then

        Dim h As Long
        h = 16

        If ((Part = exDrawRightHistogram)) Then
            Dim r As RECT
            r.Left = X + 2
            r.Right = r.Left + Width - 4
            r.Top = Y + 1
            r.Bottom = r.Top + h - 2

            DrawText hdc, "Histogram", 9, r, DT_SINGLELINE + DT_CENTER
        End If

        Y = Y + h
        Height = Height - h
    End If
End Sub
```


Jun 12, '05
Jun 19, '05
Jun 26, '05
Tue, 21
Tasks
Project 1
Task 1
Task 2
Task 3
Task 4
Task 5
Task 6
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
Histogram
8
6
4
2

event BeforeExpandItem (Item as HITEM, Cancel as Variant)

Fired before an item is about to be expanded (collapsed).

Type	Description
Item as HITEM	A long expression that indicates the handle of the item being expanded or collapsed.
Cancel as Variant	A boolean expression that indicates whether the control cancel expanding or collapsing the item.

The BeforeExpandItem event notifies your application that an item is about to be collapsed or expanded. Use the BeforeExpandItem event to cancel expanding or collapsing items. Use the BeforeExpandItem event to load new items when filling a virtual tree. The AfterExpandItem event is fired after an item is expanded or collapsed. Use the ExpandItem method to programmatically expand or collapse an item. Use the ExpandOnSearch property to expand items while user types characters to search for items using incremental search feature.

Syntax for BeforeExpandItem event, **/NET** version, on:

C#
```
private void BeforeExpandItem(object sender,int Item,ref object Cancel)
{
}
```

VB
```
Private Sub BeforeExpandItem(ByVal sender As System.Object,ByVal Item As
Integer,ByRef Cancel As Object) Handles BeforeExpandItem
End Sub
```

Syntax for BeforeExpandItem event, **/COM** version, on:

C#
```
private void BeforeExpandItem(object sender,
AxEXG2ANTTLib._IG2anttEvents_BeforeExpandItemEvent e)
{
}
```

C++
```
void OnBeforeExpandItem(long Item,VARIANT FAR* Cancel)
{
}
```

C++ Builder
```
void __fastcall BeforeExpandItem(TObject *Sender,Exg2anttlib_tlb::HITEM
```

```
Item,Variant * Cancel)
{
}
```

Delphi

```
procedure BeforeExpandItem(ASender: TObject; Item : HITEM;var Cancel :
OleVariant);
begin
end;
```

Delphi 8 (.NET only)

```
procedure BeforeExpandItem(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_BeforeExpandItemEvent);
begin
end;
```

Powe…

```
begin event BeforeExpandItem(long Item,any Cancel)
end event BeforeExpandItem
```

VB.NET

```
Private Sub BeforeExpandItem(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_BeforeExpandItemEvent) Handles
BeforeExpandItem
End Sub
```

VB6

```
Private Sub BeforeExpandItem(ByVal Item As EXG2ANTTLibCtl.HITEM,Cancel As
Variant)
End Sub
```

VBA

```
Private Sub BeforeExpandItem(ByVal Item As Long,Cancel As Variant)
End Sub
```

VFP

```
LPARAMETERS Item,Cancel
```

Xbas…

```
PROCEDURE OnBeforeExpandItem(oG2antt,Item,Cancel)
RETURN
```

Syntax for BeforeExpandItem event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="BeforeExpandItem(Item,Cancel)" LANGUAGE="JScript">
```

```
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function BeforeExpandItem(Item,Cancel)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComBeforeExpandItem HITEM llItem Variant llCancel
    Forward Send OnComBeforeExpandItem llItem llCancel
End_Procedure
```

Visual Objects
```
METHOD OCX_BeforeExpandItem(Item,Cancel) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_BeforeExpandItem(int _Item,COMVariant /*variant*/ _Cancel)
{
}
```

XBasic
```
function BeforeExpandItem as v (Item as OLE::Exontrol.G2antt.1::HITEM,Cancel as
A)
end function
```

dBASE
```
function nativeObject_BeforeExpandItem(Item,Cancel)
return
```

The following VB sample cancels expanding or collapsing items:

```
Private Sub G2antt1_BeforeExpandItem(ByVal Item As EXG2ANTTLibCtl.HITEM, Cancel As
Variant)
    Cancel = True
End Sub
```

The following VB sample prints the item's state when it is expanded or collapsed:

```
Private Sub G2antt1_AfterExpandItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
    Debug.Print "The " & Item & " item was " & Iif(G2antt1.Items.ExpandItem(Item),
"expanded", "collapsed")
End Sub
```

The following C# sample cancels expanding or collapsing items:

```
private void axG2antt1_BeforeExpandItem(object sender,
AxEXG2ANTTLib._IG2anttEvents_BeforeExpandItemEvent e)
{
   e.cancel = true;
}
```

The following VB.NET sample cancels expanding or collapsing items:

```
Private Sub AxG2antt1_BeforeExpandItem(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_BeforeExpandItemEvent) Handles
AxG2antt1.BeforeExpandItem
   e.cancel = True
End Sub
```

The following C++ sample cancels expanding or collapsing items:

```
void OnBeforeExpandItemG2antt1(long Item, VARIANT FAR* Cancel)
{
   V_VT( Cancel ) = VT_BOOL;
   V_BOOL( Cancel ) = VARIANT_TRUE;
}
```

The following VFP sample cancels expanding or collapsing items:

```
*** ActiveX Control Event ***
LPARAMETERS item, cancel

cancel = .t.
```

event ButtonClick (Item as HITEM, ColIndex as Long, Key as Variant)

Occurs when user clicks on the cell's button.

Type	Description
Item as HITEM	A long expression that determines the item's handle. If the Item parameter is 0, and the ColIndex property is different than zero, the ColIndex indicates the handle of the cell where the state is changed.
ColIndex as Long	A long expression that indicates the column's index, if the Item parameter is not zero, a long expression that indicates the handle of the cell if the Item parameter is 0.
Key as Variant	Specifies the button's key that's clicked. If the Key parameter is empty, the user clicked the drop down button of the editor.

Use the ButtonClick event to notify your application that a button is clicked. Use the ColumnClick event to notify your application that the user clicks the column's header. Use the CellImageClick event to notify your application that the user clicks an icon in the cell. You can assign a button to a cell using any of the following ways:

- The CellHasButton property specifies whether the cell displays a button. Use the CellValue property indicates the button's caption. In this case the Key parameter is empty.
- The AddButton method adds a button to an editor. The Key parameter indicates the key of the button being clicked. A drop down type editor like ButtonType, DropDownType, DropDownListType, PickEditType, DateType, ColorType, FontType and PictureType includes a drop down button. The Key parameter is empty, for a drop down button.

Syntax for ButtonClick event, **/NET** version, on:

C#
```
private void ButtonClick(object sender,int Item,int ColIndex,object Key)
{
}
```

VB
```
Private Sub ButtonClick(ByVal sender As System.Object,ByVal Item As Integer,ByVal
ColIndex As Integer,ByVal Key As Object) Handles ButtonClick
End Sub
```

Syntax for ButtonClick event, **/COM** version, on:

C#
```
private void ButtonClick(object sender,
AxEXG2ANTTLib._IG2anttEvents_ButtonClickEvent e)
{
}
```

C++
```
void OnButtonClick(long Item,long ColIndex,VARIANT Key)
{
}
```

C++ Builder
```
void __fastcall ButtonClick(TObject *Sender,Exg2anttlib_tlb::HITEM Item,long
ColIndex,Variant Key)
{
}
```

Delphi
```
procedure ButtonClick(ASender: TObject; Item : HITEM;ColIndex : Integer;Key :
OleVariant);
begin
end;
```

Delphi 8 (.NET only)
```
procedure ButtonClick(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_ButtonClickEvent);
begin
end;
```

Powe…
```
begin event ButtonClick(long Item,long ColIndex,any Key)
end event ButtonClick
```

VB.NET
```
Private Sub ButtonClick(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_ButtonClickEvent) Handles ButtonClick
End Sub
```

VB6
```
Private Sub ButtonClick(ByVal Item As EXG2ANTTLibCtl.HITEM,ByVal ColIndex As
Long,ByVal Key As Variant)
End Sub
```

VBA
```
Private Sub ButtonClick(ByVal Item As Long,ByVal ColIndex As Long,ByVal Key As
Variant)
```

```
End Sub
```

VFP

```
LPARAMETERS Item,ColIndex,Key
```

Xbas…

```
PROCEDURE OnButtonClick(oG2antt,Item,ColIndex,Key)
RETURN
```

Syntax for ButtonClick event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="ButtonClick(Item,ColIndex,Key)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function ButtonClick(Item,ColIndex,Key)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComButtonClick HITEM llItem Integer llColIndex Variant llKey
    Forward Send OnComButtonClick llItem llColIndex llKey
End_Procedure
```

Visual Objects

```
METHOD OCX_ButtonClick(Item,ColIndex,Key) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_ButtonClick(int _Item,int _ColIndex,COMVariant _Key)
{
}
```

XBasic

```
function ButtonClick as v (Item as OLE::Exontrol.G2antt.1::HITEM,ColIndex as N,Key as A)
end function
```

dBASE

```
function nativeObject_ButtonClick(Item,ColIndex,Key)
return
```

The following VB sample displays the key of the button being clicked:

```
With G2antt1.Columns.Add("Editor").Editor
   .EditType = EditType
   .AddButton "Key1", 1
   .AddButton "Key2", 2, EXG2ANTTLibCtl.AlignmentEnum.RightAlignment, "This is a bit of text that should be displayed when the cursor is over the button", "Some information"
   .AddButton "Key3", 3, EXG2ANTTLibCtl.AlignmentEnum.RightAlignment
End With
```

```
...
```

```
Private Sub G2antt1_ButtonClick(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal ColIndex As Long, ByVal Key As Variant)
   ' Displays the button's key that was clicked
   Dim mes As String
   mes = "You have pressed the button"
   mes = mes + Iif(Len(Key) = 0, "", " '" & Key & "'")
   mes = mes + " of cell '" & G2antt1.Items.CellValue(Item) & "'."
   Debug.Print mes
End Sub
```

The following VB sample displays the caption of the cell where a button is clicked:

```
Private Sub G2antt1_ButtonClick(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal ColIndex As Long, ByVal Key As Variant)
   With G2antt1.Items
      Debug.Print .CellValue(Item, ColIndex) & ", Key = '" & Key & "'"
   End With
End Sub
```

The following C++ sample displays the caption of the cell where a button is clicked:

```
#include "Items.h"
void OnButtonClickG2antt1(long Item, long ColIndex, const VARIANT FAR& Key)
{
   CItems items = m_g2antt.GetItems();
   CString strFormat;
   strFormat.Format( "%s, Key = '%s'", items.GetCellValue( COleVariant( Item ), COleVariant( ColIndex ) ), V2S( (LPVARIANT)&Key ) );
```

```
    OutputDebugString( strFormat );
}
```

where the V2S string may look like follows:

```
static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
    if ( pv )
    {
        if ( pv->vt == VT_ERROR )
            return szDefault;

        COleVariant vt;
        vt.ChangeType( VT_BSTR, pv );
        return V_BSTR( &vt );
    }
    return szDefault;
}
```

or

```
static string V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
    if ( pv )
    {
        if ( pv->vt == VT_ERROR )
            return szDefault;

        CComVariant vt;
        if ( SUCCEEDED( vt.ChangeType( VT_BSTR, pv ) ) )
        {
            USES_CONVERSION;
            return OLE2T(V_BSTR( &vt ));
        }
    }
    return szDefault;
}
```

if you are using STL.

The following VB.NET sample displays the caption of the cell where a button is clicked:

```
Private Sub AxG2antt1_ButtonClick(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_ButtonClickEvent) Handles AxG2antt1.ButtonClick
   With AxG2antt1.Items
      Dim strKey As String = ""
      If Not (e.key Is Nothing) Then
         strKey = e.key.ToString()
      End If
      Debug.Print(.CellValue(e.item, e.colIndex).ToString() + ", Key = " + strKey)
   End With
End Sub
```

The following C# sample displays the caption of the cell where a button is clicked:

```
private void axG2antt1_ButtonClick(object sender,
AxEXG2ANTTLib._IG2anttEvents_ButtonClickEvent e)
{
   string strKey = "";
   if (e.key != null)
      strKey = e.key.ToString();
   System.Diagnostics.Debug.WriteLine(axG2antt1.Items.get_CellValue(e.item, e.colIndex) +
", Key = " + strKey);
}
```

The following VFP sample displays the caption of the cell where a button is clicked:

```
*** ActiveX Control Event ***
LPARAMETERS item, colindex, key

with thisform.G2antt1.Items
   .DefaultItem = item
   wait window nowait .CellValue(0, colindex)
endwith
```

event CellImageClick (Item as HITEM, ColIndex as Long)

Occurs when the user clicks the cell's icon.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item where the user clicks the cell's icon.
ColIndex as Long	A long expression that indicates the index of the column where the user clicks the cell's icon, or a long expression that indicates the handle of the cell being clicked, if the Item parameter is 0.

The CellImageClick event is fired when user clicks on the cell's image. Use the CellImage property to assign an icon to a cell. Use the CellImages property to assign multiple icons to a cell. Use the ItemFromPoint property to determine the index of the icon being clicked, in case the cell displays multiple icons using the CellImages property. Use the CellHasCheckBox or CellHasRadioButton property to assign a check box or a radio button to a cell.

Syntax for CellImageClick event, **/NET** version, on:

C#
```
private void CellImageClick(object sender,int Item,int ColIndex)
{
}
```

VB
```
Private Sub CellImageClick(ByVal sender As System.Object,ByVal Item As
Integer,ByVal ColIndex As Integer) Handles CellImageClick
End Sub
```

Syntax for CellImageClick event, **/COM** version, on:

C#
```
private void CellImageClick(object sender,
AxEXG2ANTTLib._IG2anttEvents_CellImageClickEvent e)
{
}
```

C++
```
void OnCellImageClick(long Item,long ColIndex)
{
}
```

C++ Builder

```
void __fastcall CellImageClick(TObject *Sender,Exg2anttlib_tlb::HITEM Item,long ColIndex)
{
}
```

Delphi

```
procedure CellImageClick(ASender: TObject; Item : HITEM;ColIndex : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure CellImageClick(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_CellImageClickEvent);
begin
end;
```

Powe…

```
begin event CellImageClick(long Item,long ColIndex)
end event CellImageClick
```

VB.NET

```
Private Sub CellImageClick(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_CellImageClickEvent) Handles CellImageClick
End Sub
```

VB6

```
Private Sub CellImageClick(ByVal Item As EXG2ANTTLibCtl.HITEM,ByVal ColIndex
As Long)
End Sub
```

VBA

```
Private Sub CellImageClick(ByVal Item As Long,ByVal ColIndex As Long)
End Sub
```

VFP

```
LPARAMETERS Item,ColIndex

```

Xbas…

```
PROCEDURE OnCellImageClick(oG2antt,Item,ColIndex)
RETURN
```

Syntax for CellImageClick event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="CellImageClick(Item,ColIndex)" LANGUAGE="JScript">
```

```
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function CellImageClick(Item,ColIndex)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComCellImageClick HITEM llItem Integer llColIndex
    Forward Send OnComCellImageClick llItem llColIndex
End_Procedure
```

Visual Objects

```
METHOD OCX_CellImageClick(Item,ColIndex) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_CellImageClick(int _Item,int _ColIndex)
{
}
```

XBasic

```
function CellImageClick as v (Item as OLE::Exontrol.G2antt.1::HITEM,ColIndex as N)
end function
```

dBASE

```
function nativeObject_CellImageClick(Item,ColIndex)
return
```

The following VB sample assigns an icon to each cell that's added, and changes the cell's icon when the user clicks the icon:

```
Private Sub G2antt1_AddItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
    G2antt1.Items.CellImage(Item, 0) = 1
End Sub

Private Sub G2antt1_CellImageClick(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal ColIndex
As Long)
    G2antt1.Items.CellImage(Item, ColIndex) = G2antt1.Items.CellImage(Item, ColIndex)
Mod 2 + 1
End Sub
```

The following VB sample displays the index of icon being clicked:

```
Private Sub G2antt1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Dim i As HITEM, h As HitTestInfoEnum, c As Long
   With G2antt1
      i = .ItemFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY, c, h)
   End With
   If (i <> 0) or ( c <> 0 ) Then
      If exHTCellIcon = (h And exHTCellIcon) Then
         Debug.Print "The index of icon being clicked is: " & (h And &HFFFF0000) / 65536
      End If
   End If
End Sub
```

The following C++ sample changes the cell's icon being clicked:

```
#include "Items.h"
void OnCellImageClickG2antt1(long Item, long ColIndex)
{
   CItems items = m_g2antt.GetItems();
   COleVariant vtItem( Item ), vtColumn( ColIndex );
   items.SetCellImage( vtItem , vtColumn , items.GetCellImage( vtItem, vtColumn ) % 2 + 1
);
}
```

The following C# sample changes the cell's icon being clicked:

```
private void axG2antt1_CellImageClick(object sender,
AxEXG2ANTTLib._IG2anttEvents_CellImageClickEvent e)
{
   axG2antt1.Items.set_CellImage( e.item, e.colIndex, axG2antt1.Items.get_CellImage(
e.item, e.colIndex ) % 2 + 1 );
}
```

The following VB/NET sample changes the cell's icon being clicked:

```
Private Sub AxG2antt1_CellImageClick(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_CellImageClickEvent) Handles AxG2antt1.CellImageClick
   With AxG2antt1.Items
      .CellImage(e.item, e.colIndex) = .CellImage(e.item, e.colIndex) Mod 2 + 1
```

```
    End With
End Sub
```

The following VFP sample changes the cell's icon being clicked:

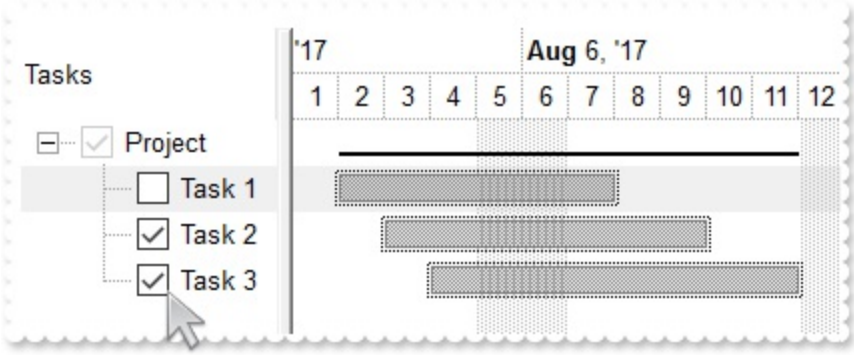
```
*** ActiveX Control Event ***
LPARAMETERS item, colindex

with thisform.G2antt1.Items
    .DefaultItem = item
    .CellImage( 0,colindex ) = .CellImage( 0,colindex ) + 1
endwith
```

event CellStateChanged (Item as HITEM, ColIndex as Long)

Fired after cell's state has been changed.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item where the cell's state is changed.
ColIndex as Long	A long expression that indicates the index of the column where the cell's state is changed, or a long expression that indicates the handle of the cell, if the Item parameter is 0.

A cell that contains a radio button or a check box button fires the CellStateChanged event when its state is changed. The control fires the CellStateChanging event just before cell's state is about to be changed. Use the CellState property to change the cell's state. Use the CellHasRadioButton or CellHasCheckBox property to enable radio or check box button into a cell. Use the Def property to assign check-boxes / radio-buttons for all cells in the column. Use the CellImage property to display an icon in the cell. Use the CellImages property to display multiple icons in the same cell. Use the PartialCheck property to enable partial check feature (check boxes with three states: partial, checked and unchecked). Use the CellChecked property to determine the handle of the cell that's checked in a radio group. Use the CellRadioGroup property to radio group cells.



Once the user clicks a check-box, radio-button, the control fires the following events:

- CellStateChanging event, where the NewState parameter indicates the new state of the cell's checkbox / radio-button. You can change the NewState parameter during this event. For instance, NewState = Items.CellState(Item,ColIndex) un-changes the cell's state once the user tries to change it.

- CellStateChanged event notifies your application that the cell's check-box or radio-button has been changed. The CellState property determines the check-box/radio-button state of the cell.

Syntax for CellStateChanged event, **/NET** version, on:

C#
```
private void CellStateChanged(object sender,int Item,int ColIndex)
{
}
```

VB
```
Private Sub CellStateChanged(ByVal sender As System.Object,ByVal Item As
Integer,ByVal ColIndex As Integer) Handles CellStateChanged
End Sub
```

Syntax for CellStateChanged event, **/COM** version, on:

C#
```
private void CellStateChanged(object sender,
AxEXG2ANTTLib._IG2anttEvents_CellStateChangedEvent e)
{
}
```

C++
```
void OnCellStateChanged(long Item,long ColIndex)
{
}
```

C++ Builder
```
void __fastcall CellStateChanged(TObject *Sender,Exg2anttlib_tlb::HITEM Item,long
ColIndex)
{
}
```

Delphi
```
procedure CellStateChanged(ASender: TObject; Item : HITEM;ColIndex : Integer);
begin
end;
```

Delphi 8 (.NET only)
```
procedure CellStateChanged(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_CellStateChangedEvent);
begin
end;
```

Powe…
```
begin event CellStateChanged(long Item,long ColIndex)
end event CellStateChanged
```

VB.NET
```
Private Sub CellStateChanged(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_CellStateChangedEvent) Handles
```

```
CellStateChanged
End Sub
```

VB6

```
Private Sub CellStateChanged(ByVal Item As EXG2ANTTLibCtl.HITEM,ByVal ColIndex As Long)
End Sub
```

VBA

```
Private Sub CellStateChanged(ByVal Item As Long,ByVal ColIndex As Long)
End Sub
```

VFP

```
LPARAMETERS Item,ColIndex

```

Xbas…

```
PROCEDURE OnCellStateChanged(oG2antt,Item,ColIndex)
RETURN
```

Syntax for CellStateChanged event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="CellStateChanged(Item,ColIndex)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function CellStateChanged(Item,ColIndex)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComCellStateChanged HITEM llItem Integer llColIndex
    Forward Send OnComCellStateChanged llItem llColIndex
End_Procedure
```

Visual Objects

```
METHOD OCX_CellStateChanged(Item,ColIndex) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_CellStateChanged(int _Item,int _ColIndex)
{
}
```

XBasic

```
function CellStateChanged as v (Item as OLE::Exontrol.G2antt.1::HITEM,ColIndex as
```

```
N)
end function
```

dBASE

```
function nativeObject_CellStateChanged(Item,ColIndex)
return
```

The following VB sample displays a message when the user clicks a check box or a radio button:

```
Private Sub G2antt1_CellStateChanged(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal ColIndex As Long)
    Debug.Print "The cell """ & G2antt1.Items.CellValue(Item, ColIndex) & """ has changed its state. The new state is " & IIf(G2antt1.Items.CellState(Item, ColIndes) = 0, "Unchecked", "Checked")
End Sub
```

The following VC sample displays the caption of the cell whose checkbox's state is changed:

```
#include "Items.h"
```

```
static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
   if ( pv )
   {
      if ( pv->vt == VT_ERROR )
         return szDefault;

      COleVariant vt;
      vt.ChangeType( VT_BSTR, pv );
      return V_BSTR( &vt );
   }
   return szDefault;
}
void OnCellStateChangedG2antt1(long Item, long ColIndex)
{
   CItems items = m_g2antt.GetItems();
   COleVariant vtItem( Item ), vtColumn( ColIndex );
```

```
    CString strCellValue = V2S( &items.GetCellValue( vtItem, vtColumn ) );
    CString strOutput;
    strOutput.Format( "'%s''s checkbox state is %i\r\n", strCellValue, items.GetCellState(
vtItem, vtColumn ) );
    OutputDebugString( strOutput );
}
```

The following VB.NET sample displays a message when the user clicks a check box or a radio button:

```
Private Sub AxG2antt1_CellStateChanged(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_CellStateChangedEvent) Handles
AxG2antt1.CellStateChanged
    Debug.WriteLine("The cell """ & AxG2antt1.Items.CellValue(e.item, e.colIndex) & """ has
changed its state. The new state is " & IIf(AxG2antt1.Items.CellState(e.item, e.colIndex) = 0,
"Unchecked", "Checked"))
End Sub
```

The following C# sample outputs a message when the user clicks a check box or a radio button:

```
private void axG2antt1_CellStateChanged(object sender,
AxEXG2ANTTLib._IG2anttEvents_CellStateChangedEvent e)
{
    string strOutput = axG2antt1.Items.get_CellValue( e.item, e.colIndex ).ToString();
    strOutput += " state = " + axG2antt1.Items.get_CellState(e.item, e.colIndex).ToString() ;
    System.Diagnostics.Debug.WriteLine( strOutput );
}
```

The following VFP sample prints a message when the user clicks a check box or a radio button:

```
*** ActiveX Control Event ***
LPARAMETERS item, colindex

local sOutput
sOutput = ""
with thisform.G2antt1.Items
    .DefaultItem = item
```

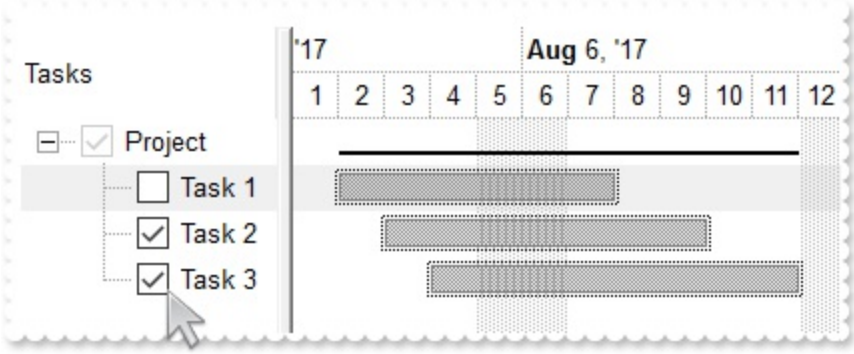
```
    sOutput = .CellValue( 0, colindex )
    sOutput = sOutput + ", state = " + str(.CellState( 0, colindex ))
    wait window nowait sOutput
endwith
```

event CellStateChanging (Item as HITEM, ColIndex as Long, NewState as Long)

Fired before cell's state is about to be changed.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item where the cell's state is about to be changed.
ColIndex as Long	A long expression that indicates the index of the column where the cell's state is changed, or a long expression that indicates the handle of the cell, if the Item parameter is 0.
NewState as Long	A long expression that specifies the new state of the cell (0- unchecked, 1 - checked, 2 - partial checked)

The control fires the CellStateChanging event just before cell's state is about to be changed. For instance, you can prevent changing the cell's state, by calling the NewState = Items.CellState(Item,ColIndex). A cell that contains a radio button or a check box button fires the CellStateChanged event when its state is changed. Use the CellState property to change the cell's state. Use the CellHasRadioButton or CellHasCheckBox property to enable radio or check box button into a cell. Use the Def property to assign check-boxes / radio-buttons for all cells in the column. Use the CellImage property to display an icon in the cell. Use the CellImages property to display multiple icons in the same cell. Use the PartialCheck property to enable partial check feature (check boxes with three states: partial, checked and unchecked). Use the CellChecked property to determine the handle of the cell that's checked in a radio group. Use the CellRadioGroup property to radio group cells. We would not recommend changing the CellState property during the CellStateChanging event, to prevent recursive calls, instead you can change the NewState parameter which is passed by reference.



Once the user clicks a check-box, radio-button, the control fires the following events:

- CellStateChanging event, where the NewState parameter indicates the new state of the cell's checkbox / radio-button.

- CellStateChanged event notifies your application that the cell's check-box or radio-button has been changed. The CellState property determines the check-box/radio-button state of the cell.

For instance, the following VB sample prevents changing the cell's checkbox/radio-button, when the control's ReadOnly property is set:

```
Private Sub G2antt1_CellStateChanging(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal
ColIndex As Long, NewState As Long)
   With G2antt1
      If (.ReadOnly) Then
         With .Items
            NewState = .CellState(Item, ColIndex)
         End With
      End If
   End With
End Sub
```

Syntax for CellStateChanging event, **/NET** version, on:

C#

```
private void CellStateChanging(object sender,int   Item,int   ColIndex,ref int
NewState)
{
}
```

VB

```
Private Sub CellStateChanging(ByVal sender As System.Object,ByVal Item As
Integer,ByVal ColIndex As Integer,ByRef NewState As Integer) Handles
CellStateChanging
End Sub
```

Syntax for CellStateChanging event, **/COM** version, on:

C#

```
private void CellStateChanging(object sender,
AxEXG2ANTTLib._IG2anttEvents_CellStateChangingEvent e)
{
}
```

C++

```
void OnCellStateChanging(long   Item,long   ColIndex,long FAR*   NewState)
{
}
```

C++ Builder

```
void __fastcall CellStateChanging(TObject *Sender,Exg2anttlib_tlb::HITEM
Item,long   ColIndex,long *   NewState)
{
}
```

Delphi

```
procedure CellStateChanging(ASender: TObject; Item : HITEM;ColIndex :
Integer;var NewState : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure CellStateChanging(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_CellStateChangingEvent);
begin
end;
```

Powe…

```
begin event CellStateChanging(long  Item,long  ColIndex,long  NewState)

end event CellStateChanging
```

VB.NET

```
Private Sub CellStateChanging(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_CellStateChangingEvent) Handles
CellStateChanging
End Sub
```

VB6

```
Private Sub CellStateChanging(ByVal Item As EXG2ANTTLibCtl.HITEM,ByVal
ColIndex As Long,NewState As Long)
End Sub
```

VBA

```
Private Sub CellStateChanging(ByVal Item As Long,ByVal ColIndex As
Long,NewState As Long)
End Sub
```

VFP

```
LPARAMETERS Item,ColIndex,NewState

```

Xbas…

```
PROCEDURE OnCellStateChanging(oG2antt,Item,ColIndex,NewState)

RETURN
```

Syntax for CellStateChanging event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="CellStateChanging(Item,ColIndex,NewState)"
LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function CellStateChanging(Item,ColIndex,NewState)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComCellStateChanging HITEM llItem Integer llColIndex Integer
llNewState
    Forward Send OnComCellStateChanging llItem llColIndex llNewState
End_Procedure
```

Visual Objects

```
METHOD OCX_CellStateChanging(Item,ColIndex,NewState) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_CellStateChanging(int _Item,int _ColIndex,COMVariant /*long*/
_NewState)
{
}
```

XBasic

```
function CellStateChanging as v (Item as OLE::Exontrol.G2antt.1::HITEM,ColIndex
as N,NewState as N)
end function
```

dBASE

```
function nativeObject_CellStateChanging(Item,ColIndex,NewState)
return
```

event Change (Item as HITEM, ColIndex as Long, NewValue as Variant)

Occurs when the user changes the cell's content.

Type	Description
Item as HITEM	A long expression that determines the item's handle. If the Item parameter is 0, and the ColIndex property is different than zero, the ColIndex indicates the handle of the cell where the state is changed.
ColIndex as Long	A long expression that indicates the column's index, if the Item parameter is not zero, a long expression that indicates the handle of the cell if the Item parameter is 0.
NewValue as Variant	A Variant value that indicates the changed cell's value

The Change event notifies your application that the user changes the control's content. The Change event is fired when the CellValue property is changed. During the Change event it is possible to have *recursive calls*, if you are changing the CellValue property (only when you assign a value to a cell, not when you are retrieving the cell's value).

Syntax for Change event, **/NET** version, on:

C#
```
private void Change(object sender,int Item,int ColIndex,ref object NewValue)
{
}
```

VB
```
Private Sub Change(ByVal sender As System.Object,ByVal Item As Integer,ByVal
ColIndex As Integer,ByRef NewValue As Object) Handles Change
End Sub
```

Syntax for Change event, **/COM** version, on:

C#
```
private void Change(object sender, AxEXG2ANTTLib._IG2anttEvents_ChangeEvent
e)
{
}
```

C++
```
void OnChange(long Item,long ColIndex,VARIANT FAR* NewValue)
{
}
```

C++ Builder

```
void __fastcall Change(TObject *Sender,Exg2anttlib_tlb::HITEM Item,long ColIndex,Variant * NewValue)
{
}
```

Delphi

```
procedure Change(ASender: TObject; Item : HITEM;ColIndex : Integer;var NewValue : OleVariant);
begin
end;
```

Delphi 8 (.NET only)

```
procedure Change(sender: System.Object; e: AxEXG2ANTTLib._IG2anttEvents_ChangeEvent);
begin
end;
```

Powe…

```
begin event Change(long Item,long ColIndex,any NewValue)
end event Change
```

VB.NET

```
Private Sub Change(ByVal sender As System.Object, ByVal e As AxEXG2ANTTLib._IG2anttEvents_ChangeEvent) Handles Change
End Sub
```

VB6

```
Private Sub Change(ByVal Item As EXG2ANTTLibCtl.HITEM,ByVal ColIndex As Long,NewValue As Variant)
End Sub
```

VBA

```
Private Sub Change(ByVal Item As Long,ByVal ColIndex As Long,NewValue As Variant)
End Sub
```

VFP

```
LPARAMETERS Item,ColIndex,NewValue

```

Xbas…

```
PROCEDURE OnChange(oG2antt,Item,ColIndex,NewValue)
RETURN
```

Syntax for Change event, */COM* version (others), on:

Java…
```
<SCRIPT EVENT="Change(Item,ColIndex,NewValue)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function Change(Item,ColIndex,NewValue)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComChange HITEM llItem Integer llColIndex Variant llNewValue
    Forward Send OnComChange llItem llColIndex llNewValue
End_Procedure
```

Visual Objects
```
METHOD OCX_Change(Item,ColIndex,NewValue) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_Change(int _Item,int _ColIndex,COMVariant /*variant*/ _NewValue)
{
}
```

XBasic
```
function Change as v (Item as OLE::Exontrol.G2antt.1::HITEM,ColIndex as
N,NewValue as A)
end function
```

dBASE
```
function nativeObject_Change(Item,ColIndex,NewValue)
return
```

If you are changing the *other cell's value*, during the Change event you have to add a C++ code like follows in order to avoid *recursive calls*:

```
static sg_ChangeCounter = 0;
void OnChangeG2antt1(long Item, long ColIndex, VARIANT FAR* NewValue)
{
   if ( sg_ChangeCounter == 0)
   {
      sg_ChangeCounter++;
```

```
        m_Items.SetCellValue( COleVariant( Item ), COleVariant( (long)othercolumn ),
*NewValue );
        sg_ChangeCounter--;
    }
}
```

or in VB you could have like this:

```
Private sg_ChangeCounter As Long
Private Sub G2antt1_Change(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal ColIndex As
Long, NewValue As Variant)
    If (sg_ChangeCounter = 0) Then
        sg_ChangeCounter = sg_ChangeCounter + 1
        G2antt1.Items.CellValue(Item, othercolumn) = NewValue
        sg_ChangeCounter = sg_ChangeCounter - 1
    End If
End Sub
```

Use the CellEditor or Editor property to assign an editor to a cell or to a column. Use the Edit event to notify your application that the editing operation begins. The Change event notifies that the editing focused cell ended. If the control is bounded to an ADO recordset the Change event is automatically called when the user changes the focused cell, and it updates the recordset too. The control fires the ValidateValue event before calling the Change event, if the CauseValidateValue property is True. Please note that the Change event is called also when loading, or adding new items , so you need to use an internal counter (like explained bellow) to avoid calling the Change event during adding or loading the items, if it is not case (increases the iChanging variable before loading items, and decreases the iChanging member when adding items is done). Call the Refresh method, when changing the value for a cell that has the CellSingleLine property on False.

The following VB sample displays the newly value of the focused cell:

```
Private Sub G2antt1_Change(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal ColIndex As
Long, NewValue As Variant)
    ' Displays the old/new cell's value
    Debug.Print "The current cell's value is '" & G2antt1.Items.CellValue(Item, ColIndex) &
"'."
    Debug.Print "The newly cell's value is '" & NewValue & "'."
End Sub
```

You can change the newly cell's value by changing the NewValue parameter of the Change event. If you are changing the CellValue property during the Change event a recursive calls occurs, so you need to protect recursive calls using an internal counter that's increased when Change event starts, and decreased when the Change event ends like in the following VB sample:

```
Private iChanging As Long

Private Sub G2antt1_Change(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal ColIndex As
Long, NewValue As Variant)
   If (iChanging = 0) Then
      iChanging = iChanging + 1
       ' here's safe to change the Items.CellValue property
      iChanging = iChanging - 1
   End If
End Sub
```

The following sample is the C++ equivalent:

```
long iChanging = 0;
void OnChangeG2antt1(long Item, long ColIndex, VARIANT FAR* NewValue)
{
   if ( iChanging == 0 )
   {
      iChanging++;
      // here's safe to call Items.CellValue property, to avoid recursive calls.
      iChanging--;
   }
}
```

The following C++ sample displays the newly value of the focused cell:

```
#include "Items.h"
void OnChangeG2antt1(long Item, long ColIndex, VARIANT FAR* NewValue)
{
   if ( ::IsWindow( m_g2antt.m_hWnd ) )
   {
      CItems items = m_g2antt.GetItems();
      COleVariant vtItem( Item ), vtColumn( ColIndex );
```

```
        CString strFormat;
        strFormat.Format( "'%s' = %s", V2S( &items.GetCellValue( vtItem, vtColumn ) ), V2S(
NewValue ) );
        OutputDebugString( strFormat );
    }
}
```

where the V2S function converts a VARIANT to a string value, and may look like follows:

```
static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
    if ( pv )
    {
        if ( pv->vt == VT_ERROR )
            return szDefault;

        COleVariant vt;
        vt.ChangeType( VT_BSTR, pv );
        return V_BSTR( &vt );
    }
    return szDefault;
}
```

The following VB.NET sample displays the newly value of the focused cell:

```
Private Sub AxG2antt1_Change(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_ChangeEvent) Handles AxG2antt1.Change
    With AxG2antt1.Items
        Debug.Print("Old Value: " & .CellValue(e.item, e.colIndex) & " New Value " &
e.newValue.ToString())
    End With
End Sub
```

The following C# sample displays the newly value of the focused cell:

```
private void axG2antt1_Change(object sender,
AxEXG2ANTTLib._IG2anttEvents_ChangeEvent e)
{
    System.Diagnostics.Debug.WriteLine("Old Value " +
```

```
axG2antt1.Items.get_CellValue(e.item, e.colIndex).ToString() + " New Value " +
e.newValue.ToString());
}
```

The following VFP sample displays the newly value of the focused cell:

```
*** ActiveX Control Event ***
LPARAMETERS item, colindex, newvalue

with thisform.G2antt1.Items
    .DefaultItem = item
    local oldvalue
    oldvalue = .CellValue(0,colindex)
    wait window nowait "Old Value " + str(oldvalue)
    wait window nowait "New Value " + str(newvalue)
endwith
```

event ChartEndChanging (Operation as BarOperationEnum)

Occurs when the chart is about to be changed.

Type	Description
Operation as BarOperationEnum	A BarOperationEnum expression that specifies the operation that ends

The ChartEndChanging event notifies your application once the user ends resizing or moving a bar at runtime using the mouse. The ChartStartChanging event occurs once the operation beings. The ChartEndChaning event is fired in the following cases:

- Move bars
- Resizes the start of the bar.
- Resizes the end of the bar.
- Adds a new link.
- Resizes the percent value of the bar.
- Creates a new bar.
- Resizes a time scale unit in the base level area.
- Magnifies a time scale unit by double clicking the base level area.
- The user selects or unselects a date.
- The user moves the chart's vertical splitter.
- The user moves the chart's horizontal splitter (histogram)
- An Undo/Redo operation is performed.

Syntax for ChartEndChanging event, **/NET** version, on:

C#
```
private void ChartEndChanging(object
sender,exontrol.EXG2ANTTLib.BarOperationEnum Operation)
{
}
```

VB
```
Private Sub ChartEndChanging(ByVal sender As System.Object,ByVal Operation As
exontrol.EXG2ANTTLib.BarOperationEnum) Handles ChartEndChanging
End Sub
```

Syntax for ChartEndChanging event, **/COM** version, on:

C#
```
private void ChartEndChanging(object sender,
AxEXG2ANTTLib._IG2anttEvents_ChartEndChangingEvent e)
{
}
```

C++

```
void OnChartEndChanging(long Operation)
{
}
```

C++ Builder

```
void __fastcall ChartEndChanging(TObject
*Sender,Exg2anttlib_tlb::BarOperationEnum Operation)
{
}
```

Delphi

```
procedure ChartEndChanging(ASender: TObject; Operation : BarOperationEnum);
begin
end;
```

Delphi 8 (.NET only)

```
procedure ChartEndChanging(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_ChartEndChangingEvent);
begin
end;
```

Powe…

```
begin event ChartEndChanging(long Operation)
end event ChartEndChanging
```

VB.NET

```
Private Sub ChartEndChanging(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_ChartEndChangingEvent) Handles
ChartEndChanging
End Sub
```

VB6

```
Private Sub ChartEndChanging(ByVal Operation As
EXG2ANTTLibCtl.BarOperationEnum)
End Sub
```

VBA

```
Private Sub ChartEndChanging(ByVal Operation As Long)
End Sub
```

VFP

```
LPARAMETERS Operation

```

Xbas…

```
PROCEDURE OnChartEndChanging(oG2antt,Operation)
RETURN
```

Syntax for ChartEndChanging event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="ChartEndChanging(Operation)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function ChartEndChanging(Operation)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComChartEndChanging OLEBarOperationEnum llOperation
    Forward Send OnComChartEndChanging llOperation
End_Procedure
```

Visual Objects
```
METHOD OCX_ChartEndChanging(Operation) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_ChartEndChanging(int _Operation)
{
}
```

XBasic
```
function ChartEndChanging as v (Operation as
OLE::Exontrol.G2antt.1::BarOperationEnum)
end function
```

dBASE
```
function nativeObject_ChartEndChanging(Operation)
return
```

For instance, you can use the ChartStartChanging event to shows the grid lines while resizing, and use the ChartEndChaning to hide the grid lines.

Use the StartBlockUndoRedo / EndBlockUndoRedo methods to collect the user operations as a block, so next time the Undo/Redo operation is performed, the entire block of operations is performed or restored at once. For instance, if you have a bar related to several other bars, and so moving a bar implies moving several other bars, each moving is recorded as a single undo/redo operation, so the operations are restored once at the time. Instead, if you use the StartBlockUndoRedo / EndBlockUndoRedo methods when your operation starts / ends, the collection of operations is recorded as a block of instructions,

so the next time Undo operation is called the entire block is restored or performed at once.

event ChartSelectionChanged ()

Occurs when the user selects objects in the chart area.

Type	Description

The ChartSelectionChanged event notifies your application when the user select objects like bars or links in the chart area. Use the AllowSelectObjects property to specify whether the user can select bars or/and links at runtime, using the mouse. Use the SelectedObject property to retrieve a collection of selected bars or/and links. Use the ItemBar(exBarSelected) property to select or unselect programmatically a bar. Use the Link(exLinkSelected) property to select or unselect programmatically a link. Use the RemoveSelection property to remove objects in the chart's selection. Use the ExecuteTemplate property to execute and returns the result of a x-script.

Syntax for ChartSelectionChanged event, **/NET** version, on:

C#
```
private void ChartSelectionChanged(object sender)
{
}
```

VB
```
Private Sub ChartSelectionChanged(ByVal sender As System.Object) Handles
ChartSelectionChanged
End Sub
```

Syntax for ChartSelectionChanged event, **/COM** version, on:

C#
```
private void ChartSelectionChanged(object sender, EventArgs e)
{
}
```

C++
```
void OnChartSelectionChanged()
{
}
```

C++ Builder
```
void __fastcall ChartSelectionChanged(TObject *Sender)
{
}
```

Delphi
```
procedure ChartSelectionChanged(ASender: TObject; );
begin
```

```
end;
```

Delphi 8 (.NET only)

```
procedure ChartSelectionChanged(sender: System.Object; e: System.EventArgs);
begin
end;
```

Powe…

```
begin event ChartSelectionChanged()
end event ChartSelectionChanged
```

VB.NET

```
Private Sub ChartSelectionChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ChartSelectionChanged
End Sub
```

VB6

```
Private Sub ChartSelectionChanged()
End Sub
```

VBA

```
Private Sub ChartSelectionChanged()
End Sub
```

VFP

```
LPARAMETERS nop

```

Xbas…

```
PROCEDURE OnChartSelectionChanged(oG2antt)
RETURN
```

Syntax for ChartSelectionChanged event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="ChartSelectionChanged()" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function ChartSelectionChanged()
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComChartSelectionChanged
   Forward Send OnComChartSelectionChanged
End_Procedure
```

Visual Objects

```
METHOD OCX_ChartSelectionChanged() CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_ChartSelectionChanged()
{
}
```

XBasic

```
function ChartSelectionChanged as v ()
end function
```

dBASE

```
function nativeObject_ChartSelectionChanged()
return
```

The following VB sample displays the list of selected bars:

```
Private Sub G2antt1_ChartSelectionChanged()
    Dim c As Variant
    For Each c In G2antt1.Items.SelectedObjects(exSelectBarsOnly)
        Debug.Print c
    Next
End Sub
```

The following VB sample displays only the bars being selected and un-selected since last selection change event:

```
Private Sub G2antt1_ChartSelectionChanged()
    Dim c As Variant
    With G2antt1
        For Each c In .Items.SelectedObjects(exSelectBarsOnly Or exObjectsJustAdded)
            Debug.Print "Added " & .ExecuteTemplate("Items.ItemBar(" & c & "," & exBarName
& ")")
        Next
        For Each c In .Items.SelectedObjects(exSelectBarsOnly Or exObjectsJustRemoved)
            Debug.Print "Removed " & .ExecuteTemplate("Items.ItemBar(" & c & "," &
exBarName & ")")
        Next
    End With
End Sub
```

The following VB sample displays the name of the bars being selected:

```
Private Sub G2antt1_ChartSelectionChanged()
   Dim c As Variant
   With G2antt1
      For Each c In .Items.SelectedObjects(exSelectBarsOnly)
         Debug.Print .ExecuteTemplate("Items.ItemBar(" & c & "," & exBarName & ")")
      Next
   End With
End Sub
```

The following VB.NET sample displays the list of selected bars:

```
Private Sub AxG2antt1_ChartSelectionChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles AxG2antt1.ChartSelectionChanged
   Dim c As String
   For Each c In
AxG2antt1.Items.SelectedObjects(EXG2ANTTLib.SelectObjectsEnum.exSelectBarsOnly)
      Debug.Print(c)
   Next
End Sub
```

The following VB.NET sample displays only the bars being selected and un-selected since last selection change event:

```
Private Sub AxG2antt1_ChartSelectionChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles AxG2antt1.ChartSelectionChanged
   Dim c As String
   With AxG2antt1
      For Each c In
.Items.SelectedObjects(EXG2ANTTLib.SelectObjectsEnum.exSelectBarsOnly Or
EXG2ANTTLib.SelectObjectsEnum.exObjectsJustAdded)
         Dim t As String = "Items.ItemBar(" + c + "," +
Int(EXG2ANTTLib.ItemBarPropertyEnum.exBarName).ToString() + ")"
         Debug.Print("Added: " + .ExecuteTemplate(t))
      Next
      For Each c In
.Items.SelectedObjects(EXG2ANTTLib.SelectObjectsEnum.exSelectBarsOnly Or
```

```
EXG2ANTTLib.SelectObjectsEnum.exObjectsJustRemoved)
        Dim t As String = "Items.ItemBar(" + c + "," +
Int(EXG2ANTTLib.ItemBarPropertyEnum.exBarName).ToString() + ")"
        Debug.Print("Removed: " + .ExecuteTemplate(t))
    Next
  End With
End Sub
```

The following VB.NET sample displays the name of the bars being selected:

```
Private Sub AxG2antt1_ChartSelectionChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles AxG2antt1.ChartSelectionChanged
  Dim c As String
  With AxG2antt1
    For Each c In
.Items.SelectedObjects(EXG2ANTTLib.SelectObjectsEnum.exSelectBarsOnly)
        Dim t As String = "Items.ItemBar(" + c + "," +
Int(EXG2ANTTLib.ItemBarPropertyEnum.exBarName).ToString() + ")"
        Debug.Print(.ExecuteTemplate(t))
    Next
  End With
End Sub
```

The following C# sample displays the list of selected bars:

```
private void axG2antt1_ChartSelectionChanged(object sender, EventArgs e)
{
  foreach (string c in
axG2antt1.Items.get_SelectedObjects(EXG2ANTTLib.SelectObjectsEnum.exSelectBarsOnly)
as Array )
  {
    System.Diagnostics.Debug.WriteLine( c );
  }
}
```

The following C# sample displays only the bars being selected and un-selected since last selection change event:

```
private void axG2antt1_ChartSelectionChanged(object sender, EventArgs e)
```

```
{
   foreach (string c in
axG2antt1.Items.get_SelectedObjects(EXG2ANTTLib.SelectObjectsEnum.exSelectBarsOnly |
EXG2ANTTLib.SelectObjectsEnum.exObjectsJustAdded) as Array)
   {
      String t = "Items.ItemBar(" + c + "," +
((long)EXG2ANTTLib.ItemBarPropertyEnum.exBarName).ToString() + ")";
      System.Diagnostics.Debug.WriteLine("Added: " + axG2antt1.ExecuteTemplate(t));
   }
   foreach (string c in
axG2antt1.Items.get_SelectedObjects(EXG2ANTTLib.SelectObjectsEnum.exSelectBarsOnly |
EXG2ANTTLib.SelectObjectsEnum.exObjectsJustRemoved) as Array)
   {
      String t = "Items.ItemBar(" + c + "," +
((long)EXG2ANTTLib.ItemBarPropertyEnum.exBarName).ToString() + ")";
      System.Diagnostics.Debug.WriteLine("Removed: " + axG2antt1.ExecuteTemplate(t));
   }
}
```

The following C# sample displays the name of the bars being selected:

```
private void axG2antt1_ChartSelectionChanged(object sender, EventArgs e)
{
   foreach (string c in
axG2antt1.Items.get_SelectedObjects(EXG2ANTTLib.SelectObjectsEnum.exSelectBarsOnly)
as Array)
   {
      String t = "Items.ItemBar(" + c + "," +
((long)EXG2ANTTLib.ItemBarPropertyEnum.exBarName).ToString() + ")";
      System.Diagnostics.Debug.WriteLine(axG2antt1.ExecuteTemplate(t));
   }
}
```

The following C++ sample displays the list of selected bars:

```
#include "Items.h"
void OnChartSelectionChangedG2antt1()
{
```

```
   COleVariant vtSelected = m_g2antt.GetItems().GetSelectedObjects( 1 );
//exSelectBarsOnly
   if ( V_VT( &vtSelected ) & VT_ARRAY | VT_VARIANT )
   {
      SAFEARRAY* pArray = V_ARRAY( &vtSelected );
      void* pData = NULL;
      if ( SUCCEEDED( SafeArrayAccessData( pArray, &pData ) ) )
      {
         VARIANT* p = (VARIANT*)pData;
         for ( long i = 0; i < (long)pArray->rgsabound[0].cElements ; i++, p++ )
            OutputDebugString( V2S( p ) );
         SafeArrayUnaccessData( pArray );
      }
   }
}
```

The following C++ sample displays only the bars being selected and un-selected since last selection change event:

```
void OnChartSelectionChangedG2antt1()
{
   COleVariant vtAdded = m_g2antt.GetItems().GetSelectedObjects( 1
/*exSelectBarsOnly*/ | 0x20 /*exObjectsJustAdded*/ );
   if ( V_VT( &vtAdded ) & VT_ARRAY | VT_VARIANT )
   {
      SAFEARRAY* pArray = V_ARRAY( &vtAdded );
      void* pData = NULL;
      if ( SUCCEEDED( SafeArrayAccessData( pArray, &pData ) ) )
      {
         VARIANT* p = (VARIANT*)pData;
         for ( long i = 0; i < (long)pArray->rgsabound[0].cElements ; i++, p++ )
         {
            CString strT = "Items.ItemBar(" + V2S( p ) + ",0)"; /*builds the
Items.ItemBar(Handle,Key,exBarName) template*/
            OutputDebugString( "Added: " + V2S( &m_g2antt.ExecuteTemplate( strT ) ) +
"\n" );
         }
         SafeArrayUnaccessData( pArray );
```

```
      }
   }
   COleVariant vtRemoved = m_g2antt.GetItems().GetSelectedObjects( 1
/*exSelectBarsOnly*/ | 0x40 /*exObjectsJustRemoved*/ );
   if ( V_VT( &vtRemoved ) & VT_ARRAY | VT_VARIANT )
   {
      SAFEARRAY* pArray = V_ARRAY( &vtRemoved );
      void* pData = NULL;
      if ( SUCCEEDED( SafeArrayAccessData( pArray, &pData ) ) )
      {
         VARIANT* p = (VARIANT*)pData;
         for ( long i = 0; i < (long)pArray->rgsabound[0].cElements ; i++, p++ )
         {
            CString strT = "Items.ItemBar(" + V2S( p ) + ",0)"; /*builds the
Items.ItemBar(Handle,Key,exBarName) template*/
            OutputDebugString( "Removed: " + V2S( &m_g2antt.ExecuteTemplate( strT ) ) +
"\n" );
         }
         SafeArrayUnaccessData( pArray );
      }
   }
}
```

The following C++ sample displays the name of the bars being selected:

```
#include "Items.h"
void OnChartSelectionChangedG2antt1()
{
   COleVariant vtSelected = m_g2antt.GetItems().GetSelectedObjects( 1
/*exSelectBarsOnly*/ );
   if ( V_VT( &vtSelected ) & VT_ARRAY | VT_VARIANT )
   {
      SAFEARRAY* pArray = V_ARRAY( &vtSelected );
      void* pData = NULL;
      if ( SUCCEEDED( SafeArrayAccessData( pArray, &pData ) ) )
      {
         VARIANT* p = (VARIANT*)pData;
```

```
        for ( long i = 0; i < (long)pArray->rgsabound[0].cElements ; i++, p++ )
        {
            CString strT = "Items.ItemBar(" + V2S( p ) + ",0)"; /*builds the
Items.ItemBar(Handle,Key,exBarName) template*/
            OutputDebugString( V2S( &m_g2antt.ExecuteTemplate( strT ) ) );
        }
        SafeArrayUnaccessData( pArray );
    }
  }
}
```

where the V2S string may look like follows:

```
static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
  if ( pv )
  {
    if ( pv->vt == VT_ERROR )
      return szDefault;

    COleVariant vt;
    vt.ChangeType( VT_BSTR, pv );
    return V_BSTR( &vt );
  }
  return szDefault;
}
```

or

```
static string V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
  if ( pv )
  {
    if ( pv->vt == VT_ERROR )
      return szDefault;

    CComVariant vt;
    if ( SUCCEEDED( vt.ChangeType( VT_BSTR, pv ) ) )
```

```
    {
        USES_CONVERSION;
        return OLE2T(V_BSTR( &vt ));
    }
  }
  return szDefault;
}
```

The following VFP sample displays the list of selected bars:

```
*** ActiveX Control Event ***

local c
For Each c In thisform.G2antt1.Items.SelectedObjects(1)
  wait window c
Next
```

The following VFP sample displays only the bars being selected and un-selected since last selection change event:

```
*** ActiveX Control Event ***

local c
For Each c In thisform.G2antt1.Items.SelectedObjects( 1 + 0x20 )
  local t
  t = "Items.ItemBar(" + c + ",0)"
  wait window "Added: " + thisform.G2antt1.ExecuteTemplate(t)
Next
For Each c In thisform.G2antt1.Items.SelectedObjects( 1 + 0x40 )
  local t
  t = "Items.ItemBar(" + c + ",0)"
  wait window "Removed: " + thisform.G2antt1.ExecuteTemplate(t)
Next
```

The following VFP sample displays the name of the bars being selected:

```
*** ActiveX Control Event ***

local c
```

```
For Each c In thisform.G2antt1.Items.SelectedObjects(1)
   local t
   t = "Items.ItemBar(" + c + ",0)"
   wait window thisform.G2antt1.ExecuteTemplate(t)
Next
```

event ChartStartChanging (Operation as BarOperationEnum)

Occurs when the chart is about to be changed.

Type	Description
Operation as BarOperationEnum	A BarOperationEnum expression that specifies the operation to start

The ChartStartChaning event notifies your application once the user starts resizing or moving a bar at runtime using the mouse. The ChartEndChanging event occurs once the operation ends. The ChartStartChaning event is fired in the following cases:

- Move bars
- Resizes the start of the bar.
- Resizes the end of the bar.
- Adds a new link.
- Resizes the percent value of the bar.
- Creates a new bar.
- Resizes a time scale unit in the base level area.
- Magnifies a time scale unit by double clicking the base level area.
- The user selects or unselects a date.
- The user moves the chart's vertical splitter.
- The user moves the chart's horizontal splitter (histogram)
- An Undo/Redo operation is performed.

Syntax for ChartStartChanging event, **/NET** version, on:

C#
```
private void ChartStartChanging(object
sender,exontrol.EXG2ANTTLib.BarOperationEnum Operation)
{
}
```

VB
```
Private Sub ChartStartChanging(ByVal sender As System.Object,ByVal Operation
As exontrol.EXG2ANTTLib.BarOperationEnum) Handles ChartStartChanging
End Sub
```

Syntax for ChartStartChanging event, **/COM** version, on:

C#
```
private void ChartStartChanging(object sender,
AxEXG2ANTTLib._IG2anttEvents_ChartStartChangingEvent e)
{
}
```

C++

```
void OnChartStartChanging(long Operation)
{
}
```

C++ Builder

```
void __fastcall ChartStartChanging(TObject
*Sender,Exg2anttlib_tlb::BarOperationEnum Operation)
{
}
```

Delphi

```
procedure ChartStartChanging(ASender: TObject; Operation :
BarOperationEnum);
begin
end;
```

Delphi 8 (.NET only)

```
procedure ChartStartChanging(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_ChartStartChangingEvent);
begin
end;
```

Powe…

```
begin event ChartStartChanging(long Operation)
end event ChartStartChanging
```

VB.NET

```
Private Sub ChartStartChanging(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_ChartStartChangingEvent) Handles
ChartStartChanging
End Sub
```

VB6

```
Private Sub ChartStartChanging(ByVal Operation As
EXG2ANTTLibCtl.BarOperationEnum)
End Sub
```

VBA

```
Private Sub ChartStartChanging(ByVal Operation As Long)
End Sub
```

VFP

```
LPARAMETERS Operation

```

Xbas…

```
PROCEDURE OnChartStartChanging(oG2antt,Operation)
```

```
RETURN
```

Syntax for ChartStartChanging event, **/COM** version (others), on:

Java...
```
<SCRIPT EVENT="ChartStartChanging(Operation)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc...
```
<SCRIPT LANGUAGE="VBScript">
Function ChartStartChanging(Operation)
End Function
</SCRIPT>
```

Visual Data...
```
Procedure OnComChartStartChanging OLEBarOperationEnum llOperation
    Forward Send OnComChartStartChanging llOperation
End_Procedure
```

Visual Objects
```
METHOD OCX_ChartStartChanging(Operation) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_ChartStartChanging(int _Operation)
{
}
```

XBasic
```
function ChartStartChanging as v (Operation as
OLE::Exontrol.G2antt.1::BarOperationEnum)
end function
```

dBASE
```
function nativeObject_ChartStartChanging(Operation)
return
```

For instance, you can use the ChartStartChanging event to shows the grid lines while resizing, and use the ChartEndChaning to hide the grid lines.

Use the StartBlockUndoRedo / EndBlockUndoRedo methods to collect the user operations as a block, so next time the Undo/Redo operation is performed, the entire block of operations is performed or restored at once. For instance, if you have a bar related to several other bars, and so moving a bar implies moving several other bars, each moving is recorded as a single undo/redo operation, so the operations are restored once at the time.

Instead, if you use the StartBlockUndoRedo / EndBlockUndoRedo methods when your operation starts / ends, the collection of operations is recorded as a block of instructions, so the next time Undo operation is called the entire block is restored or performed at once.

event Click ()

Occurs when the user presses and then releases the left mouse button over the control.

Type	Description

The Click event is fired when the user releases the left mouse button over the control. Use a MouseDown or MouseUp event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the Click and DblClick events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers.

Syntax for Click event, **/NET** version, on:

C#

```
private void Click(object sender)
{
}
```

VB

```
Private Sub Click(ByVal sender As System.Object) Handles Click
End Sub
```

Syntax for Click event, **/COM** version, on:

C#

```
private void ClickEvent(object sender, EventArgs e)
{
}
```

C++

```
void OnClick()
{
}
```

C++ Builder

```
void __fastcall Click(TObject *Sender)
{
}
```

Delphi

```
procedure Click(ASender: TObject; );
begin
end;
```

Delphi 8 (.NET only)

```
procedure ClickEvent(sender: System.Object; e: System.EventArgs);
begin
end;
```

Powe…

```
begin event Click()
end event Click
```

VB.NET

```
Private Sub ClickEvent(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ClickEvent
End Sub
```

VB6

```
Private Sub Click()
End Sub
```

VBA

```
Private Sub Click()
End Sub
```

VFP

```
LPARAMETERS nop
```

Xbas…

```
PROCEDURE OnClick(oG2antt)
RETURN
```

Syntax for Click event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="Click()" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function Click()
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComClick
    Forward Send OnComClick
```

```
End_Procedure
```

Visual Objects

```
METHOD OCX_Click() CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_Click()
{
}
```

XBasic

```
function Click as v ()
end function
```

dBASE

```
function nativeObject_Click()
return
```

event ColumnClick (Column as Column)

Fired after the user clicks on column's header.

Type	Description
Column as Column	A Column object that indicates clicked column.

The ColumnClick event is fired when the user clicks the column's header. By default, the control sorts by the column when user clicks the column's header. Use the SortOnClick property to specify the operation that control does when user clicks the column's caption. Use the ColumnFromPoint property to access the column from point. Use the ItemFromPoint property to access the item from point. The control fires Sort method when the control sorts a column. Use the MouseDown or MouseUp event to notify the control when the user clicks the control, including the columns.

Syntax for ColumnClick event, **/NET** version, on:

C#
```
private void ColumnClick(object sender,exontrol.EXG2ANTTLib.Column Column)
{
}
```

VB
```
Private Sub ColumnClick(ByVal sender As System.Object,ByVal Column As
exontrol.EXG2ANTTLib.Column) Handles ColumnClick
End Sub
```

Syntax for ColumnClick event, **/COM** version, on:

C#
```
private void ColumnClick(object sender,
AxEXG2ANTTLib._IG2anttEvents_ColumnClickEvent e)
{
}
```

C++
```
void OnColumnClick(LPDISPATCH Column)
{
}
```

C++ Builder
```
void __fastcall ColumnClick(TObject *Sender,Exg2anttlib_tlb::IColumn *Column)
{
}
```

Delphi

```
procedure ColumnClick(ASender: TObject; Column : IColumn);
begin
end;
```

Delphi 8 (.NET only)

```
procedure ColumnClick(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_ColumnClickEvent);
begin
end;
```

Powe…

```
begin event ColumnClick(oleobject Column)
end event ColumnClick
```

VB.NET

```
Private Sub ColumnClick(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_ColumnClickEvent) Handles ColumnClick
End Sub
```

VB6

```
Private Sub ColumnClick(ByVal Column As EXG2ANTTLibCtl.IColumn)
End Sub
```

VBA

```
Private Sub ColumnClick(ByVal Column As Object)
End Sub
```

VFP

```
LPARAMETERS Column
```

Xbas…

```
PROCEDURE OnColumnClick(oG2antt,Column)
RETURN
```

Syntax for ColumnClick event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="ColumnClick(Column)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function ColumnClick(Column)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComColumnClick Variant llColumn
    Forward Send OnComColumnClick llColumn
End_Procedure
```

Visual Objects

```
METHOD OCX_ColumnClick(Column) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_ColumnClick(COM _Column)
{
}
```

XBasic

```
function ColumnClick as v (Column as OLE::Exontrol.G2antt.1::IColumn)
end function
```

dBASE

```
function nativeObject_ColumnClick(Column)
return
```

The following VB sample displays the caption of the column being clicked:

```
Private Sub G2antt1_ColumnClick(ByVal Column As EXG2ANTTLibCtl.IColumn)
    Debug.Print Column.Caption
End Sub
```

The following C++ sample displays the caption of the column being clicked:

```
#include "Column.h"
void OnColumnClickG2antt1(LPDISPATCH Column)
{
    CColumn column( Column );
    column.m_bAutoRelease = FALSE;
    MessageBox( column.GetCaption() );

}
```

The following VB.NET sample displays the caption of the column being clicked:

```
Private Sub AxG2antt1_ColumnClick(ByVal sender As Object, ByVal e As
```

```
AxEXG2ANTTLib._IG2anttEvents_ColumnClickEvent) Handles AxG2antt1.ColumnClick
    MessageBox.Show(e.column.Caption)
End Sub
```

The following C# sample displays the caption of the column being clicked:

```
private void axG2antt1_ColumnClick(object sender,
AxEXG2ANTTLib._IG2anttEvents_ColumnClickEvent e)
{
    MessageBox.Show( e.column.Caption );
}
```

The following VFP sample displays the caption of the column being clicked:

```
*** ActiveX Control Event ***
LPARAMETERS column

with column
    wait window nowait .Caption
endwith
```

event CreateBar (Item as HITEM, DateStart as Date, DateEnd as Date)

Fired when the user creates a new bar.

Type	Description
Item as HITEM	A HITEM expression that indicates the handle of the item where the bar is created. Newer versions of the component, may pass the Item parameter as a negative value, which indicates the number of new items you must add in order to cover the clicked area, if the AllowCreateBar property is exCreateBarManual.
DateStart as Date	A DATE expression that indicates where the bar starts.
DateEnd as Date	A DATE expression that indicates where the bar ends.

The CreateBar event is fired when the user releases the mouse in the chart area. The CreateBar event is fired only if the AllowCreateBar property is not zero. *By default, the AllowCreateBar property is exCreateBarManual*.

- If the AllowCreateBar property is **exCreateBarAuto**, the control automatically adds a new bar to the item, with the key "newbar", of "Task" type, so it looks like this: . Use the ItemBar property to change the key or the name or any other property of the newly created bar whose exBarKey property is "newbar" and it's exBarName is "Task". In this case, if the CreateBar event is not handled, the user can't add more than a single bar to the selected item, as the "newbar" is not unique, instead, if you handle the CreateBar event, and assign a different key for the newly created bar, several bars can be added to the same item. If the user clicks the empty / non-items zone of the chart, the control may fire the AddItem event for the newly added items so the newly bar will be shown in the clicked area. In this case, the Item parameter indicates the handle of the item that has been added at the last. In other words, the control automatically adds new items and creates the newly bar on the last added item, if the Chart.AllowCreateBar property is exCreateBarAuto.
- If the AllowCreateBar property is **exCreateBarManual**, you need to handle the CreateBar event to add new bars using the AddBar method. Samples, are shown bellow. If the Item parameter of the CreateBar event is negative, its absolute value indicates the number of items to be added from the last visible item, so it fits the clicked part of the chart. For instance, CreateBar(-3,Start,End) indicates that a 3 more items should be added so it covers the clicked zone.

Click here to watch a movie on how you can create bars at runtime using the

AllowCreateBar property and CreateBar event.

Syntax for CreateBar event, **/NET** version, on:

C#

```
private void CreateBar(object sender,int Item,DateTime DateStart,DateTime
DateEnd)
{
}
```

VB

```
Private Sub CreateBar(ByVal sender As System.Object,ByVal Item As Integer,ByVal
DateStart As Date,ByVal DateEnd As Date) Handles CreateBar
End Sub
```

Syntax for CreateBar event, **/COM** version, on:

C#

```
private void CreateBar(object sender,
AxEXG2ANTTLib._IG2anttEvents_CreateBarEvent e)
{
}
```

C++

```
void OnCreateBar(long Item,DATE DateStart,DATE DateEnd)
{
}
```

C++ Builder

```
void __fastcall CreateBar(TObject *Sender,Exg2anttlib_tlb::HITEM Item,DATE
DateStart,DATE DateEnd)
{
}
```

Delphi

```
procedure CreateBar(ASender: TObject; Item : HITEM;DateStart :
TDateTime;DateEnd : TDateTime);
begin
end;
```

Delphi 8 (.NET only)

```
procedure CreateBar(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_CreateBarEvent);
begin
end;
```

Powe…

```
begin event CreateBar(long Item,datetime DateStart,datetime DateEnd)
end event CreateBar
```

VB.NET

```
Private Sub CreateBar(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_CreateBarEvent) Handles CreateBar
End Sub
```

VB6

```
Private Sub CreateBar(ByVal Item As EXG2ANTTLibCtl.HITEM,ByVal DateStart As
Date,ByVal DateEnd As Date)
End Sub
```

VBA

```
Private Sub CreateBar(ByVal Item As Long,ByVal DateStart As Date,ByVal DateEnd
As Date)
End Sub
```

VFP

```
LPARAMETERS Item,DateStart,DateEnd

```

Xbas…

```
PROCEDURE OnCreateBar(oG2antt,Item,DateStart,DateEnd)
RETURN
```

Syntax for CreateBar event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="CreateBar(Item,DateStart,DateEnd)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function CreateBar(Item,DateStart,DateEnd)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComCreateBar HITEM llItem DateTime llDateStart DateTime
llDateEnd
    Forward Send OnComCreateBar llItem llDateStart llDateEnd
End_Procedure
```

Visual Objects

```
METHOD OCX_CreateBar(Item,DateStart,DateEnd) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_CreateBar(int _Item,date _DateStart,date _DateEnd)
{
}
```

XBasic

```
function CreateBar as v (Item as OLE::Exontrol.G2antt.1::HITEM,DateStart as
T,DateEnd as T)
end function
```

dBASE

```
function nativeObject_CreateBar(Item,DateStart,DateEnd)
return
```

Newer versions of the component, allows you to use the CreateBar event when the user starts creating the bar on an empty/non-items part of the chart. The Item parameter of the event may be negative if the user used an empty part to create the bar. In this case, the absolute value of the Item parameter indicates the number of items to be added to it covers the clicked area like in the following VB sample:

```
Private Sub G2antt1_CreateBar(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal DateStart As
Date, ByVal DateEnd As Date)
    With G2antt1
        .BeginUpdate
        With .Items
            If (Item < 0) Then
                Dim h As HITEM
                For i = 1 To -Item
                    h = .AddItem("")
                Next
                Item = h
            End If
            .AddBar Item, "Task", DateStart, DateEnd
        End With
        .EndUpdate
    End With
```

```
End Sub
```

The sample adds new items and a new bar when the AllowCreateBar property is exCreateBarManual. If the user clicks an empty zone (Item < 0), the sample adds a number of items as its absolute value indicates, and lastly the new bar is added to the last or to the item from the cursor.

The similar sample in VB.NET could be such as:

```
Private Sub Exg2antt1_CreateBar(ByVal sender As System.Object, ByVal Item As
System.Int32, ByVal DateStart As System.DateTime, ByVal DateEnd As System.DateTime)
Handles Exg2antt1.CreateBar
   With Exg2antt1
      .BeginUpdate()
      With .Items
         If (Item < 0) Then
            Dim i, h As Integer
            For i = 1 To -Item
               h = .AddItem("Item " & .ItemCount + 1)
            Next
            Item = h
         End If
         .set_SelectItem(Item, True)
         iBars = iBars + 1
         Dim s As String
         s = "T" & iBars
         .AddBar(Item, "Task", DateStart, DateEnd, s)
      End With
      .EndUpdate()
   End With
End Sub
```

If the AllowCreateBar property is exCreateBarAuto, the following samples change the key and the type of the bar being displayed as soon as the CreateBar event is called:

The following VB sample changes the key of the newly created bar "newbar", and the name of the bar being displayed as "Task" to "Progress":

```
Private Sub G2antt1_CreateBar(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal DateStart As
```

```
Date, ByVal DateEnd As Date)
   With G2antt1.Items
      .ItemBar(Item, "newbar", exBarName) = "Progress"
      .ItemBar(Item, "newbar", exBarKey) = DateStart
   End With
End Sub
```

The following C# sample changes the key of the newly created bar "newbar", and the name of the bar being displayed as "Task" to "Progress":

```
private void axG2antt1_CreateBar(object sender,
AxEXG2ANTTLib._IG2anttEvents_CreateBarEvent e)
{
   axG2antt1.Items.set_ItemBar(e.item, "newbar",
EXG2ANTTLib.ItemBarPropertyEnum.exBarName, "Progress");
   axG2antt1.Items.set_ItemBar(e.item, "newbar",
EXG2ANTTLib.ItemBarPropertyEnum.exBarKey, e.dateStart );
}
```

The following VB.NET sample changes the key of the newly created bar "newbar", and the name of the bar being displayed as "Task" to "Progress":

```
Private Sub AxG2antt1_CreateBar(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_CreateBarEvent) Handles AxG2antt1.CreateBar
   With AxG2antt1.Items
      .ItemBar(e.item, "newbar", EXG2ANTTLib.ItemBarPropertyEnum.exBarName) =
"Progress"
      .ItemBar(e.item, "newbar", EXG2ANTTLib.ItemBarPropertyEnum.exBarKey) =
e.dateStart
   End With
End Sub
```

The following C++ sample changes the key of the newly created bar "newbar", and the name of the bar being displayed as "Task" to "Progress":

```
void OnCreateBarG2antt1(long Item, DATE DateStart, DATE DateEnd)
{
   CItems items = m_g2antt.GetItems();
   items.SetItemBar( Item, COleVariant( _T("newbar") ), 0 /*exBarName*/, COleVariant(
```

```
_T("Progress") ) );
    items.SetItemBar( Item, COleVariant( _T("newbar") ), 9 /*exBarKey*/, COleVariant(
DateStart ) );
}
```

The following VFP sample changes the key of the newly created bar "newbar", and the name of the bar being displayed as "Task" to "Progress":

```
*** ActiveX Control Event ***
LPARAMETERS item, datestart, dateend

with thisform.G2antt1.Items
    .DefaultItem = item
    thisform.G2antt1.Template = "Items.ItemBar(0,`newbar`,0) = `Progress`"
    thisform.G2antt1.Template = "Items.ItemBar(0,`newbar`,9) = `" + dtos(datestart) + "`"
endwith
```

The Template property helps you to call any of the control's property using x-script.

If the AllowCreateBar property is exCreateBarManual, the following samples adds a new task bar, as soon as the CreateBar is called:

The following C# sample adds a new task, when the user releases the mouse:

```
private void axG2antt1_CreateBar(object sender,
AxEXG2ANTTLib._IG2anttEvents_CreateBarEvent e)
{
    Random randomKey = new Random();
    axG2antt1.BeginUpdate();
    axG2antt1.Items.AddBar(e.item, "Task", e.dateStart, e.dateEnd, randomKey.Next(), "");
    axG2antt1.EndUpdate();
}
```

The following C++ sample adds a new task, when the user releases the mouse:

```
void OnCreateBarG2antt1(long Item, DATE DateStart, DATE DateEnd)
{
    m_g2antt.BeginUpdate();
    CItems items = m_g2antt.GetItems();
    items.AddBar( Item, COleVariant( "Task" ), COleVariant( DateStart ), COleVariant( DateEnd
```

```
), COleVariant( (long)rand() ), COleVariant( "" ) );
    m_g2antt.EndUpdate();
}
```

The following VB sample adds a new task, when the user releases the mouse:

```
Private Sub G2antt1_CreateBar(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal DateStart As
Date, ByVal DateEnd As Date)
    With G2antt1
        .BeginUpdate
        With .Items
            .AddBar Item, "Task", DateStart, DateEnd, Rnd
        End With
        .EndUpdate
    End With
End Sub
```

The following VFP sample adds a new task, when the user releases the mouse:

```
*** ActiveX Control Event ***
LPARAMETERS item, datestart, dateend

with thisform.G2antt1
    .BeginUpdate
    with .Items
        .AddBar( item, "Task", datestart, dateend, RAND() )
    endwith
    .EndUpdate
endwith
```

The following VB.NET sample adds a new task, when the user releases the mouse:

```
Private Sub AxG2antt1_CreateBar(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_CreateBarEvent) Handles AxG2antt1.CreateBar
    With AxG2antt1
        .BeginUpdate()
        With .Items
            .AddBar(e.item, "Task", e.dateStart, e.dateEnd, Rnd())
        End With
```

```
        .EndUpdate()
    End With
End Sub
```

event DateChange ()

Occurs when the first visible date is changed.

Type	Description

The DateChange event is fired when the first visible date is changed. Use the FirstVisibleDate property to specify the first visible date. Use the ScrollTo method to ensure that a specified date is visible. Use the FormatDate property to format a date to a specified format.

Syntax for DateChange event, **/NET** version, on:

C#

```
private void DateChange(object sender)
{
}
```

VB

```
Private Sub DateChange(ByVal sender As System.Object) Handles DateChange
End Sub
```

Syntax for DateChange event, **/COM** version, on:

C#

```
private void DateChange(object sender, EventArgs e)
{
}
```

C++

```
void OnDateChange()
{
}
```

C++ Builder

```
void __fastcall DateChange(TObject *Sender)
{
}
```

Delphi

```
procedure DateChange(ASender: TObject; );
begin
end;
```

Delphi 8 (.NET only)

```
procedure DateChange(sender: System.Object; e: System.EventArgs);
begin
end;
```

Powe…
```
begin event DateChange()
end event DateChange
```

VB.NET
```
Private Sub DateChange(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles DateChange
End Sub
```

VB6
```
Private Sub DateChange()
End Sub
```

VBA
```
Private Sub DateChange()
End Sub
```

VFP
```
LPARAMETERS nop
```

Xbas…
```
PROCEDURE OnDateChange(oG2antt)
RETURN
```

Syntax for DateChange event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="DateChange()" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function DateChange()
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComDateChange
    Forward Send OnComDateChange
End_Procedure
```

Visual Objects
```
METHOD OCX_DateChange() CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_DateChange()
{
```

```
}
```

XBasic

```
function DateChange as v ()
end function
```

dBASE

```
function nativeObject_DateChange()
return
```

The following VB sample displays the first visible date when the user changes the first visible date:

```
Private Sub G2antt1_DateChange()
   With G2antt1.Chart
      Debug.Print FormatDateTime(.FirstVisibleDate)
   End With
End Sub
```

The following VB sample limits the scrolling area of the chart from 1/1/2005 to 31/12/2005:

```
Private Function LastVisibleDate(ByVal g As EXG2ANTTLibCtl.G2antt) As Date
   With G2antt1
      With .Chart
         Dim d As Date
         d = .FirstVisibleDate
         Do While .IsDateVisible(d)
            d = .NextDate(d, exDay, 1)
         Loop
      End With
   End With
   LastVisibleDate = d - 1
End Function

Private Sub G2antt1_DateChange()
   Dim dMin As Date, dMax As Date
   dMin = "1/1/2005"
   dMax = "31/12/2005"
   With G2antt1.Chart
      If .FirstVisibleDate < dMin Then
```

```
        .FirstVisibleDate = dMin
    End If
    If LastVisibleDate(G2antt1) > dMax Then
        .FirstVisibleDate = dMax - (LastVisibleDate(G2antt1) - .FirstVisibleDate) + 1
    End If
  End With
End Sub
```

or you can use the FormatDate method like follows:

```
Private Sub G2antt1_DateChange()
  With G2antt1.Chart
    Debug.Print .FormatDate(.FirstVisibleDate, "<%yyyy%>-<%m%>-<%d%>")
  End With
End Sub
```

The following C++ sample displays the first visible date when the user changes the first visible date:

```
#include "G2antt.h"
#include "Chart.h"

static DATE V2D( VARIANT* pvtDate )
{
  COleVariant vtDate;
  vtDate.ChangeType( VT_DATE, pvtDate );
  return V_DATE( &vtDate );
}

void OnDateChangeG2antt1()
{
  if ( m_g2antt.GetControlUnknown() )
  {
    CChart chart = m_g2antt.GetChart();
    TCHAR szDate[1024] = _T("");
    SYSTEMTIME stDate = {0};
    VariantTimeToSystemTime( V2D( &chart.GetFirstVisibleDate() ), &stDate );
    GetDateFormat( LOCALE_SYSTEM_DEFAULT, LOCALE_USE_CP_ACP, &stDate, NULL,
```

```
szDate, 1024 );
        OutputDebugString( szDate );
    }
}
```

The following VB.NET sample displays the first visible date when the user changes the first visible date:

```
Private Sub AxG2antt1_DateChange(ByVal sender As Object, ByVal e As System.EventArgs)
Handles AxG2antt1.DateChange
    Debug.Write(AxG2antt1.Chart.FirstVisibleDate.ToString())
End Sub
```

The following C# sample displays the first visible date when the user changes the first visible date:

```
private void axG2antt1_DateChange(object sender, EventArgs e)
{
    System.Diagnostics.Debug.Write(axG2antt1.Chart.FirstVisibleDate.ToString());
}
```

The following VFP sample displays the first visible date when the user changes the first visible date:

```
*** ActiveX Control Event ***

with thisform.G2antt1.Chart
    wait window nowait .FormatDate(.FirstVisibleDate, "<%yyyy%>-<%m%>-<%d%>")
endwith
```

event DateTimeChanged (DateTime as Date)

Notifies your application that the current time is changed.

Type	Description
DateTime as Date	A Date-Time expression that indicates the new current time.

The DateTimeChanged event notifies your application when the current date-time is changed. The DateTimeChanged event is fired ONLY if the MarkNowColor property is not zero (0). Use the FirstVisibleDate property to specify the first visible Date-Time in the control's chart. The MarkNowUnit property specifies the unit of time to count for. For instance, you can show the current date-time from current second, to next second, from minute to next minute, and so on. Use the MarkNowCount property to specify the number of units of date-time to count from. For instance, you can show the current date-time from 5 seconds to 5 seconds, and so on. The MarkNowWidth property specifies the width in pixels of the vertical bar that shows the current date-time. The MarkNowTransparent property specifies the percent of transparency to show the vertical bar that indicates the current date-time. The MarkNow/MarkNowDelay property can be used to specify the current date-time or your custom date time.

Syntax for DateTimeChanged event, **/NET** version, on:

C#

```
private void DateTimeChanged(object sender,DateTime DateTime)
{
}
```

VB

```
Private Sub DateTimeChanged(ByVal sender As System.Object,ByVal DateTime As Date) Handles DateTimeChanged
End Sub
```

Syntax for DateTimeChanged event, **/COM** version, on:

C#

```
private void DateTimeChanged(object sender,
AxEXG2ANTTLib._IG2anttEvents_DateTimeChangedEvent e)
{
}
```

C++

```
void OnDateTimeChanged(DATE DateTime)
{
}
```

C++ Builder

```
void __fastcall DateTimeChanged(TObject *Sender,DATE DateTime)
{
}
```

Delphi

```
procedure DateTimeChanged(ASender: TObject; DateTime : TDateTime);
begin
end;
```

Delphi 8 (.NET only)

```
procedure DateTimeChanged(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_DateTimeChangedEvent);
begin
end;
```

Powe…

```
begin event DateTimeChanged(datetime DateTime)
end event DateTimeChanged
```

VB.NET

```
Private Sub DateTimeChanged(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_DateTimeChangedEvent) Handles
DateTimeChanged
End Sub
```

VB6

```
Private Sub DateTimeChanged(ByVal DateTime As Date)
End Sub
```

VBA

```
Private Sub DateTimeChanged(ByVal DateTime As Date)
End Sub
```

VFP

```
LPARAMETERS DateTime

```

Xbas…

```
PROCEDURE OnDateTimeChanged(oG2antt,DateTime)
RETURN
```

Syntax for DateTimeChanged event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="DateTimeChanged(DateTime)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function DateTimeChanged(DateTime)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComDateTimeChanged DateTime llDateTime
    Forward Send OnComDateTimeChanged llDateTime
End_Procedure
```

Visual Objects

```
METHOD OCX_DateTimeChanged(DateTime) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_DateTimeChanged(date _DateTime)
{
}
```

XBasic

```
function DateTimeChanged as v (DateTime as T)
end function
```

dBASE

```
function nativeObject_DateTimeChanged(DateTime)
return
```

event DblClick (Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occurs when the user dblclk the left mouse button over an object.

Type	Description
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in container coordinates

The DblClick event is fired when user double clicks the control. Use the ItemFromPoint method to determine the cell over the cursor. Use the ExpandOnDblClk property to specify whether an item is expanded or collapsed when user double clicks it. Use the ColumnFromPoint property to retrieve the column from cursor. Use the DateFromPoint property to specify the date from the cursor. Use the BarFromPoint property to get the bar from the point. Use the LinkFromPoint property to get the link from the point. *Almost all properties that get an object from point supports -1,-1 coordinate that specifies the current cursor position, so no conversion is required for X and Y coordinates.*

Syntax for DblClick event, **/NET** version, on:

C#
```
private void DblClick(object sender,short Shift,int X,int Y)
{
}
```

VB
```
Private Sub DblClick(ByVal sender As System.Object,ByVal Shift As Short,ByVal X
As Integer,ByVal Y As Integer) Handles DblClick
End Sub
```

Syntax for DblClick event, **/COM** version, on:

C#
```
private void DblClick(object sender,
AxEXG2ANTTLib._IG2anttEvents_DblClickEvent e)
{
}
```

C++

```
void OnDblClick(short Shift,long X,long Y)
{
}
```

C++ Builder

```
void __fastcall DblClick(TObject *Sender,short Shift,int X,int Y)
{
}
```

Delphi

```
procedure DblClick(ASender: TObject; Shift : Smallint;X : Integer;Y : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure DblClick(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_DblClickEvent);
begin
end;
```

Powe…

```
begin event DblClick(integer Shift,long X,long Y)
end event DblClick
```

VB.NET

```
Private Sub DblClick(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_DblClickEvent) Handles DblClick
End Sub
```

VB6

```
Private Sub DblClick(Shift As Integer,X As Single,Y As Single)
End Sub
```

VBA

```
Private Sub DblClick(ByVal Shift As Integer,ByVal X As Long,ByVal Y As Long)
End Sub
```

VFP

```
LPARAMETERS Shift,X,Y
```

Xbas…

```
PROCEDURE OnDblClick(oG2antt,Shift,X,Y)
RETURN
```

Syntax for DblClick event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="DblClick(Shift,X,Y)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function DblClick(Shift,X,Y)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComDblClick Short llShift OLE_XPOS_PIXELS llX OLE_YPOS_PIXELS
llY
    Forward Send OnComDblClick llShift llX llY
End_Procedure
```

Visual Objects

```
METHOD OCX_DblClick(Shift,X,Y) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_DblClick(int _Shift,int _X,int _Y)
{
}
```

XBasic

```
function DblClick as v (Shift as N,X as OLE::Exontrol.G2antt.1::OLE_XPOS_PIXELS,Y
as OLE::Exontrol.G2antt.1::OLE_YPOS_PIXELS)
end function
```

dBASE

```
function nativeObject_DblClick(Shift,X,Y)
return
```

The following Access sample prints a message when an item has been double clicked:

```
Private Sub G2antt1_DblClick(ByVal Shift As Integer, ByVal X As Long, ByVal Y As Long)
    Dim h As HITEM
    Dim c As Long, hit As Long
    h = G2antt1.ItemFromPoint(-1, -1, c, hit)
    If Not (h = 0) Then
        MsgBox "The """ & G2antt1.Items.CellValue(h, c) & """ cell has been double clicked."
    End If
```

```
End Sub
```

The following VB sample prints a message when an item has been double clicked:

```
Private Sub G2antt1_DblClick(Shift As Integer, X As Single, Y As Single)
   ' Converts the container coordinates to client coordinates
   X = X / Screen.TwipsPerPixelX
   Y = Y / Screen.TwipsPerPixelY
   Dim h As HITEM
   Dim c As Long, hit as Long
   ' Gets the item from (X,Y)
   h = G2antt1.ItemFromPoint(X, Y, c, hit)
   If Not (h = 0) Then
      MsgBox "The " & h & " item has been double clicked."
   End If
End Sub
```

The following VB sample displays a message when a cell has been double clicked:

```
Private Sub G2antt1_DblClick(Shift As Integer, X As Single, Y As Single)
   ' Converts the container coordinates to client coordinates
   X = X / Screen.TwipsPerPixelX
   Y = Y / Screen.TwipsPerPixelY
   Dim h As HITEM
   Dim c As Long, hit as Long
   ' Gets the item from (X,Y)
   h = G2antt1.ItemFromPoint(X, Y, c, hit)
   If Not (h = 0) Then
      MsgBox "The """ & G2antt1.Items.CellValue(h, c) & """ cell has been double clicked."
   End If
End Sub
```

The following C++ sample displays the caption of the cell being double clicked (including the inner cells):

```
#include "Items.h"

static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
```

```
    if ( pv )
    {
        if ( pv->vt == VT_ERROR )
            return szDefault;

        COleVariant vt;
        vt.ChangeType( VT_BSTR, pv );
        return V_BSTR( &vt );
    }
    return szDefault;
}

void OnDblClickG2antt1(short Shift, long X, long Y)
{
    long c = NULL, hit = NULL;
    long h = m_g2antt.GetItemFromPoint( X, Y, &c, &hit );
    if ( ( h != 0 ) || ( c != 0 ) )
    {
        COleVariant vtItem( h ), vtColumn( c );
        CString strCaption = V2S( &m_g2antt.GetItems().GetCellValue( vtItem, vtColumn ) );
        MessageBox( strCaption );
    }
}
```

The following VB.NET sample displays the caption of the cell being double clicked (including the inner cells):

```
Private Sub AxG2antt1_DblClick(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_DblClickEvent) Handles AxG2antt1.DblClick
    Dim h As Integer, c As Integer, hit As EXG2ANTTLib.HitTestInfoEnum
    With AxG2antt1
        h = .get_ItemFromPoint(e.x, e.y, c, hit)
        If Not (h = 0) Or Not (c = 0) Then
            MessageBox.Show(.Items.CellValue(h, c))
        End If
    End With
End Sub
```

The following C# sample displays the caption of the cell being double clicked (including the inner cells):

```
private void axG2antt1_DblClick(object sender,
AxEXG2ANTTLib._IG2anttEvents_DblClickEvent e)
{
   EXG2ANTTLib.HitTestInfoEnum hit;
   int c = 0, h = axG2antt1.get_ItemFromPoint( e.x, e.y, out c, out hit );
   if ( ( h != 0 ) || ( c != 0 ) )
      MessageBox.Show( axG2antt1.Items.get_CellValue( h, c ).ToString() );
}
```

The following VFP sample displays the caption of the cell being double clicked:

```
*** ActiveX Control Event ***
LPARAMETERS shift, x, y

local c, hit
c = 0
hit = 0

with thisform.G2antt1
   .Items.DefaultItem = .ItemFromPoint( x, y, @c, @hit )
   if ( .Items.DefaultItem != 0 )
      wait window nowait .Items.CellValue( 0, c )
   endif
endwith
```

event Edit (Item as HITEM, ColIndex as Long, Cancel as Boolean)

Occurs just before editing the focused cell.

Type	Description
Item as HITEM	A long expression that determines the item's handle. If the Item parameter is 0, and the ColIndex property is different than zero, the ColIndex indicates the handle of the cell where the state is changed.
ColIndex as Long	A long expression that indicates the column's index, if the Item parameter is not zero, a long expression that indicates the handle of the cell if the Item parameter is 0.
Cancel as Boolean	A boolean expression that indicates whether the editing operation is canceled.

The Edit event is fired when the edit operation is about to begin. Use the Edit event to disable editing specific cells. The Edit event is not fired if the user changes programmatically the CellValue property. Use the EditOpen event to notify your application that editing the cell started. Use the EditClose event to notify your application that editing the cell ended. Use the Change event to notify your application that user changes the cell's value. Use the Edit method to edit a cell by code. Use the CellEditor or Editor property to assign an editor to a cell or to a column.

The edit events are fired in the following order:

1. Edit event. Prevents editing cells, before showing the cell's editor.
2. EditOpen event. The edit operation started, the cell's editor is shown. The Editing property gives the window's handle of the built-in editor being started.
3. Change event. The Change event is fired only if the user types ENTER key, or the user selects a new value from a predefined data list.
4. EditClose event. The cell's editor is hidden and closed.

Syntax for Edit event, **/NET** version, on:

C#

```
private void EditEvent(object sender,int Item,int ColIndex,ref bool Cancel)
{
}
```

VB

```
Private Sub EditEvent(ByVal sender As System.Object,ByVal Item As Integer,ByVal
ColIndex As Integer,ByRef Cancel As Boolean) Handles EditEvent
```

```
End Sub
```

Syntax for Edit event, **/COM** version, on:

C#

```
private void EditEvent(object sender, AxEXG2ANTTLib._IG2anttEvents_EditEvent e)
{
}
```

C++

```
void OnEdit(long Item,long ColIndex,BOOL FAR* Cancel)
{
}
```

C++ Builder

```
void __fastcall Edit(TObject *Sender,Exg2anttlib_tlb::HITEM Item,long
ColIndex,VARIANT_BOOL * Cancel)
{
}
```

Delphi

```
procedure Edit(ASender: TObject; Item : HITEM;ColIndex : Integer;var Cancel :
WordBool);
begin
end;
```

Delphi 8 (.NET only)

```
procedure EditEvent(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_EditEvent);
begin
end;
```

Powe…

```
begin event Edit(long Item,long ColIndex,boolean Cancel)
end event Edit
```

VB.NET

```
Private Sub EditEvent(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_EditEvent) Handles EditEvent
End Sub
```

VB6

```
Private Sub Edit(ByVal Item As EXG2ANTTLibCtl.HITEM,ByVal ColIndex As
Long,Cancel As Boolean)
End Sub
```

VBA

```
Private Sub Edit(ByVal Item As Long,ByVal ColIndex As Long,Cancel As Boolean)
End Sub
```

VFP

```
LPARAMETERS Item,ColIndex,Cancel

```

Xbas…

```
PROCEDURE OnEdit(oG2antt,Item,ColIndex,Cancel)
RETURN
```

Syntax for Edit event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="Edit(Item,ColIndex,Cancel)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function Edit(Item,ColIndex,Cancel)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComEdit HITEM llItem Integer llColIndex Boolean llCancel
    Forward Send OnComEdit llItem llColIndex llCancel
End_Procedure
```

Visual Objects

```
METHOD OCX_Edit(Item,ColIndex,Cancel) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_Edit(int _Item,int _ColIndex,COMVariant /*bool*/ _Cancel)
{
}
```

XBasic

```
function Edit as v (Item as OLE::Exontrol.G2antt.1::HITEM,ColIndex as N,Cancel as L)
end function
```

dBASE

```
function nativeObject_Edit(Item,ColIndex,Cancel)
return
```

The following VB sample disables editing cells in the first column:

```
Private Sub G2antt1_Edit(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal ColIndex As Long, Cancel As Boolean)
   ' Cancels editing first column
   Cancel = IIf(ColIndex = 0, True, False)
End Sub
```

The following VB sample changes the cell's value to a default value, if the user enters an empty value:

```
Private Sub G2antt1_Edit(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal ColIndex As Long, Cancel As Boolean)
   ' Sets the 'default' value for empty cell
   With G2antt1.Items
      If (Len(.CellValue(Item, ColIndex)) = 0) Then
         .CellValue(Item, ColIndex) = "default"
      End If
   End With
End Sub
```

The following C++ sample disables editing cells in the first column:

```
void OnEditG2antt1(long Item, long ColIndex, BOOL FAR* Cancel)
{
   if ( ColIndex == 0 )
      *Cancel = TRUE;
}
```

The following VB.NET sample disables editing cells in the first column:

```
Private Sub AxG2antt1_EditEvent(ByVal sender As Object, ByVal e As AxEXG2ANTTLib._IG2anttEvents_EditEvent) Handles AxG2antt1.EditEvent
   If (e.colIndex = 0) Then
      e.cancel = True
   End If
End Sub
```

The following C# sample disables editing cells in the first column:

```
private void axG2antt1_EditEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_EditEvent e)
{
    if (e.colIndex == 0)
        e.cancel = true;
}
```

The following VFP sample disables editing cells in the first column:

```
*** ActiveX Control Event ***
LPARAMETERS item, colindex, cancel

if ( colindex = 0 )
    cancel = .t.
endif
```

event EditClose ()

Occurs when the edit operation ends.

Type	Description

Use the EditClose event to notify your application that the editor is closed. The EditClose event is fired when the focused cell ends editing. Use the FocusItem property to determine the handle of the item where the edit operation ends. Use the FocusColumnIndex property to determine the index of the column where the edit operation ends. The Editing specifies the window's handle of the built-in editor while the control is running in edit mode. The EditingText property returns the caption being shown on the editor while the control runs in edit mode. Use the EditClose method to closes the current editor, by code. For instance, the EditClose event is not fired when user hides the drop down portion of the editor. Use the Edit event to prevent editing a cell.

The edit events are fired in the following order:

1. Edit event. Prevents editing cells, before showing the cell's editor.
2. EditOpen event. The edit operation started, the cell's editor is shown. The Editing property gives the window's handle of the built-in editor being started.
3. Change event. The Change event is fired only if the user types ENTER key, or the user selects a new value from a predefined data list.
4. EditClose event. The cell's editor is hidden and closed.

Syntax for EditClose event, **/NET** version, on:

C#
```
private void EditCloseEvent(object sender)
{
}
```

VB
```
Private Sub EditCloseEvent(ByVal sender As System.Object) Handles
EditCloseEvent
End Sub
```

Syntax for EditClose event, **/COM** version, on:

C#
```
private void EditCloseEvent(object sender, EventArgs e)
{
}
```

C++
```
void OnEditClose()
{
}
```

C++ Builder
```
void __fastcall EditClose(TObject *Sender)
{
}
```

Delphi
```
procedure EditClose(ASender: TObject; );
begin
end;
```

Delphi 8 (.NET only)
```
procedure EditCloseEvent(sender: System.Object; e: System.EventArgs);
begin
end;
```

Powe…
```
begin event EditClose()
end event EditClose
```

VB.NET
```
Private Sub EditCloseEvent(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles EditCloseEvent
End Sub
```

VB6
```
Private Sub EditClose()
End Sub
```

VBA
```
Private Sub EditClose()
End Sub
```

VFP
```
LPARAMETERS nop
```

Xbas…
```
PROCEDURE OnEditClose(oG2antt)
RETURN
```

Syntax for EditClose event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="EditClose()" LANGUAGE="JScript">
```

```
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function EditClose()
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComEditClose
    Forward Send OnComEditClose
End_Procedure
```

Visual Objects

```
METHOD OCX_EditClose() CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_EditClose()
{
}
```

XBasic

```
function EditClose as v ()
end function
```

dBASE

```
function nativeObject_EditClose()
return
```

The following VB sample displays the window's handle of the built-in editor being closed:

```
Private Sub G2antt1_EditClose()
    Debug.Print "EditClose " & G2antt1.Editing
End Sub
```

The following VB sample displays the caption of the cell where the edit operation ends:

```
Private Sub G2antt1_EditClose()
    With G2antt1.Items
        Debug.Print "EditClose on '"; .CellCaption(.FocusItem, G2antt1.FocusColumnIndex) &
"'."
    End With
End Sub
```

The following C++ sample displays the handle of the built-in editor being closed:

```
#include "Items.h"
void OnEditCloseG2antt1()
{
   CItems items = m_g2antt.GetItems();
   COleVariant vtItem( items.GetFocusItem() ), vtColumn(
m_g2antt.GetFocusColumnIndex() );
   CString strFormat;
   strFormat.Format( "'%s'  %i", V2S( &items.GetCellValue( vtItem, vtColumn ) ),
m_g2antt.GetEditing() );
   OutputDebugString( strFormat );
}
```

The following VB.NET sample displays the handle of the built-in editor being closed:

```
Private Sub AxG2antt1_EditCloseEvent(ByVal sender As Object, ByVal e As
System.EventArgs) Handles AxG2antt1.EditCloseEvent
   With AxG2antt1
      Debug.Print(.Items.CellValue(.Items.FocusItem, .FocusColumnIndex) & " " &
.Editing.ToString())
   End With
End Sub
```

The following C# sample displays the handle of the built-in editor being closed:

```
private void axG2antt1_EditCloseEvent(object sender, EventArgs e)
{
   object cellValue = axG2antt1.Items.get_CellValue(axG2antt1.Items.FocusItem,
axG2antt1.FocusColumnIndex);
   string strOutput = "'" + (cellValue != null ? cellValue.ToString() : "") + "' " +
axG2antt1.Editing.ToString();
   System.Diagnostics.Debug.WriteLine( strOutput );
}
```

The following VFP sample displays the handle of the built-in editor being closed:

```
*** ActiveX Control Event ***
```

```
with thisform.G2antt1.Items
    .DefaultItem = .FocusItem()
    wait window nowait str(.CellValue( 0, thisform.G2antt1.FocusColumnIndex() ))
    wait window nowait str(thisform.G2antt1.Editing())
endwith
```

event EditOpen ()

Occurs when the edit operation starts.

Type	Description

Use the EditOpen event to notify your application that the cell's editor is shown and ready to edit the cell. The Editing specifies the window's handle of the built-in editor while the control is running in edit mode. The EditingText property returns the caption being shown on the editor while the control runs in edit mode.

The edit events are fired in the following order:

1. Edit event. Prevents editing cells, before showing the cell's editor.
2. EditOpen event. The edit operation started, the cell's editor is shown. The Editing property gives the window's handle of the built-in editor being started.
3. Change event. The Change event is fired only if the user types ENTER key, or the user selects a new value from a predefined data list.
4. EditClose event. The cell's editor is hidden and closed.

Syntax for EditOpen event, **/NET** version, on:

C#
```
private void EditOpen(object sender)
{
}
```

VB
```
Private Sub EditOpen(ByVal sender As System.Object) Handles EditOpen
End Sub
```

Syntax for EditOpen event, **/COM** version, on:

C#
```
private void EditOpen(object sender, EventArgs e)
{
}
```

C++
```
void OnEditOpen()
{
}
```

C++ Builder

```
void __fastcall EditOpen(TObject *Sender)
{
}
```

Delphi

```
procedure EditOpen(ASender: TObject; );
begin
end;
```

Delphi 8 (.NET only)

```
procedure EditOpen(sender: System.Object; e: System.EventArgs);
begin
end;
```

Powe…

```
begin event EditOpen()
end event EditOpen
```

VB.NET

```
Private Sub EditOpen(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles EditOpen
End Sub
```

VB6

```
Private Sub EditOpen()
End Sub
```

VBA

```
Private Sub EditOpen()
End Sub
```

VFP

```
LPARAMETERS nop
```

Xbas…

```
PROCEDURE OnEditOpen(oG2antt)
RETURN
```

Syntax for EditOpen event, ***/COM*** version (others), on:

Java…

```
<SCRIPT EVENT="EditOpen()" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function EditOpen()
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComEditOpen
    Forward Send OnComEditOpen
End_Procedure
```

Visual Objects

```
METHOD OCX_EditOpen() CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_EditOpen()
{
}
```

XBasic

```
function EditOpen as v ()
end function
```

dBASE

```
function nativeObject_EditOpen()
return
```

event Error (Error as Long, Description as String)

Fired when an internal error occurs.

Type	Description
Error as Long	A long expression that indicates the error number.
Description as String	A string expression that describes the error.

The Error event is fired each time when an internal error occurs. The Error event is usually fired when the control is bounded to an ADO Recordset. For instance, if the user changes a field, the control tries to update the current record. If it fails, the Error event is fired. Use the DataSource property to bind the control to a database.

Syntax for Error event, **/NET** version, on:

C#
```
private void Error(object sender,int Err,string Description)
{
}
```

VB
```
Private Sub Error(ByVal sender As System.Object,ByVal Err As Integer,ByVal
Description As String) Handles Error
End Sub
```

Syntax for Error event, **/COM** version, on:

C#
```
private void Error(object sender, AxEXG2ANTTLib._IG2anttEvents_ErrorEvent e)
{
}
```

C++
```
void OnError(long Error,LPCTSTR Description)
{
}
```

C++ Builder
```
void __fastcall Error(TObject *Sender,long Error,BSTR Description)
{
}
```

Delphi
```
procedure Error(ASender: TObject; Error : Integer;Description : WideString);
begin
end;
```

Delphi 8 (.NET only)

```
procedure Error(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_ErrorEvent);
begin
end;
```

Powe…

```
begin event Error(long Error,string Description)
end event Error
```

VB.NET

```
Private Sub Error(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_ErrorEvent) Handles Error
End Sub
```

VB6

```
Private Sub Error(ByVal Error As Long,ByVal Description As String)
End Sub
```

VBA

```
Private Sub Error(ByVal Error As Long,ByVal Description As String)
End Sub
```

VFP

```
LPARAMETERS Error,Description

```

Xbas…

```
PROCEDURE OnError(oG2antt,Error,Description)
RETURN
```

Syntax for Error event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="Error(Error,Description)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function Error(Error,Description)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComError Integer llError String llDescription
    Forward Send OnComError llError llDescription
End_Procedure
```

Visual Objects

```
METHOD OCX_Error(Error,Description) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_Error(int _Error,str _Description)
{
}
```

XBasic

```
function Error as v (Error as N,Description as C)
end function
```

dBASE

```
function nativeObject_Error(Error,Description)
return
```

event Event (EventID as Long)

Notifies the application once the control fires an event.

Type	Description
EventID as Long	A Long expression that specifies the identifier of the event. Each internal event of the control has an unique identifier. Use the EventParam(-2) to display entire information about fired event (such as name, identifier, and properties). The EventParam(-1) retrieves the number of parameters of fired event

The Event notification occurs ANY time the control fires an event. For instance if a BarResize event occurs, then a Event(120) occurs also, so the events inside the Event are differentiated by its EventID. Print the EventParam(-2) during the Event notification, and you get debugging information for the name, ID, and parameters of the fired event. So, back to the BarResize which is defined as event BarResize (Item as HITEM, Key as Variant), it means it has 2 parameters, Item and Key, so the EventParam(0) gets the Item parameter, while the EventParam(1) gets the Key of the bar being resized, when the EventID is 120, where 120 indicates the identifier of the BarResize event. The number of parameters different from event to event. For instance, Click event has no parameter, which means that the EventParam(-1) gets 0, and for BarResize gets the 2.

Click here to watch a movie on how you can use the eXHelper to get information about the fired events using the Event handler. The Event notification is sent any time the control fires a specified event. For instance, if the BarResize event occurs, the order of the events are Event(120) and next BarResize. You can use any of these notifications based on your requirements or limitations of the programming environment you are using.

This is useful for X++, which does not support event with parameters passed by reference. Also, this could be useful for C++ Builder or Delphi, which does not handle properly the events with parameters of VARIANT type.

In X++ the "Error executing code: FormActiveXControl (data source), method ... called with invalid parameters" occurs when handling events that have parameters passed by reference. Passed by reference, means that in the event handler, you can change the value for that parameter, and so the control will takes the new value, and use it. The X++ is NOT able to handle properly events with parameters by reference, so we have the solution.

The solution is using and handling the Event notification and EventParam method., instead handling the event that gives the "invalid parameters" error executing code.

Let's assume that we need to handle the BarParentChange event to change the _Cancel parameter from false to true, which fires the "Error executing code: FormActiveXControl

(data source), method onEvent_BarParentChange called with invalid parameters." We need to know the identifier of the BarParentChange event (each event has an unique identifier and it is static, defined in the control's type library). If you are not familiar with what a type library means just handle the Event of the control as follows:

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
   print exg2antt1.EventParam(-2).toString();
}
```

This code allows you to display the information for each event of the control being fired as in the list bellow:

```
"MouseMove/-606( 1 , 0 , 145 , 36 )" VT_BSTR
"BarParentChange/125( 192998632 , 'B' , 192999592 , =false )" VT_BSTR
"BeforeDrawPart/54( 2 , -1962866148 , =0 , =0 , =0 , =0 , =false )" VT_BSTR
"AfterDrawPart/55( 2 , -1962866148 , 0 , 0 , 0 , 0 )" VT_BSTR
"MouseMove/-606( 1 , 0 , 145 , 35 )" VT_BSTR
```

Each line indicates an event, and the following information is provided: the name of the event, its identifier, and the list of parameters being passed to the event. The parameters that starts with = character, indicates a parameter by reference, in other words one that can changed during the event handler.

Now, we can see that the identifier for the BarParentChange event is 125, so we need to handle the Event event as:

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
   ;
   if ( _EventID == 125 ) /*event BarParentChange (Item as HITEM, Key as Variant, NewItem as HITEM, Cancel as Boolean) */
      exg2antt1.EventParam( 3 /*Cancel*/, COMVariant::createFromBoolean(true) );
}
```

The code checks if the BarParentChange (_EventID == 125) event is fired, and changes the third parameter of the event to true. The definition for BarParentChange event can be consulted in the control's documentation or in the ActiveX explorer. So, anytime you need to access the original parameters for the event you should use the EventParam method that

allows you to get or set a parameter. If the parameter is not passed by reference, you can not change the parameter's value.

Now, let's add some code to see a complex sample, so let's say that we need to prevent moving the bar from an item to any disabled item. So, we need to specify the Cancel parameter as not Items.EnableItem(NewItem), in other words cancels if the new parent is disabled. Shortly the code will be:

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
   ;
   if ( _EventID == 125 ) /*event BarParentChange (Item as HITEM, Key as Variant, NewItem as HITEM, Cancel as Boolean) */
      if ( !exg2antt1.Items().EnableItem( exg2antt1.EventParam( 2 /*NewItem*/ ) ) )
         exg2antt1.EventParam( 3 /*Cancel*/, COMVariant::createFromBoolean(true) );
}
```

In conclusion, anytime the X++ fires the "invalid parameters." while handling an event, you can use and handle the Event notification and EventParam methods of the control

Syntax for Event event, **/NET** version, on:

C#

```
private void Event(object sender,int EventID)
{
}
```

VB

```
Private Sub Event(ByVal sender As System.Object,ByVal EventID As Integer)
Handles Event
End Sub
```

Syntax for Event event, **/COM** version, on:

C#

```
private void Event(object sender, AxEXG2ANTTLib._IG2anttEvents_EventEvent e)
{
}
```

C++

```
void OnEvent(long EventID)
{
}
```

C++ Builder
```
void __fastcall Event(TObject *Sender,long EventID)
{
}
```

Delphi
```
procedure Event(ASender: TObject; EventID : Integer);
begin
end;
```

Delphi 8 (.NET only)
```
procedure Event(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_EventEvent);
begin
end;
```

Powe…
```
begin event Event(long EventID)
end event Event
```

VB.NET
```
Private Sub Event(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_EventEvent) Handles Event
End Sub
```

VB6
```
Private Sub Event(ByVal EventID As Long)
End Sub
```

VBA
```
Private Sub Event(ByVal EventID As Long)
End Sub
```

VFP
```
LPARAMETERS EventID
```

Xbas…
```
PROCEDURE OnEvent(oG2antt,EventID)
RETURN
```

Syntax for Event event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="Event(EventID)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
```

```
Function Event(EventID)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComEvent Integer llEventID
    Forward Send OnComEvent llEventID
End_Procedure
```

Visual Objects

```
METHOD OCX_Event(EventID) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_Event(int _EventID)
{
}
```

XBasic

```
function Event as v (EventID as N)
end function
```

dBASE

```
function nativeObject_Event(EventID)
return
```

event FilterChange ()

Occurs when filter was changed.

Type	Description

Use the FilterChange event to notify your application that the control's filter is changed. The FilterChanging event occurs just before applying the filter. Use the Filter and FilterType properties to retrieve the column's filter string, if case, and the column's filter type. The ApplyFilter and ClearFilter methods fire the FilterChange event. Use the DisplayFilterButton property to add a filter bar button to the column's caption. Use the FilterBarHeight property to specify the height of the control's filter bar. Use the FilterBarFont property to specify the font for the control's filter bar.

Syntax for FilterChange event, **/NET** version, on:

C#

```
private void FilterChange(object sender)
{
}
```

VB

```
Private Sub FilterChange(ByVal sender As System.Object) Handles FilterChange
End Sub
```

Syntax for FilterChange event, **/COM** version, on:

C#

```
private void FilterChange(object sender, EventArgs e)
{
}
```

C++

```
void OnFilterChange()
{
}
```

C++ Builder

```
void __fastcall FilterChange(TObject *Sender)
{
}
```

Delphi

```
procedure FilterChange(ASender: TObject; );
begin
end;
```

Delphi 8 (.NET only)

```
procedure FilterChange(sender: System.Object; e: System.EventArgs);
begin
end;
```

Powe…

```
begin event FilterChange()
end event FilterChange
```

VB.NET

```
Private Sub FilterChange(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles FilterChange
End Sub
```

VB6

```
Private Sub FilterChange()
End Sub
```

VBA

```
Private Sub FilterChange()
End Sub
```

VFP

```
LPARAMETERS nop
```

Xbas…

```
PROCEDURE OnFilterChange(oG2antt)
RETURN
```

Syntax for FilterChange event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="FilterChange()" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function FilterChange()
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComFilterChange
    Forward Send OnComFilterChange
```

```
End_Procedure
```

Visual Objects

```
METHOD OCX_FilterChange() CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_FilterChange()
{
}
```

XBasic

```
function FilterChange as v ()
end function
```

dBASE

```
function nativeObject_FilterChange()
return
```

event FilterChanging ()

Notifies your application that the filter is about to change.

Type	Description

The FilterChanging event occurs just before applying the filter. The FilterChange event occurs once the filter is applied, so the list gets filtered. Use the Filter and FilterType properties to retrieve the column's filter string, if case, and the column's filter type. The ApplyFilter and ClearFilter methods fire the FilterChange event. Use the DisplayFilterButton property to add a filter bar button to the column's caption. Use the FilterBarHeight property to specify the height of the control's filter bar. Use the FilterBarFont property to specify the font for the control's filter bar. For instance, you can use the FilterChanging event to start a timer, and count the time to get the filter applied, when the FilterChange event is fired.

Syntax for FilterChanging event, **/NET** version, on:

C#
```
private void FilterChanging(object sender)
{
}
```

VB
```
Private Sub FilterChanging(ByVal sender As System.Object) Handles
FilterChanging
End Sub
```

Syntax for FilterChanging event, **/COM** version, on:

C#
```
private void FilterChanging(object sender, EventArgs e)
{
}
```

C++
```
void OnFilterChanging()
{
}
```

C++ Builder
```
void __fastcall FilterChanging(TObject *Sender)
{
}
```

Delphi
```
procedure FilterChanging(ASender: TObject; );
begin
```

```
end;
```

Delphi 8 (.NET only)

```
procedure FilterChanging(sender: System.Object; e: System.EventArgs);
begin
end;
```

Powe…

```
begin event FilterChanging()
end event FilterChanging
```

VB.NET

```
Private Sub FilterChanging(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles FilterChanging
End Sub
```

VB6

```
Private Sub FilterChanging()
End Sub
```

VBA

```
Private Sub FilterChanging()
End Sub
```

VFP

```
LPARAMETERS nop
```

Xbas…

```
PROCEDURE OnFilterChanging(oG2antt)
RETURN
```

Syntax for FilterChanging event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="FilterChanging()" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function FilterChanging()
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComFilterChanging
    Forward Send OnComFilterChanging
End_Procedure
```

Visual Objects

```
METHOD OCX_FilterChanging() CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_FilterChanging()
{
}
```

XBasic

```
function FilterChanging as v ()
end function
```

dBASE

```
function nativeObject_FilterChanging()
return
```

event FocusChanged ()

Occurs when a new cell is focused.

Type	Description

The FocusChanged event occurs when a new cell is focused. The SelectionChanged event notifies your application once a new item is selected/unselected.

Syntax for FocusChanged event, **/NET** version, on:

C#
```
private void FocusChanged(object sender)
{
}
```

VB
```
Private Sub FocusChanged(ByVal sender As System.Object) Handles
FocusChanged
End Sub
```

Syntax for FocusChanged event, **/COM** version, on:

C#
```
private void FocusChanged(object sender, EventArgs e)
{
}
```

C++
```
void OnFocusChanged()
{
}
```

C++ Builder
```
void __fastcall FocusChanged(TObject *Sender)
{
}
```

Delphi
```
procedure FocusChanged(ASender: TObject; );
begin
end;
```

Delphi 8 (.NET only)
```
procedure FocusChanged(sender: System.Object; e: System.EventArgs);
begin
end;
```

Powe…
```
begin event FocusChanged()
end event FocusChanged
```

VB.NET
```
Private Sub FocusChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles FocusChanged
End Sub
```

VB6
```
Private Sub FocusChanged()
End Sub
```

VBA
```
Private Sub FocusChanged()
End Sub
```

VFP
```
LPARAMETERS nop
```

Xbas…
```
PROCEDURE OnFocusChanged(oG2antt)
RETURN
```

Syntax for FocusChanged event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="FocusChanged()" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function FocusChanged()
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComFocusChanged
    Forward Send OnComFocusChanged
End_Procedure
```

Visual Objects
```
METHOD OCX_FocusChanged() CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_FocusChanged()
{
```

```
}
```

XBasic

```
function FocusChanged as v ()
end function
```

dBASE

```
function nativeObject_FocusChanged()
return
```

event FormatColumn (Item as HITEM, ColIndex as Long, Value as Variant)

Fired when a cell requires to format its caption.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item being formatted.
ColIndex as Long	A long expression that indicates the index of the column being formatted.
Value as Variant	A Variant value that indicates the value being displayed in the cell. By default, the Value parameter is initialized with the CellValue property.

Use the FormatColumn event to display a string different than the CellValue property. The FormatColumn event is fired only if the FireFormatColumn property of the Column is True. The FormatColumn event lets the user to provide the cell's caption before it is displayed on the control's list. For instance, the FormatColumn event is useful when the column cells contains prices(numbers), and you want to display that column formatted as currency, like $50 instead 50. Also, you can use the FormatColumn event to display item's index in the column, or to display the result of some operations based on the cells in the item (totals, currency conversion and so on).

The CellValue property of the cell is being shown as:

- formatted using the FormatCell property, if it is valid
- formatted using the FormatColumn property, if it is valid

In other words, all cells applies the format of the FormatColumn property, excepts the cells with the FormatCell property being set. If the cell belongs to a column with the FireFormatColumn property on True, the Value parameter of the FormatColumn event shows the newly caption for the cell to be shown.

Syntax for FormatColumn event, **/NET** version, on:

C#
```
private void FormatColumn(object sender,int Item,int ColIndex,ref object Value)
{
}
```

VB
```
Private Sub FormatColumn(ByVal sender As System.Object,ByVal Item As Integer,ByVal ColIndex As Integer,ByRef Value As Object) Handles FormatColumn
End Sub
```

Syntax for FormatColumn event, **/COM** version, on:

C#
```
private void FormatColumn(object sender,
AxEXG2ANTTLib._IG2anttEvents_FormatColumnEvent e)
{
}
```

C++
```
void OnFormatColumn(long Item,long ColIndex,VARIANT FAR* Value)
{
}
```

C++ Builder
```
void __fastcall FormatColumn(TObject *Sender,Exg2anttlib_tlb::HITEM Item,long
ColIndex,Variant * Value)
{
}
```

Delphi
```
procedure FormatColumn(ASender: TObject; Item : HITEM;ColIndex : Integer;var
Value : OleVariant);
begin
end;
```

Delphi 8 (.NET only)
```
procedure FormatColumn(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_FormatColumnEvent);
begin
end;
```

Powe…
```
begin event FormatColumn(long Item,long ColIndex,any Value)
end event FormatColumn
```

VB.NET
```
Private Sub FormatColumn(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_FormatColumnEvent) Handles FormatColumn
End Sub
```

VB6
```
Private Sub FormatColumn(ByVal Item As EXG2ANTTLibCtl.HITEM,ByVal ColIndex
As Long,Value As Variant)
End Sub
```

VBA
```
Private Sub FormatColumn(ByVal Item As Long,ByVal ColIndex As Long,Value As
```

```
Variant)
End Sub
```

VFP

```
LPARAMETERS Item,ColIndex,Value
```

Xbas…

```
PROCEDURE OnFormatColumn(oG2antt,Item,ColIndex,Value)
RETURN
```

Syntax for FormatColumn event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="FormatColumn(Item,ColIndex,Value)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function FormatColumn(Item,ColIndex,Value)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComFormatColumn HITEM llItem Integer llColIndex Variant llValue
    Forward Send OnComFormatColumn llItem llColIndex llValue
End_Procedure
```

Visual Objects

```
METHOD OCX_FormatColumn(Item,ColIndex,Value) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_FormatColumn(int _Item,int _ColIndex,COMVariant /*variant*/
_Value)
{
}
```

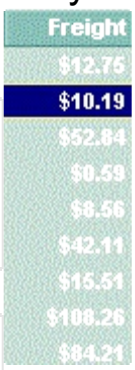
XBasic

```
function FormatColumn as v (Item as OLE::Exontrol.G2antt.1::HITEM,ColIndex as
N,Value as A)
end function
```

dBASE

```
function nativeObject_FormatColumn(Item,ColIndex,Value)
return
```

The following VB samples use the FormatCurrency function, to display a number as a currency. The FormatCurrency VB function returns an expression formatted as a currency value using the currency symbol defined in the system control panel.




```
G2antt1.Columns("Freight").FireFormatColumn = True
G2antt1.Columns("Freight").HeaderBold = True
G2antt1.Columns("Freight").Alignment = RightAlignment
```

```
Private Sub G2antt1_FormatColumn(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal
ColIndex As Long, Value As Variant)
```

```
    Value = FormatCurrency(Value, 2) ' The FormatCurrency is a VB function
```

```
End Sub
```

if the sample looks like following:


```
G2antt1.Columns("Freight").FireFormatColumn = False
G2antt1.Columns("Freight").HeaderBold = True
G2antt1.Columns("Freight").Alignment = RightAlignment
```

For instance, you can use the FormatColumn event to display "Yes" or "No" caption for a boolean column. The following VB sample shows how to do it:

```
Private Sub G2antt1_FormatColumn(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal
ColIndex As Long, Value As Variant)
   Value = IIf(Value < 50, "Yes", "No")
End Sub
```

The following VB sample displays the result of adding (concatenating) of two cells:

```
Private Sub G2antt1_FormatColumn(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal
ColIndex As Long, Value As Variant)
   With G2antt1.Items
      Value = .CellValue(Item, 0) + .CellValue(Item, 1)
   End With
End Sub
```

The FormatColumn event is fired before displaying a cell, so you can handle the FormatColumn to display anything on the cell at runtime. This way you can display the row

position, you can display the value using the currency format, and so on. The FireFormatColumn property allows the control to fire the FormatColumn event for the column. The Position property specifies the position of the column.

- If your chart does *not* display a tree or a hierarchy this property is ok to be used with FormatColumn event to display the position

The following VB sample handles the FormatColumn event to display the row position:

```
Private Sub G2antt1_FormatColumn(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal
ColIndex As Long, Value As Variant)
   Value = G2antt1.Items.ItemPosition(Item)
End Sub
```

- If your chart displays a tree or a hierarchy the position of the item must be determined relative to the FirstVisibleItem as shown in the following VB sample:

```
Private Sub G2antt1_FormatColumn(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal
ColIndex As Long, Value As Variant)
   Value = G2antt1.ScrollPos(True) + RelPos(Item)
End Sub

Private Function RelPos(ByVal hVisible As Long) As Long
   With G2antt1.Items
      Dim h As Long, i As Long, n As Long
      i = 0
      n = .VisibleCount + 1
      h = .FirstVisibleItem
      While (i <= n) And h <> 0 And h <> hVisible
         i = i + 1
         h = .NextVisibleItem(h)
      Wend
      RelPos = i
   End With
End Function
```

The following C++ sample displays a date column using a format like "Saturday, January 31, 2004":

```
void OnFormatColumnG2antt1(long Item, long ColIndex, VARIANT FAR* Value)
```

```
{
   COleDateTime date( *Value );
   COleVariant vtNewValue( date.Format( _T("%A, %B %d, %Y") ) );
   VariantCopy( Value, vtNewValue );
}
```

The following VB.NET sample displays a date column using LongDate format:

```
Private Sub AxG2antt1_FormatColumn(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_FormatColumnEvent) Handles AxG2antt1.FormatColumn
   e.value = DateTime.Parse(e.value).ToLongDateString()
End Sub
```

The following C# sample displays a date column using LongDate format:

```
private void axG2antt1_FormatColumn(object sender,
AxEXG2ANTTLib._IG2anttEvents_FormatColumnEvent e)
{
   e.value = DateTime.Parse(e.value.ToString()).ToLongDateString();
}
```

The following VFP sample displays the item's index using the FormatColumn event:

```
*** ActiveX Control Event ***
LPARAMETERS item, colindex, value

with thisform.G2antt1.Items
   .DefaultItem = item
   value = .ItemToIndex(0)
endwith
```

before running the sample please make sure that the :

```
application.AutoYield = .f.
```

is called during the Form.Init event.

event HistogramBoundsChanged (X as Long, Y as Long, Width as Long, Height as Long)

Occurs when the location and the size of the histogram is changed.

Type	Description
X as Long	A Long expression that specifies the location of the chart's histogram (client coordinate)
Y as Long	A Long expression that specifies the location of the chart's histogram (client coordinate)
Width as Long	A Long expression that specifies the width of the chart's histogram.
Height as Long	A Long expression that specifies the height of the chart's histogram.

The HistogramBoundsChanged event notifies your application when the bounds of the left part of the chart's histogram is changed. The /NET assembly passes the bounds of the chart's histogram as a Rectangle, instead passing all coordinates as X, Y, Width or Height. The HistogramVisible property specifies whether the control displays the chart's histogram in the bottom side. Changing the HistogramVisible property invokes the HistogramBoundsChanged event with an empty rectangle, if the histogram is being hidden, or the new position if the histogram is shown. For instance, you can hide or show the histogram legend component, when the Width and Heigh parameters are not zero. Use the HistogramBoundsChanged event to resize a component being displayed in the chart's histogram. Use the BeforeDrawPart and AfterDrawPart events to add your custom drawing to be shown in the component.

Syntax for HistogramBoundsChanged event, **/NET** version, on:

C#
```
private void HistogramBoundsChanged(object sender,int X,int Y,int Width,int Height)
{
}
```

VB
```
Private Sub HistogramBoundsChanged(ByVal sender As System.Object,ByVal X As Integer,ByVal Y As Integer,ByVal Width As Integer,ByVal Height As Integer) Handles HistogramBoundsChanged
End Sub
```

Syntax for HistogramBoundsChanged event, **/COM** version, on:

C#

```
private void HistogramBoundsChanged(object sender,
AxEXG2ANTTLib._IG2anttEvents_HistogramBoundsChangedEvent e)
{
}
```

C++

```
void OnHistogramBoundsChanged(long X,long Y,long Width,long Height)
{
}
```

C++ Builder

```
void __fastcall HistogramBoundsChanged(TObject *Sender,long X,long Y,long
Width,long Height)
{
}
```

Delphi

```
procedure HistogramBoundsChanged(ASender: TObject; X : Integer;Y :
Integer;Width : Integer;Height : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure HistogramBoundsChanged(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_HistogramBoundsChangedEvent);
begin
end;
```

Powe…

```
begin event HistogramBoundsChanged(long X,long Y,long Width,long Height)
end event HistogramBoundsChanged
```

VB.NET

```
Private Sub HistogramBoundsChanged(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_HistogramBoundsChangedEvent) Handles
HistogramBoundsChanged
End Sub
```

VB6

```
Private Sub HistogramBoundsChanged(ByVal X As Long,ByVal Y As Long,ByVal
Width As Long,ByVal Height As Long)
End Sub
```

VBA

```
Private Sub HistogramBoundsChanged(ByVal X As Long,ByVal Y As Long,ByVal Width As Long,ByVal Height As Long)
End Sub
```

VFP

```
LPARAMETERS X,Y,Width,Height
```

Xbas…

```
PROCEDURE OnHistogramBoundsChanged(oG2antt,X,Y,Width,Height)
RETURN
```

Syntax for HistogramBoundsChanged event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="HistogramBoundsChanged(X,Y,Width,Height)"
LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function HistogramBoundsChanged(X,Y,Width,Height)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComHistogramBoundsChanged Integer llX Integer llY Integer llWidth Integer llHeight
    Forward Send OnComHistogramBoundsChanged llX llY llWidth llHeight
End_Procedure
```

Visual Objects

```
METHOD OCX_HistogramBoundsChanged(X,Y,Width,Height) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_HistogramBoundsChanged(int _X,int _Y,int _Width,int _Height)
{
}
```

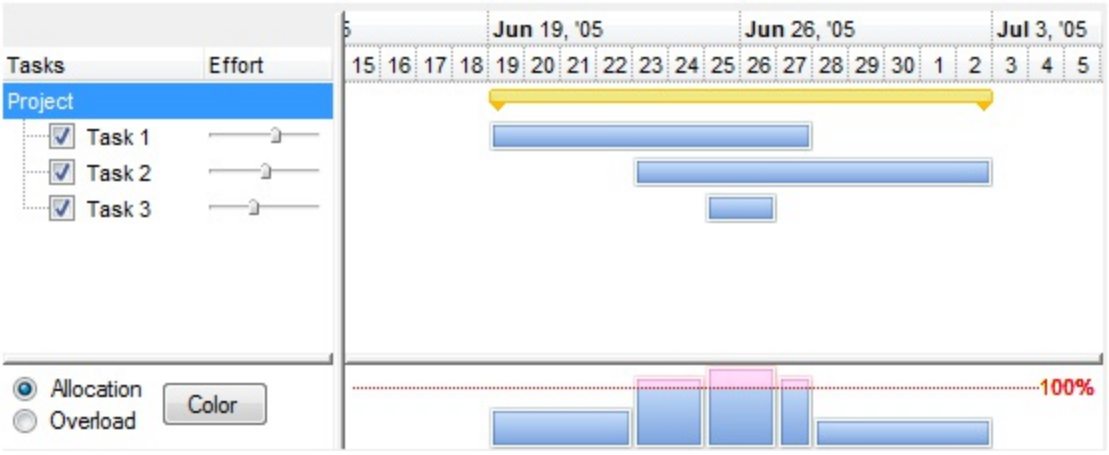
XBasic

```
function HistogramBoundsChanged as v (X as N,Y as N,Width as N,Height as N)
end function
```

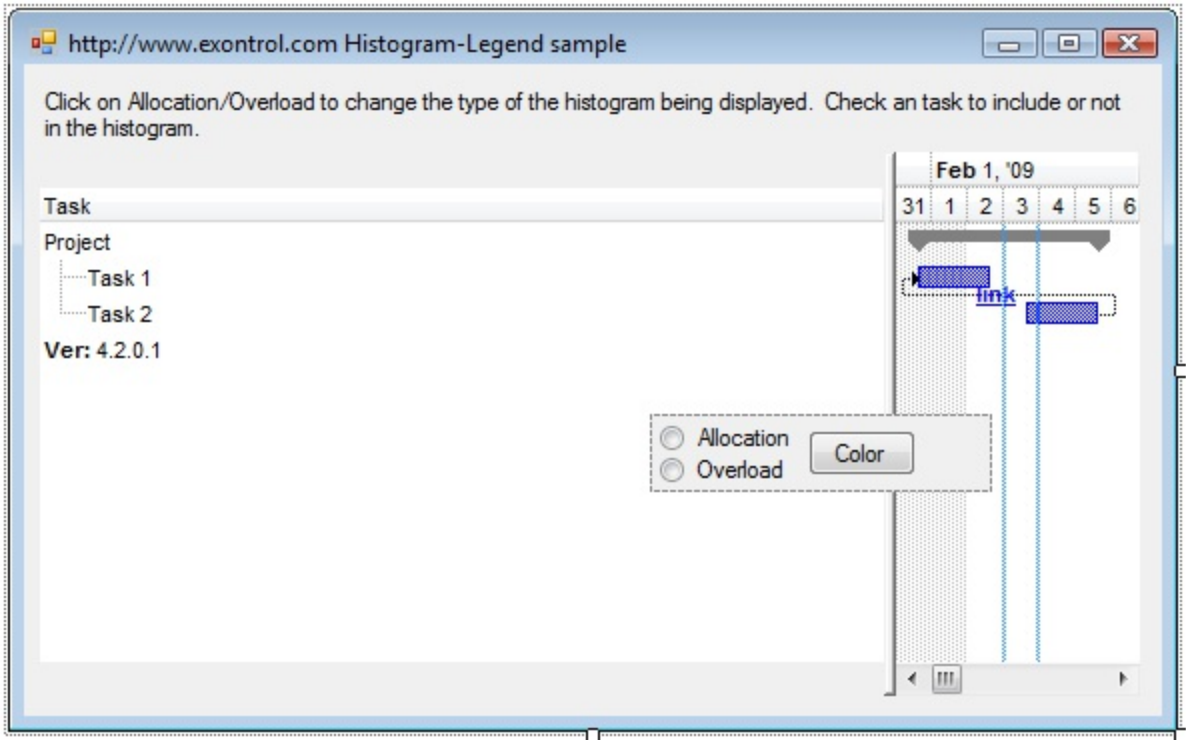
dBASE

```
function nativeObject_HistogramBoundsChanged(X,Y,Width,Height)
return
```

For instance, you can use the Controls property of the /NET assembly to add new components inside the control, and using the HistogramBoundsChanged event you can control it so they will be displayed in the left part of the chart's histogram as show in the following screen shot:



The "Allocation/Overload, Color" panel is being added as a Panel component that belongs to the form, as shown in the following screen shot:



You can define your legend for the histogram by doing the following:

- Add a Panel component to the form, in case you require multiple components being

displayed in the chart's histogram.

- Add all components to the panel, that you might want to use in the histogram.
- Call the exgantt1.Controls.Add method to add the newly panel/ component as being a child of the control.
- Handle the HistogramBoundsChanged event so you update the location and the size of the panel, with the new coordinates as follows:

```
private void exg2antt1_HistogramBoundsChanged(object sender, Rectangle Bounds)
{
    panel1.Bounds = Bounds;
}
```

The events of the added panel/component can be handled in the usual manner, as they still send events to the form, not to the exg2antt control.

A similar technique can be used for environment such as VB6/C++/eDeveloper/Clarion using the SetParent and SetWindowPos API functions as follow:

- Add a control or a collection of controls to the same form where the control is hosted. In our sample we will add another g2antt control named G2antt2
- Add API declarations as

```
Private Declare Function SetParent Lib "user32" (ByVal hWndChild As Long, ByVal
hWndNewParent As Long) As Long
Private Declare Function SetWindowPos Lib "user32" (ByVal hwnd As Long, ByVal
hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long, ByVal cx As Long, ByVal cy
As Long, ByVal wFlags As Long) As Long
Private Const SWP_NOZORDER = &H4
Private Const SWP_SHOWWINDOW = &H40
```

- Handle the HistogramBoundsChanged event and add the code

```
Private Sub G2antt1_HistogramBoundsChanged(ByVal x As Long, ByVal y As Long,
ByVal Width As Long, ByVal Height As Long)
On Error Resume Next
   With G2antt2
      Width = Width - 20
      SetWindowPos .hwnd, 0, x, y, Width, Height, SWP_NOZORDER Or
```

```
SWP_SHOWWINDOW
        .Left = x * Screen.TwipsPerPixelX
        .Top = y * Screen.TwipsPerPixelY
        .Width = Width * Screen.TwipsPerPixelX
        .Height = Height * Screen.TwipsPerPixelY
    End With
End Sub
```

- Add the Form_load event as:

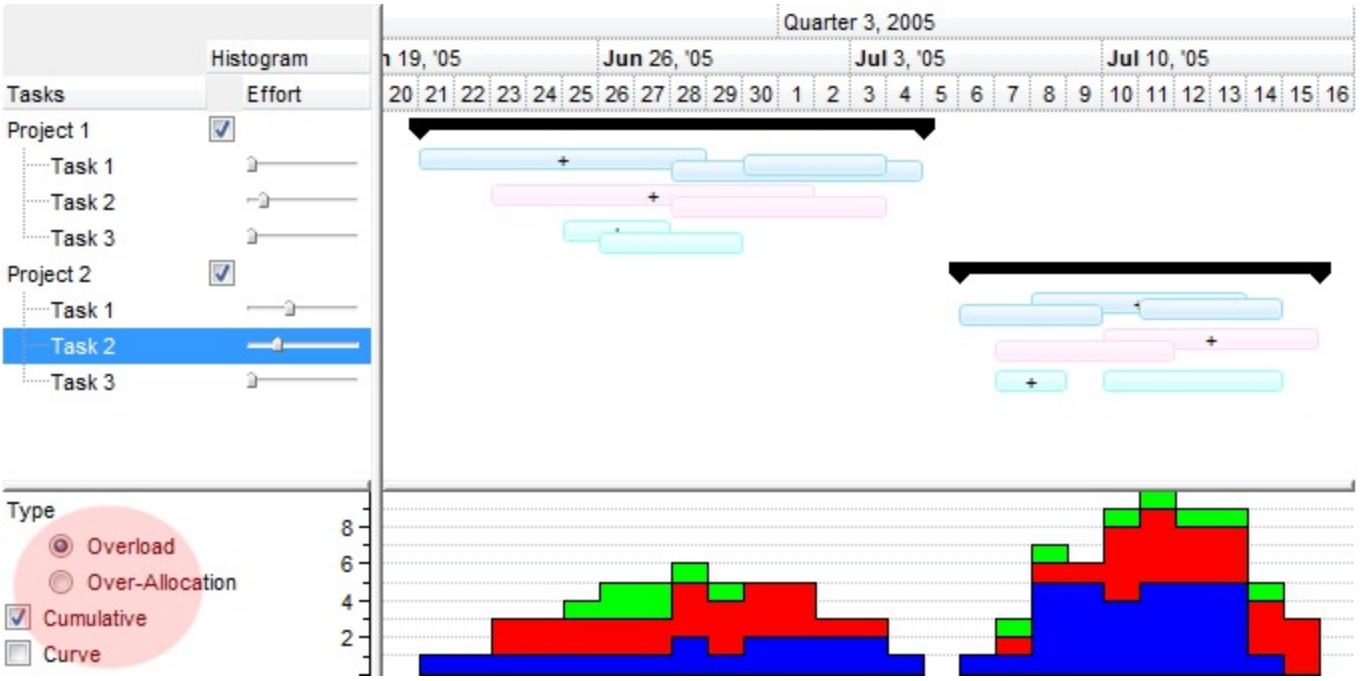
```
Private Sub Form_Load()
    With G2antt1
        .BeginUpdate
            SetParent G2antt2.hwnd, G2antt1.hwnd
            With .Chart
                .HistogramVisible = True
                .HistogramHeight = 134
            End With
        .EndUpdate
    End With
End Sub
```

The sample changes the parent of the g2antt control to be the g2antt1 control and reposition the inside control. In the following screen shot the left side of the histogram is another eXG2antt control with the template:

```
BeginUpdate()
BackColor = RGB(255,255,255)
Appearance = 0
ScrollBars = 0
HeaderVisible = False
OnResizeControl = 128
Columns.Add("Diagram")
{
    AllowSort = False
    AllowDragging = False
}
Chart
```

```
{
   PaneWidth(1) = 0
   ScrollBar = False
}
SelBackColor = BackColor
SelForeColor = ForeColor
ShowFocusRect = False
HasLines = 0
Items
{
   Dim h, h1,h2
   h = AddItem("Type")
   h1 = InsertItem(h,,"Overload")
   CellHasRadioButton(h1,0) = True
   CellState(h1,0) = 1
   CellRadioGroup(h1,0) = 1234
   h1 = InsertItem(h,,"Over-Allocation")
   CellHasRadioButton(h1,0) = True
   CellRadioGroup(h1,0) = 1234
   ExpandItem(h) = True
   h = AddItem("Cumulative")
   CellHasCheckBox(h,0) = 1
   h = AddItem("Curve")
   CellHasCheckBox(h,0) = 1
}
EndUpdate()
```

The following screen shot shows in red the inside eXG2antt control being placed in the left part of the histogram.


Quarter 3, 2005
Histogram
Tasks
Effort
Jun 26, '05
Jul 3, '05
Jul 10, '05
20 21 22 23 24 25 26 27 28 29 30 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Project 1
Task 1
Task 2
Task 3
Project 2
Task 1
Task 2
Task 3
Type
Overload
Over-Allocation
Cumulative
Curve
8
6
4
2

event HyperLinkClick (Item as HITEM, ColIndex as Long)

Occurs when the user clicks on a hyperlink cell.

Type	Description
Item as HITEM	A long expression that indicates the item's handle.
ColIndex as Long	A long expression that indicates the column's index.

The HyperLinkClick event is fired when user clicks a hyperlink cell. A hyperlink cell has the CellHyperLink property on True. The control changes the shape of the cursor when the mouse hovers a hyper linkcell. Use the HyperLinkClick event to notify your application that a hyperlink cell is clicked. Use the HyperLinkColor property to specify the hyperlink color. The HyperLinkClick event is fired only if the user clicks a cell that has the CellHyperLink property on True. Use the ItemFromPoint property to get an item or a cell from point. Use the ColumnFromPoint property to get the column from point.

Syntax for HyperLinkClick event, **/NET** version, on:

C#

```
private void HyperLinkClick(object sender,int Item,int ColIndex)
{
}
```

VB

```
Private Sub HyperLinkClick(ByVal sender As System.Object,ByVal Item As
Integer,ByVal ColIndex As Integer) Handles HyperLinkClick
End Sub
```

Syntax for HyperLinkClick event, **/COM** version, on:

C#

```
private void HyperLinkClick(object sender,
AxEXG2ANTTLib._IG2anttEvents_HyperLinkClickEvent e)
{
}
```

C++

```
void OnHyperLinkClick(long Item,long ColIndex)
{
}
```

C++ Builder

```
void __fastcall HyperLinkClick(TObject *Sender,Exg2anttlib_tlb::HITEM Item,long
ColIndex)
{
```

```
}
```

Delphi

```
procedure HyperLinkClick(ASender: TObject; Item : HITEM;ColIndex : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure HyperLinkClick(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_HyperLinkClickEvent);
begin
end;
```

Powe…

```
begin event HyperLinkClick(long Item,long ColIndex)
end event HyperLinkClick
```

VB.NET

```
Private Sub HyperLinkClick(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_HyperLinkClickEvent) Handles HyperLinkClick
End Sub
```

VB6

```
Private Sub HyperLinkClick(ByVal Item As EXG2ANTTLibCtl.HITEM,ByVal ColIndex
As Long)
End Sub
```

VBA

```
Private Sub HyperLinkClick(ByVal Item As Long,ByVal ColIndex As Long)
End Sub
```

VFP

```
LPARAMETERS Item,ColIndex

```

Xbas…

```
PROCEDURE OnHyperLinkClick(oG2antt,Item,ColIndex)
RETURN
```

Syntax for HyperLinkClick event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="HyperLinkClick(Item,ColIndex)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function HyperLinkClick(Item,ColIndex)
```

```
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComHyperLinkClick HITEM llItem Integer llColIndex
    Forward Send OnComHyperLinkClick llItem llColIndex
End_Procedure
```

Visual Objects

```
METHOD OCX_HyperLinkClick(Item,ColIndex) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_HyperLinkClick(int _Item,int _ColIndex)
{
}
```

XBasic

```
function HyperLinkClick as v (Item as OLE::Exontrol.G2antt.1::HITEM,ColIndex as N)
end function
```

dBASE

```
function nativeObject_HyperLinkClick(Item,ColIndex)
return
```

The following VB sample displays the caption of the hyperlink cell that's been clicked:

```
Private Sub G2antt1_HyperLinkClick(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal ColIndex
As Long)
    Debug.Print G2antt1.Items.CellValue(Item, ColIndex)
End Sub
```

The following VC sample displays the caption of the hyperlink cell that's been clicked:

```
#include "Items.h"

static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
    if ( pv )
    {
        if ( pv->vt == VT_ERROR )
            return szDefault;
```

```
        COleVariant vt;
        vt.ChangeType( VT_BSTR, pv );
        return V_BSTR( &vt );
    }
    return szDefault;
}

void OnHyperLinkClickG2antt1(long Item, long ColIndex)
{
    CItems items = m_g2antt.GetItems();
    COleVariant vtItem( Item ), vtColumn( ColIndex );
    OutputDebugString( V2S( &items.GetCellValue( vtItem, vtColumn ) ) );
}
```

The following VB.NET sample displays the caption of the hyperlink cell that's been clicked:

```
Private Sub AxG2antt1_HyperLinkClick(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_HyperLinkClickEvent) Handles AxG2antt1.HyperLinkClick
    With AxG2antt1.Items
        Debug.WriteLine(.CellValue(e.item, e.colIndex))
    End With
End Sub
```

The following C# sample displays the caption of the hyperlink cell that's been clicked:

```
private void axG2antt1_HyperLinkClick(object sender,
AxEXG2ANTTLib._IG2anttEvents_HyperLinkClickEvent e)
{
    System.Diagnostics.Debug.WriteLine( axG2antt1.Items.get_CellValue(e.item, e.colIndex )
);
}
```

The following VFP sample displays the caption of the hyperlink cell that's been clicked:

```
*** ActiveX Control Event ***
LPARAMETERS item, colindex

with thisform.G2antt1.Items
    .DefaultItem = item
```

```
    wait window nowait .CellValue( 0, colindex )
endwith
```

event InsideZoom (DateTime as Date)

Notifies your application that a date is about to be magnified.

Type	Description
DateTime as Date	A Date-Time expression that indicates the date being magnified.

The control InsideZoom event notifies your application once the user adds a new inside zoom date. Use the InsideZoom event to customize the default format for inside zoom units. The Item property of the InsideZooms collection retrieves the InsideZoom event being added. The DefaultInsideZoomFormat property retrieves the InsideZoomFormat object to customize the format of the time units being magnified. The DefaultInsideZoomFormat object is applied to all new inside zoom units, unless they are using a custom format. Use the CustomFormat property (if the AllowCustomFormat property is True) to access the custom format. If your chart displays inside zoom in the same format, you can use the DefaultInsideZoomFormat property to specify the format for all inside zoom units. If not, you can use the CustomFormat property to customize each inside zoom unit. In other words, the DefaultInsideZoomFormat is applied for all inside zoom units, that has the AllowCustomFormat property on False (by default). The ChartStartChanging(exBaseLevelDblClk) event notifies your application once the user double clicks an unit in the chart's base level (the base level defines the time scale unit being shown for the bars).

Syntax for InsideZoom event, **/NET** version, on:

C#

```
private void InsideZoom(object sender,DateTime DateTime)
{
}
```

VB

```
Private Sub InsideZoom(ByVal sender As System.Object,ByVal DateTime As Date)
Handles InsideZoom
End Sub
```

Syntax for InsideZoom event, **/COM** version, on:

C#

```
private void InsideZoom(object sender,
AxEXG2ANTTLib._IG2anttEvents_InsideZoomEvent e)
{
}
```

C++

```
void OnInsideZoom(DATE DateTime)
```

```
{
}
```

C++ Builder

```
void __fastcall InsideZoom(TObject *Sender,DATE DateTime)
{
}
```

Delphi

```
procedure InsideZoom(ASender: TObject; DateTime : TDateTime);
begin
end;
```

Delphi 8 (.NET only)

```
procedure InsideZoom(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_InsideZoomEvent);
begin
end;
```

Powe…

```
begin event InsideZoom(datetime DateTime)
end event InsideZoom
```

VB.NET

```
Private Sub InsideZoom(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_InsideZoomEvent) Handles InsideZoom
End Sub
```

VB6

```
Private Sub InsideZoom(ByVal DateTime As Date)
End Sub
```

VBA

```
Private Sub InsideZoom(ByVal DateTime As Date)
End Sub
```

VFP

```
LPARAMETERS DateTime
```

Xbas…

```
PROCEDURE OnInsideZoom(oG2antt,DateTime)
RETURN
```

Syntax for InsideZoom event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="InsideZoom(DateTime)" LANGUAGE="JScript">
```

```
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function InsideZoom(DateTime)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComInsideZoom DateTime llDateTime
    Forward Send OnComInsideZoom llDateTime
End_Procedure
```

Visual Objects

```
METHOD OCX_InsideZoom(DateTime) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_InsideZoom(date _DateTime)
{
}
```

XBasic

```
function InsideZoom as v (DateTime as T)
end function
```

dBASE

```
function nativeObject_InsideZoom(DateTime)
return
```

event ItemOleEvent (Item as HITEM, Ev as OleEvent)

Fired when an ActiveX control hosted by an item has fired an event.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item that hosts an ActiveX control.
Ev as OleEvent	An OleEvent object that contains information about the fired event.

The Exontrol's ExG2antt control supports ActiveX hosting. The InsertItemControl method inserts an item that hosts an ActiveX control. The ItemOleEvent event notifies your application that a hosted ActiveX control fires an event. The ItemObject property gets the ActiveX object hosted by an item that is inserted using the InsertControlItem method. The ItemObject property gets nothing if the item doesn't host an ActiveX control, or if inserting an ActiveX control failed).

Syntax for ItemOleEvent event, **/NET** version, on:

C#
```
private void ItemOleEvent(object sender,int Item,exontrol.EXG2ANTTLib.OleEvent Ev)
{
}
```

VB
```
Private Sub ItemOleEvent(ByVal sender As System.Object,ByVal Item As Integer,ByVal Ev As exontrol.EXG2ANTTLib.OleEvent) Handles ItemOleEvent
End Sub
```

Syntax for ItemOleEvent event, **/COM** version, on:

C#
```
private void ItemOleEvent(object sender, AxEXG2ANTTLib._IG2anttEvents_ItemOleEventEvent e)
{
}
```

C++
```
void OnItemOleEvent(long Item,LPDISPATCH Ev)
{
}
```

C++ Builder

```
void __fastcall ItemOleEvent(TObject *Sender,Exg2anttlib_tlb::HITEM
Item,Exg2anttlib_tlb::IOleEvent *Ev)
{
}
```

Delphi

```
procedure ItemOleEvent(ASender: TObject; Item : HITEM;Ev : IOleEvent);
begin
end;
```

Delphi 8 (.NET only)

```
procedure ItemOleEvent(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_ItemOleEventEvent);
begin
end;
```

Powe…

```
begin event ItemOleEvent(long Item,oleobject Ev)
end event ItemOleEvent
```

VB.NET

```
Private Sub ItemOleEvent(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_ItemOleEventEvent) Handles ItemOleEvent
End Sub
```

VB6

```
Private Sub ItemOleEvent(ByVal Item As EXG2ANTTLibCtl.HITEM,ByVal Ev As
EXG2ANTTLibCtl.IOleEvent)
End Sub
```

VBA

```
Private Sub ItemOleEvent(ByVal Item As Long,ByVal Ev As Object)
End Sub
```

VFP

```
LPARAMETERS Item,Ev

```

Xbas…

```
PROCEDURE OnItemOleEvent(oG2antt,Item,Ev)
RETURN
```

Syntax for ItemOleEvent event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="ItemOleEvent(Item,Ev)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function ItemOleEvent(Item,Ev)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComItemOleEvent HITEM llItem Variant llEv
    Forward Send OnComItemOleEvent llItem llEv
End_Procedure
```

Visual Objects

```
METHOD OCX_ItemOleEvent(Item,Ev) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_ItemOleEvent(int _Item,COM _Ev)
{
}
```

XBasic

```
function ItemOleEvent as v (Item as OLE::Exontrol.G2antt.1::HITEM,Ev as
OLE::Exontrol.G2antt.1::IOleEvent)
end function
```

dBASE

```
function nativeObject_ItemOleEvent(Item,Ev)
return
```

The following VB sample adds an item that hosts the Microsoft Calendar Control and prints each event fired by that ActiveX control:

```
G2antt1.Items.ItemHeight(G2antt1.Items.InsertControlItem(, "MSCal.Calendar")) = 256
```

```
Private Sub G2antt1_ItemOleEvent(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal Ev As
EXG2ANTTLibCtl.IOleEvent)
  Debug.Print "Event name:" & Ev.Name
  If (Ev.CountParam = 0) Then
    Debug.Print "The event has no arguments."
  Else
    Debug.Print "The event has the following arguments:"
    Dim i As Long
    For i = 0 To Ev.CountParam - 1
      Debug.Print Ev(i).Name; " = " & Ev(i).Value
    Next
  End If
End Sub
```

The following VC sample displays the events that an ActiveX control is firing while it is hosted by an item:

```
#import <exg2antt.dll> rename( "GetItems", "exGetItems" )

static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
   if ( pv )
   {
      if ( pv->vt == VT_ERROR )
         return szDefault;

      COleVariant vt;
      vt.ChangeType( VT_BSTR, pv );
      return V_BSTR( &vt );
   }
   return szDefault;
}

void OnItemOleEventG2antt1(long Item, LPDISPATCH Ev)
{
   EXG2ANTTLib::IOleEventPtr spEvent( Ev );
   CString strOutput;
```

```
    strOutput.Format( "Event's name: %s\n", spEvent->Name.operator const char *() );
    OutputDebugString( strOutput );
    if ( spEvent->CountParam == 0 )
        OutputDebugString( "The event has no parameters." );
    else
    {
        for ( long i = 0; i < spEvent->CountParam; i++ )
        {
            EXG2ANTTLib::IOleEventParamPtr spParam = spEvent->GetParam( COleVariant( i ) );
            strOutput.Format( "Name: %s, Value: %s\n", spParam->Name.operator const char *
(), V2S( &spParam->Value ) );
            OutputDebugString( strOutput );
        }
    }
    OutputDebugString( "" );
}
```

The #import clause is required to get the wrapper classes for IOleEvent and IOleEventParam objects, that are not defined by the MFC class wizard. The same #import statement defines the EXG2ANTTLib namespace that include all objects and types of the control's TypeLibrary. In case your exg2antt.dll library is located to another place than the system folder or well known path, the path to the library should be provided, in order to let the VC finds the type library.

The following VB.NET sample displays the events that an ActiveX control is firing while it is hosted by an item:

```
Private Sub AxG2antt1_ItemOleEvent(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_ItemOleEventEvent) Handles AxG2antt1.ItemOleEvent
    Debug.WriteLine("Event's name: " & e.ev.Name)
    Dim i As Long
    For i = 0 To e.ev.CountParam - 1
        Dim eP As EXG2ANTTLib.OleEventParam
        eP = e.ev(i)
        Debug.WriteLine("Name: " & e.ev.Name & " Value: " & eP.Value)
    Next
End Sub
```

The following C# sample displays the events that an ActiveX control is firing while it is

hosted by an item:

```
private void axG2antt1_ItemOleEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_ItemOleEventEvent e)
{
   System.Diagnostics.Debug.WriteLine( "Event's name: " + e.ev.Name.ToString() );
   for ( int i= 0; i < e.ev.CountParam ; i++ )
   {
      EXG2ANTTLib.IOleEventParam evP = e.ev[i];
      System.Diagnostics.Debug.WriteLine( "Name: " + evP.Name.ToString() + ", Value: " +
evP.Value.ToString() );
   }
}
```

The following VFP sample displays the events that an ActiveX control fires when it is hosted by an item:

```
*** ActiveX Control Event ***
LPARAMETERS item, ev

local s
s = "Event's name: " + ev.Name
for i = 0 to ev.CountParam - 1
      s = s +  "Name: " + ev.Param(i).Name + " ,Value: " + Str(ev.Param(i).Value)
endfor
wait window nowait s
```

event KeyDown (KeyCode as Integer, Shift as Integer)

Occurs when the user presses a key while an object has the focus.

Type	Description
KeyCode as Integer	An integer that represent the key code.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT are pressed, the value of shift is 6.

Use KeyDown and KeyUp event procedures if you need to respond to both the pressing and releasing of a key. Use the ExpandOnKeys property to specify whether the user expands or collapses the focused items using arrow keys. You test for a condition by first assigning each result to a temporary integer variable and then comparing shift to a bit mask. Use the And operator with the shift argument to test whether the condition is greater than 0, indicating that the modifier was pressed, as in this example:

ShiftDown = (Shift And 1) > 0
CtrlDown = (Shift And 2) > 0
AltDown = (Shift And 4) > 0
In a procedure, you can test for any combination of conditions, as in this example:
If AltDown And CtrlDown Then

Syntax for KeyDown event, **/NET** version, on:

C#

```
private void KeyDown(object sender,ref short KeyCode,short Shift)
{
}
```

VB

```
Private Sub KeyDown(ByVal sender As System.Object,ByRef KeyCode As
Short,ByVal Shift As Short) Handles KeyDown
End Sub
```

Syntax for KeyDown event, **/COM** version, on:

C#

```
private void KeyDownEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_KeyDownEvent e)
{
}
```

C++

```
void OnKeyDown(short FAR* KeyCode,short Shift)
{
}
```

C++ Builder

```
void __fastcall KeyDown(TObject *Sender,short * KeyCode,short Shift)
{
}
```

Delphi

```
procedure KeyDown(ASender: TObject; var KeyCode : Smallint;Shift : Smallint);
begin
end;
```

Delphi 8 (.NET only)

```
procedure KeyDownEvent(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_KeyDownEvent);
begin
end;
```

Powe…

```
begin event KeyDown(integer KeyCode,integer Shift)
end event KeyDown
```

VB.NET

```
Private Sub KeyDownEvent(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_KeyDownEvent) Handles KeyDownEvent
End Sub
```

VB6

```
Private Sub KeyDown(KeyCode As Integer,Shift As Integer)
End Sub
```

VBA

```
Private Sub KeyDown(KeyCode As Integer,ByVal Shift As Integer)
End Sub
```

VFP

```
LPARAMETERS KeyCode,Shift
```

Xbas…
```
PROCEDURE OnKeyDown(oG2antt,KeyCode,Shift)
RETURN
```

Syntax for KeyDown event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="KeyDown(KeyCode,Shift)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function KeyDown(KeyCode,Shift)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComKeyDown Short llKeyCode Short llShift
    Forward Send OnComKeyDown llKeyCode llShift
End_Procedure
```

Visual Objects
```
METHOD OCX_KeyDown(KeyCode,Shift) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_KeyDown(COMVariant /*short*/ _KeyCode,int _Shift)
{
}
```

XBasic
```
function KeyDown as v (KeyCode as N,Shift as N)
end function
```

dBASE
```
function nativeObject_KeyDown(KeyCode,Shift)
return
```

event KeyPress (KeyAscii as Integer)

Occurs when the user presses and releases an ANSI key.

Type	Description
KeyAscii as Integer	An integer that returns a standard numeric ANSI keycode.

The KeyPress event lets you immediately test keystrokes for validity or for formatting characters as they are typed. Changing the value of the keyascii argument changes the character displayed. Use KeyDown and KeyUp event procedures to handle any keystroke not recognized by KeyPress, such as function keys, editing keys, navigation keys, and any combinations of these with keyboard modifiers. Unlike the KeyDown and KeyUp events, KeyPress does not indicate the physical state of the keyboard; instead, it passes a character. KeyPress interprets the uppercase and lowercase of each character as separate key codes and, therefore, as two separate characters.

Syntax for KeyPress event, **/NET** version, on:

C#

```
private void KeyPress(object sender,ref short KeyAscii)
{
}
```

VB

```
Private Sub KeyPress(ByVal sender As System.Object,ByRef KeyAscii As Short)
Handles KeyPress
End Sub
```

Syntax for KeyPress event, **/COM** version, on:

C#

```
private void KeyPressEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_KeyPressEvent e)
{
}
```

C++

```
void OnKeyPress(short FAR* KeyAscii)
{
}
```

C++ Builder

```
void __fastcall KeyPress(TObject *Sender,short * KeyAscii)
{
}
```

Delphi

```
procedure KeyPress(ASender: TObject; var KeyAscii : Smallint);
begin
end;
```

Delphi 8 (.NET only)

```
procedure KeyPressEvent(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_KeyPressEvent);
begin
end;
```

Powe…

```
begin event KeyPress(integer KeyAscii)
end event KeyPress
```

VB.NET

```
Private Sub KeyPressEvent(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_KeyPressEvent) Handles KeyPressEvent
End Sub
```

VB6

```
Private Sub KeyPress(KeyAscii As Integer)
End Sub
```

VBA

```
Private Sub KeyPress(KeyAscii As Integer)
End Sub
```

VFP

```
LPARAMETERS KeyAscii

```

Xbas…

```
PROCEDURE OnKeyPress(oG2antt,KeyAscii)
RETURN
```

Syntax for KeyPress event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="KeyPress(KeyAscii)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function KeyPress(KeyAscii)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComKeyPress Short llKeyAscii
   Forward Send OnComKeyPress llKeyAscii
End_Procedure
```

Visual Objects

```
METHOD OCX_KeyPress(KeyAscii) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_KeyPress(COMVariant /*short*/ _KeyAscii)
{
}
```

XBasic

```
function KeyPress as v (KeyAscii as N)
end function
```

dBASE

```
function nativeObject_KeyPress(KeyAscii)
return
```

event KeyUp (KeyCode as Integer, Shift as Integer)

Occurs when the user releases a key while an object has the focus.

Type	Description
KeyCode as Integer	An integer that represent the key code.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT are pressed, the value of shift is 6.

Use the KeyUp event procedure to respond to the releasing of a key.

Syntax for KeyUp event, **/NET** version, on:

C#

```
private void KeyUp(object sender,ref short KeyCode,short Shift)
{
}
```

VB

```
Private Sub KeyUp(ByVal sender As System.Object,ByRef KeyCode As Short,ByVal
Shift As Short) Handles KeyUp
End Sub
```

Syntax for KeyUp event, **/COM** version, on:

C#

```
private void KeyUpEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_KeyUpEvent e)
{
}
```

C++

```
void OnKeyUp(short FAR* KeyCode,short Shift)
{
}
```

C++ Builder

```
void __fastcall KeyUp(TObject *Sender,short * KeyCode,short Shift)
```

```
{
}
```

Delphi

```
procedure KeyUp(ASender: TObject; var KeyCode : Smallint;Shift : Smallint);
begin
end;
```

Delphi 8 (.NET only)

```
procedure KeyUpEvent(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_KeyUpEvent);
begin
end;
```

Powe…

```
begin event KeyUp(integer KeyCode,integer Shift)
end event KeyUp
```

VB.NET

```
Private Sub KeyUpEvent(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_KeyUpEvent) Handles KeyUpEvent
End Sub
```

VB6

```
Private Sub KeyUp(KeyCode As Integer,Shift As Integer)
End Sub
```

VBA

```
Private Sub KeyUp(KeyCode As Integer,ByVal Shift As Integer)
End Sub
```

VFP

```
LPARAMETERS KeyCode,Shift
```

Xbas…

```
PROCEDURE OnKeyUp(oG2antt,KeyCode,Shift)
RETURN
```

Syntax for KeyUp event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="KeyUp(KeyCode,Shift)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function KeyUp(KeyCode,Shift)
```

```
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComKeyUp Short llKeyCode Short llShift
    Forward Send OnComKeyUp llKeyCode llShift
End_Procedure
```

Visual Objects

```
METHOD OCX_KeyUp(KeyCode,Shift) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_KeyUp(COMVariant /*short*/ _KeyCode,int _Shift)
{
}
```

XBasic

```
function KeyUp as v (KeyCode as N,Shift as N)
end function
```

dBASE

```
function nativeObject_KeyUp(KeyCode,Shift)
return
```

event LayoutChanged ()

Occurs when column's position or column's size is changed.

Type	Description

The LayoutChanged event notifies your application once a column is resized or moved by drag and drop. Also, the LayoutChanged event may be fired if the item's position is changed by drag and drop using the AutoDrag property.

Syntax for LayoutChanged event, **/NET** version, on:

C#
```
private void LayoutChanged(object sender)
{
}
```

VB
```
Private Sub LayoutChanged(ByVal sender As System.Object) Handles
LayoutChanged
End Sub
```

Syntax for LayoutChanged event, **/COM** version, on:

C#
```
private void LayoutChanged(object sender, EventArgs e)
{
}
```

C++
```
void OnLayoutChanged()
{
}
```

C++ Builder
```
void __fastcall LayoutChanged(TObject *Sender)
{
}
```

Delphi
```
procedure LayoutChanged(ASender: TObject; );
begin
end;
```

Delphi 8 (.NET only)
```
procedure LayoutChanged(sender: System.Object; e: System.EventArgs);
begin
```

```
end;
```

Powe…
```
begin event LayoutChanged()
end event LayoutChanged
```

VB.NET
```
Private Sub LayoutChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles LayoutChanged
End Sub
```

VB6
```
Private Sub LayoutChanged()
End Sub
```

VBA
```
Private Sub LayoutChanged()
End Sub
```

VFP
```
LPARAMETERS nop
```

Xbas…
```
PROCEDURE OnLayoutChanged(oG2antt)
RETURN
```

Syntax for LayoutChanged event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="LayoutChanged()" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function LayoutChanged()
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComLayoutChanged
    Forward Send OnComLayoutChanged
End_Procedure
```

Visual Objects
```
METHOD OCX_LayoutChanged() CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_LayoutChanged()
{
}
```

XBasic

```
function LayoutChanged as v ()
end function
```

dBASE

```
function nativeObject_LayoutChanged()
return
```

Since, the LayotChanged event may be fired on different scenarios, you can distingue the action that previously occurs by storing the ItemFromPoint and/or ColumnFromPoint during the MouseDown event like in the following VB sample:

```
Dim iItemFromPointMouseDown As Long
Dim iColumnFromPointMouseDown As Long

Private Sub Form_Load()
   iItemFromPointMouseDown = 0
   iColumnFromPointMouseDown = -1
End Sub

Private Sub G2antt1_LayoutChanged()
   If (iItemFromPointMouseDown <> 0) Then
         Debug.Print "Items section changed"
      Else
      If (iColumnFromPointMouseDown <> -1) Then
         Debug.Print "Columns section changed"
      Else
         Debug.Print "Others"
      End If
   End If
   iItemFromPointMouseDown = 0
   iColumnFromPointMouseDown = -1
End Sub
```

```
Private Sub G2antt1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
  Dim c As Long, hit As HitTestInfoEnum
  With G2antt1
    iItemFromPointMouseDown = .ItemFromPoint(-1, -1, c, hit)
    iColumnFromPointMouseDown = .ColumnFromPoint(-1, -1)
  End With
End Sub
```

The sample displays:

- "Columns section changed" if any change occurs in the Columns section, like moving a column to a new position or resizing the column.
- "Items section changed", if the user drags an item to a new position using the AutoDrag property on exAutoDragPosition, exAutoDragPositionKeepIndent and exAutoDragPositionAny

You can use the LayoutChanged event to save the columns position and size for future use. Use the Width property to retrieve the column's width. Use the Position property to retrieve the column's position. The Visible property specifies whether a column is shown or hidden. Use the ColumnAutoResize property to specify whether the visible columns fit the control's client area.

There are two options to avoid losing the columns proportions:

- Avoiding resizing the control under a specified width, like in the sample:

```
Private Sub Form_Resize()
On Error Resume Next
  If ScaleWidth / Screen.TwipsPerPixelX > 64 Then
    With G2antt1
      .Left = 0
      .Top = 0
      .Width = ScaleWidth
      .Height = ScaleHeight
    End With
  End If
End Sub
```

- Using the LayoutChanged event to store the columns proportions manually. The

following sample holds the columns proportions when LayoutChanged event is fired. The sample ensures that the proportions are saved only when the user resizes on of the control's columns, not when the user resizes the entire control. The proportions are kept by the Data property of the Column object. The sample can be changed smoothly by using a simple collection to hold the columns proportions instead using the Data property of the Column object

```
Option Explicit
Dim nFit As Long
Private Declare Function PeekMessage Lib "user32" Alias "PeekMessageA"
(lpMsg As MSG, ByVal hwnd As Long, ByVal wMsgFilterMin As Long, ByVal
wMsgFilterMax As Long, ByVal wRemoveMsg As Long) As Long
Private Declare Function TranslateMessage Lib "user32" (lpMsg As MSG) As Long
Private Declare Function DispatchMessage Lib "user32" Alias
"DispatchMessageA" (lpMsg As MSG) As Long
Private Const PM_REMOVE = &amp;amp;H1
Private Type POINTAPI
        x As Long
        y As Long
End Type
Private Type MSG
    hwnd As Long
    message As Long
    wParam As Long
    lParam As Long
    time As Long
    pt As POINTAPI
End Type

Private Sub Form_Load()
    nFit = 0

    onG2anttResize G2antt1
End Sub

Private Sub Form_Resize()
On Error Resume Next
    nFit = nFit + 1
```

```
    With G2antt1
        .Left = 0
        .Top = 0
        .Width = ScaleWidth
        .Height = ScaleHeight
    End With
    fit G2antt1

    nFit = nFit - 1
End Sub

Private Sub G2antt1_LayoutChanged()
    If (nFit = 0) Then
        onG2anttResize G2antt1
    End If
End Sub

Private Sub fit(ByVal g As EXG2ANTTLibCtl.G2antt)
    nFit = nFit + 1
    With g
        If (.ColumnAutoResize) Then
            .BeginUpdate
            .ColumnAutoResize = False
            Dim c As EXG2ANTTLibCtl.Column
            For Each c In .Columns
                c.Width = c.Data
            Next
            .ColumnAutoResize = True
            .EndUpdate
        End If
    End With
    waitToProcessMessages
    nFit = nFit - 1
End Sub

Private Sub onG2anttResize(ByVal g As EXG2ANTTLibCtl.G2antt)
    Dim c As Object
```

```
    With g
        If (.ColumnAutoResize) Then
            For Each c In .Columns
                c.Data = c.Width
            Next
        End If
    End With
End Sub

Private Sub waitToProcessMessages()
    Dim m As MSG
    While PeekMessage(m, 0, 0, 0, PM_REMOVE)
        TranslateMessage m
        DispatchMessage m
    Wend
End Sub
```

event MouseDown (Button as Integer, Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occurs when the user presses a mouse button.

Type	Description
Button as Integer	An integer that identifies the button that was pressed to cause the event as as 1 for Left Mouse Button, 2 for Right Mouse Button and 4 for Middle Mouse Button.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the button argument is pressed or released.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The X value is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The Y value is always expressed in container coordinates.

Use a MouseDown or MouseUp event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the Click and DblClick events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers. Use the ItemFromPoint property to get the item from point. Use the ColumnFromPoint property to get the column from point. Use the DateFromPoint property to specify the date from the cursor. Use the AnchorFromPoint property to retrieve the identifier of the anchor element from the cursor. The AnchorClick event notifies your application when the user clicks an anchor element. The NoteFromPoint property retrieves the note/box from the cursor. *Almost all properties that get an object from point supports -1,-1 coordinate that specifies the current cursor position, so no conversion is required for X and Y coordinates.*

Syntax for MouseDown event, **/NET** version, on:

C#

```
private void MouseDownEvent(object sender,short Button,short Shift,int X,int Y)
{
}
```

VB

```
Private Sub MouseDownEvent(ByVal sender As System.Object,ByVal Button As Short,ByVal Shift As Short,ByVal X As Integer,ByVal Y As Integer) Handles MouseDownEvent

```

```
End Sub
```

Syntax for MouseDown event, **/COM** version, on:

C#
```
private void MouseDownEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_MouseDownEvent e)
{
}
```

C++
```
void OnMouseDown(short Button,short Shift,long X,long Y)
{
}
```

C++ Builder
```
void __fastcall MouseDown(TObject *Sender,short Button,short Shift,int X,int Y)
{
}
```

Delphi
```
procedure MouseDown(ASender: TObject; Button : Smallint;Shift : Smallint;X :
Integer;Y : Integer);
begin
end;
```

Delphi 8 (.NET only)
```
procedure MouseDownEvent(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_MouseDownEvent);
begin
end;
```

Powe…
```
begin event MouseDown(integer Button,integer Shift,long X,long Y)
end event MouseDown
```

VB.NET
```
Private Sub MouseDownEvent(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_MouseDownEvent) Handles MouseDownEvent
End Sub
```

VB6
```
Private Sub MouseDown(Button As Integer,Shift As Integer,X As Single,Y As Single)
End Sub
```

VBA
```
Private Sub MouseDown(ByVal Button As Integer,ByVal Shift As Integer,ByVal X As
```

```
Long,ByVal Y As Long)
End Sub
```

VFP

```
LPARAMETERS Button,Shift,X,Y
```

Xbas…

```
PROCEDURE OnMouseDown(oG2antt,Button,Shift,X,Y)
RETURN
```

Syntax for MouseDown event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="MouseDown(Button,Shift,X,Y)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function MouseDown(Button,Shift,X,Y)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComMouseDown Short llButton Short llShift OLE_XPOS_PIXELS llX
OLE_YPOS_PIXELS llY
    Forward Send OnComMouseDown llButton llShift llX llY
End_Procedure
```

Visual Objects

```
METHOD OCX_MouseDown(Button,Shift,X,Y) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_MouseDown(int _Button,int _Shift,int _X,int _Y)
{
}
```

XBasic

```
function MouseDown as v (Button as N,Shift as N,X as
OLE::Exontrol.G2antt.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.G2antt.1::OLE_YPOS_PIXELS)
end function
```

dBASE

```
function nativeObject_MouseDown(Button,Shift,X,Y)
return
```

The following Access sample prints the cell's caption that has been clicked:

```
Private Sub G2antt1_MouseDown(ByVal Button As Integer, ByVal Shift As Integer, ByVal X As Long, ByVal Y As Long)
   Dim h As HITEM
   Dim c As Long, hit As Long
   h = G2antt1.ItemFromPoint(-1, -1, c, hit)
   If Not (h = 0) Then
      MsgBox "The """ & G2antt1.Items.CellValue(h, c) & """ cell has been double clicked."
   End If
End Sub
```

The following VB sample prints the cell's caption that has been clicked:

```
Private Sub G2antt1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
   ' Converts the container coordinates to client coordinates
   X = X / Screen.TwipsPerPixelX
   Y = Y / Screen.TwipsPerPixelY
   Dim h As HITEM
   Dim c As Long
   Dim hit As EXG2ANTTLibCtl.HitTestInfoEnum
   ' Gets the item from (X,Y)
   h = G2antt1.ItemFromPoint(X, Y, c, hit)
   If Not (h = 0) Then
      Debug.Print G2antt1.Items.CellValue(h, c) & " HT = " & hit
   End If
End Sub
```

If you need to add a context menu based on the item you can use the MouseUp event, like in the following VB sample (the sample uses the Exontrol's ExPopupMenu Component):

```
Private Sub G2antt1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
   If (Button = 2) Then
      ' Converts the container coordinates to client coordinates
      X = X / Screen.TwipsPerPixelX
      Y = Y / Screen.TwipsPerPixelY
```

```
      Dim h As HITEM
      Dim c As Long, hit as Long
      ' Gets the item from (X,Y)
      h = G2antt1.ItemFromPoint(X, Y, c, hit)
      If Not (h = 0) Then
         Dim i As Long
         PopupMenu1.Items.Add G2antt1.Items.CellValue(h, c)
         i = PopupMenu1.ShowAtCursor
      End If
   End If
End Sub
```

The following VC sample displays the caption of the cell being clicked:

```
#include "Items.h"

static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
   if ( pv )
   {
      if ( pv->vt == VT_ERROR )
         return szDefault;

      COleVariant vt;
      vt.ChangeType( VT_BSTR, pv );
      return V_BSTR( &vt );
   }
   return szDefault;
}

void OnMouseDownG2antt1(short Button, short Shift, long X, long Y)
{
   int c = 0, hit = 0, hItem = m_g2antt.GetItemFromPoint( X, Y, &c, &hit );
   if ( ( hItem != 0 ) || ( c != 0 ) )
   {
      CItems items = m_g2antt.GetItems();
      COleVariant vtItem( hItem ), vtColumn( c );
```

```
    CString strCaption = V2S( &items.GetCellValue( vtItem, vtColumn ) ), strOutput;
    strOutput.Format( "Cell: '%s', Hit = %08X\n", strCaption, hit );
    OutputDebugString( strOutput );
  }
}
```

The following VB.NET sample displays the caption from the cell being clicked:

```
Private Sub AxG2antt1_MouseDownEvent(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_MouseDownEvent) Handles
AxG2antt1.MouseDownEvent
  With AxG2antt1
    Dim i As Integer, c As Integer, hit As EXG2ANTTLib.HitTestInfoEnum
    i = .get_ItemFromPoint(e.x, e.y, c, hit)
    If (Not (i = 0) Or Not (c = 0)) Then
      Debug.WriteLine("Cell: " & .Items.CellValue(i, c) & " Hit: " & hit.ToString())
    End If
  End With
End Sub
```

The following C# sample displays the caption from the cell being clicked:

```
private void axG2antt1_MouseDownEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_MouseDownEvent e)
{
  int c = 0;
  EXG2ANTTLib.HitTestInfoEnum hit;
  int i = axG2antt1.get_ItemFromPoint( e.x, e.y, out c,out hit );
  if ( ( i != 0 ) || ( c != 0 ) )
  {
    string s = axG2antt1.Items.get_CellValue( i,c ).ToString();
    s = "Cell: " + s + ", Hit: " + hit.ToString();
    System.Diagnostics.Debug.WriteLine( s );
  }
}
```

The following VFP sample displays the caption from the cell being clicked:

```
*** ActiveX Control Event ***
```

```
LPARAMETERS button, shift, x, y

local c, hit
c = 0
hit = 0
with thisform.G2antt1
   .Items.DefaultItem = .ItemFromPoint( x, y, @c, @hit )
   if ( .Items.DefaultItem <> 0 ) or ( c <> 0 )
      wait window nowait .Items.CellValue( 0, c ) + " " + Str( hit )
   endif
endwith
```

The following VB sample displays the start data of the bar from the point:

```
Private Sub G2antt1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As
Single)
   With G2antt1
      Dim h As HITEM, c As Long, hit As HitTestInfoEnum
      h = .ItemFromPoint(-1, -1, c, hit)
      If Not (h = 0) Then
         Dim k As Variant
         k = .Chart.BarFromPoint(-1, -1)
         If Not IsEmpty(k) Then
            Debug.Print .Items.ItemBar(h, k, exBarStart)
         End If
      End If
   End With
End Sub
```

The following C++ sample displays the start data of the bar from the point:

```
#include "Items.h"
#include "Chart.h"

CString V2Date( VARIANT* pvtValue )
{
   COleVariant vtDate;
   vtDate.ChangeType( VT_BSTR, pvtValue );
```

```
    return V_BSTR( &vtDate );
}

void OnMouseDownG2antt1(short Button, short Shift, long X, long Y)
{
    long c = 0, hit = 0, h = m_g2antt.GetItemFromPoint( -1, -1, &c, &hit );
    if ( h != 0 )
    {
        COleVariant vtKey = m_g2antt.GetChart().GetBarFromPoint( -1, -1 );
        if ( V_VT( &vtKey ) != VT_EMPTY )
        {
            COleVariant vtStart = m_g2antt.GetItems().GetItemBar( h, vtKey, 1 /*exBarStart*/ );
            OutputDebugString( V2Date( &vtStart ) );
        }
    }
}
```

The following VB.NET sample displays the start data of the bar from the point:

```
Private Sub AxG2antt1_MouseDownEvent(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_MouseDownEvent) Handles
AxG2antt1.MouseDownEvent
    With AxG2antt1
        Dim c As Long, hit As EXG2ANTTLib.HitTestInfoEnum, h As Integer =
.get_ItemFromPoint(-1, -1, c, hit)
        If Not (h = 0) Then
            Dim k As Object
            k = .Chart.BarFromPoint(-1, -1)
            If Not k Is Nothing Then
                System.Diagnostics.Debug.WriteLine(.Items.ItemBar(h, k,
EXG2ANTTLib.ItemBarPropertyEnum.exBarStart))
            End If
        End If
    End With
End Sub
```

The following C# sample displays the start data of the bar from the point:

```
private void axG2antt1_MouseDownEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_MouseDownEvent e)
{
   int c = 0;
   EXG2ANTTLib.HitTestInfoEnum hit = EXG2ANTTLib.HitTestInfoEnum.exHTCell;
   int h = axG2antt1.get_ItemFromPoint(-1, -1, out c, out hit);
   if (h != 0)
   {
      object k = axG2antt1.Chart.get_BarFromPoint(-1, -1);
      if (k != null)
         System.Diagnostics.Debug.WriteLine( axG2antt1.Items.get_ItemBar( h, k,
EXG2ANTTLib.ItemBarPropertyEnum.exBarStart ) );
   }
}
```

The following VFP sample displays the start data of the bar from the point:

```
*** ActiveX Control Event ***
LPARAMETERS button, shift, x, y

With thisform.G2antt1
   local h, c, hit
   h = .ItemFromPoint(-1, -1, c, hit)
   If (h # 0) Then
      local k
      k = .Chart.BarFromPoint(-1, -1)
      If !Empty(k) Then
         ? .Items.ItemBar(h, k, 1)
      EndIf
   EndIf
EndWith
```

event MouseMove (Button as Integer, Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occurs when the user moves the mouse.

Type	Description
Button as Integer	Gets which mouse button was pressed as 1 for Left Mouse Button, 2 for Right Mouse Button and 4 for Middle Mouse Button.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in container coordinates.

The MouseMove event is generated continually as the mouse pointer moves across objects. Unless another object has captured the mouse, an object recognizes a MouseMove event whenever the mouse position is within its borders. Use the ItemFromPoint property to get the item from cursor. Use the ColumnFromPoint property to get the column from point. Use the DateFromPoint property to specify the date from the cursor. Use the BarFromPoint property to get the bar from the point. Use the LinkFromPoint property to get the link from the point. Use the LevelFromPoint property to retrieve the index of the level from the cursor. Use the AnchorFromPoint property to retrieve the identifier of the anchor element from the cursor. The NoteFromPoint property retrieves the note/box from the cursor. The TimeZoneFromPoint property retrieves the key of the time-zone from the cursor.

Almost all properties that get an object from point supports -1,-1 coordinate that specifies the current cursor position, so no conversion is required for X and Y coordinates.

Use the DrawDateTicker property to draw a ticker as cursor hovers the chart's area. Use the Background(exHoverColumn) property to change the visual appearance of the column's header when the cursor hovers it.

Syntax for MouseMove event, **/NET** version, on:

C#

```
private void MouseMoveEvent(object sender,short Button,short Shift,int X,int Y)
{
}
```

VB

```
Private Sub MouseMoveEvent(ByVal sender As System.Object,ByVal Button As Short,ByVal Shift As Short,ByVal X As Integer,ByVal Y As Integer) Handles MouseMoveEvent
End Sub
```

Syntax for MouseMove event, **/COM** version, on:

C#

```
private void MouseMoveEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_MouseMoveEvent e)
{
}
```

C++

```
void OnMouseMove(short Button,short Shift,long X,long Y)
{
}
```

C++ Builder

```
void __fastcall MouseMove(TObject *Sender,short Button,short Shift,int X,int Y)
{
}
```

Delphi

```
procedure MouseMove(ASender: TObject; Button : Smallint;Shift : Smallint;X : Integer;Y : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure MouseMoveEvent(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_MouseMoveEvent);
begin
end;
```

Powe…

```
begin event MouseMove(integer Button,integer Shift,long X,long Y)
end event MouseMove
```

VB.NET

```
Private Sub MouseMoveEvent(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_MouseMoveEvent) Handles MouseMoveEvent
End Sub
```

VB6

```
Private Sub MouseMove(Button As Integer,Shift As Integer,X As Single,Y As Single)
End Sub
```

VBA

```
Private Sub MouseMove(ByVal Button As Integer,ByVal Shift As Integer,ByVal X As Long,ByVal Y As Long)
End Sub
```

VFP

```
LPARAMETERS Button,Shift,X,Y
```

Xbas…

```
PROCEDURE OnMouseMove(oG2antt,Button,Shift,X,Y)
RETURN
```

Syntax for MouseMove event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="MouseMove(Button,Shift,X,Y)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function MouseMove(Button,Shift,X,Y)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComMouseMove Short llButton Short llShift OLE_XPOS_PIXELS llX OLE_YPOS_PIXELS llY
    Forward Send OnComMouseMove llButton llShift llX llY
End_Procedure
```

Visual Objects

```
METHOD OCX_MouseMove(Button,Shift,X,Y) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_MouseMove(int _Button,int _Shift,int _X,int _Y)
{
}
```

XBasic

```
function MouseMove as v (Button as N,Shift as N,X as OLE::Exontrol.G2antt.1::OLE_XPOS_PIXELS,Y as
```

```
OLE::Exontrol.G2antt.1::OLE_YPOS_PIXELS)
end function
```

dBASE

```
function nativeObject_MouseMove(Button,Shift,X,Y)
return
```

The following Access sample prints the cell's caption from the cursor:

```
Private Sub G2antt1_MouseMove(ByVal Button As Integer, ByVal Shift As Integer, ByVal X
As Long, ByVal Y As Long)
    Dim h As HITEM
    Dim c As Long, hit As Long
    h = G2antt1.ItemFromPoint(-1, -1, c, hit)
    If Not (h = 0) Then
        Debug.Print "The """ & G2antt1.Items.CellValue(h, c) & """ cell has been double
clicked."
    End If
End Sub
```

The following VB sample prints the cell's caption from the cursor (if the control contains no inner cells. Use the SplitCell property to insert inner cells) :

```
Private Sub G2antt1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As
Single)
    On Error Resume Next
    ' Converts the container coordinates to client coordinates
    X = X / Screen.TwipsPerPixelX
    Y = Y / Screen.TwipsPerPixelY
    Dim h As HITEM
    Dim c As Long
    Dim hit As EXG2ANTTLibCtl.HitTestInfoEnum
    ' Gets the item from (X,Y)
    h = G2antt1.ItemFromPoint(X, Y, c, hit)
    If Not (h = 0) Then
        Debug.Print G2antt1.Items.CellValue(h, c) & " HT = " & hit
    End If
End Sub
```

The following VB sample displays the cell's caption from the cursor (if the control contains inner cells):

```
Private Sub G2antt1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    On Error Resume Next
    ' Converts the container coordinates to client coordinates
    X = X / Screen.TwipsPerPixelX
    Y = Y / Screen.TwipsPerPixelY
    Dim h As HITEM
    Dim c As Long
    Dim hit As EXG2ANTTLibCtl.HitTestInfoEnum
    ' Gets the item from (X,Y)
    h = G2antt1.ItemFromPoint(X, Y, c, hit)
    If Not (h = 0) Or Not (c = 0) Then
        Debug.Print G2antt1.Items.CellValue(h, c) & " HT = " & hit
    End If
End Sub
```

The following VB sample displays the date from the cursor:

```
Private Sub G2antt1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    With G2antt1.Chart
        Dim d As Date
        d = .DateFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
        Debug.Print .FormatDate(d, "<%m%>/<%d%>/<%yyyy%>")
    End With
End Sub
```

The following C++ sample displays the cell's from the point:

```
#include "Items.h"

static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
    if ( pv )
    {
```

```
        if ( pv->vt == VT_ERROR )
            return szDefault;

        COleVariant vt;
        vt.ChangeType( VT_BSTR, pv );
        return V_BSTR( &vt );
    }
    return szDefault;
}

void OnMouseMoveG2antt1(short Button, short Shift, long X, long Y)
{
    long c = 0, hit = 0, hItem = m_g2antt.GetItemFromPoint( X, Y, &c, &hit );
    if ( ( hItem != 0 ) || ( c != 0 ) )
    {
        CItems items = m_g2antt.GetItems();
        COleVariant vtItem( hItem ), vtColumn( c );
        CString strCaption = V2S( &items.GetCellValue( vtItem, vtColumn ) ), strOutput;
        strOutput.Format( "Cell: '%s', Hit = %08X\n", strCaption, hit );
        OutputDebugString( strOutput );
    }
}
```

The following C++ sample displays the date from the point:

```
void OnMouseMoveG2antt1(short Button, short Shift, long X, long Y)
{
    CChart chart = m_g2antt.GetChart();
    DATE d = chart.GetDateFromPoint( X, Y );
    CString strFormat = chart.GetFormatDate( d, "<%m%>/<%d%>/<%yyyy%>" );
    OutputDebugString( strFormat );
}
```

The following VB.NET sample displays the cell's from the point:

```
Private Sub AxG2antt1_MouseMoveEvent(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_MouseMoveEvent) Handles AxG2antt1.MouseMoveEvent
    With AxG2antt1
```

```
    Dim i As Integer, c As Integer, hit As EXG2ANTTLib.HitTestInfoEnum
    i = .get_ItemFromPoint(e.x, e.y, c, hit)
    If (Not (i = 0) Or Not (c = 0)) Then
      Debug.WriteLine("Cell: " & .Items.CellValue(i, c) & " Hit: " & hit.ToString())
    End If
  End With
End Sub
```

The following VB.NET sample displays the date from the point:

```
Private Sub AxG2antt1_MouseMoveEvent(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_MouseMoveEvent) Handles AxG2antt1.MouseMoveEvent
  With AxG2antt1.Chart
    Dim d As Date
    d = .DateFromPoint(e.x, e.y)
    Debug.Write(.FormatDate(d, "<%m%>/<%d%>/<%yyyy%>"))
  End With
End Sub
```

The following C# sample displays the cell's from the point:

```
private void axG2antt1_MouseMoveEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_MouseMoveEvent e)
{
  int c = 0;
  EXG2ANTTLib.HitTestInfoEnum hit;
  int i = axG2antt1.get_ItemFromPoint( e.x, e.y, out c,out hit );
  if ( ( i != 0 ) || ( c != 0 ) )
  {
    object cap = axG2antt1.Items.get_CellValue(i, c);
    string s = cap != null ? cap.ToString() : "";
    s = "Cell: " + s + ", Hit: " + hit.ToString();
    System.Diagnostics.Debug.WriteLine(s);
  }
}
```

The following C# sample displays the date from the point:

```
private void axG2antt1_MouseMoveEvent(object sender,
```

```
AxEXG2ANTTLib._IG2anttEvents_MouseMoveEvent e)
{
   DateTime d = axG2antt1.Chart.get_DateFromPoint(e.x, e.y);
   System.Diagnostics.Debug.Write(axG2antt1.Chart.get_FormatDate(d, "
<%m%>/<%d%>/<%yyyy%>"));
}
```

The following VFP sample displays the cell's from the point:

```
*** ActiveX Control Event ***
LPARAMETERS button, shift, x, y

local c, hit
c = 0
hit = 0
with thisform.G2antt1
   .Items.DefaultItem = .ItemFromPoint( x, y, @c, @hit )
   if ( .Items.DefaultItem <> 0 ) or ( c <> 0 )
      wait window nowait .Items.CellValue( 0, c ) + " " + Str( hit )
   endif
endwith
```

The following VFP sample displays the date from the point:

```
*** ActiveX Control Event ***
LPARAMETERS button, shift, x, y

with thisform.G2antt1.Chart
   d = .DateFromPoint(x,y)
   wait window nowait .FormatDate(d, "<%m%>/<%d%>/<%yyyy%>" )
endwith
```

event MouseUp (Button as Integer, Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occurs when the user releases a mouse button.

Type	Description
Button as Integer	An integer that identifies the button that was pressed to cause the event as as 1 for Left Mouse Button, 2 for Right Mouse Button and 4 for Middle Mouse Button.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the button argument is pressed or released.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in container coordinates.

Use a MouseDown or MouseUp event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the Click and DblClick events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers. Use the ItemFromPoint property to get the item from point. Use the ColumnFromPoint property to get the column from point. Use the DateFromPoint property to specify the date from the cursor. The NoteFromPoint property retrieves the note/box from the cursor. Use the AnchorFromPoint property to retrieve the identifier of the anchor element from the cursor. *Almost all properties that get an object from point supports -1,-1 coordinate that specifies the current cursor position, so no conversion is required for X and Y coordinates.*

Syntax for MouseUp event, **/NET** version, on:

C#

```
private void MouseUpEvent(object sender,short Button,short Shift,int X,int Y)
{
}
```

VB

```
Private Sub MouseUpEvent(ByVal sender As System.Object,ByVal Button As
Short,ByVal Shift As Short,ByVal X As Integer,ByVal Y As Integer) Handles
MouseUpEvent
End Sub
```

Syntax for MouseUp event, **/COM** version, on:

C#
```
private void MouseUpEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_MouseUpEvent e)
{
}
```

C++
```
void OnMouseUp(short Button,short Shift,long X,long Y)
{
}
```

C++ Builder
```
void __fastcall MouseUp(TObject *Sender,short Button,short Shift,int X,int Y)
{
}
```

Delphi
```
procedure MouseUp(ASender: TObject; Button : Smallint;Shift : Smallint;X :
Integer;Y : Integer);
begin
end;
```

Delphi 8 (.NET only)
```
procedure MouseUpEvent(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_MouseUpEvent);
begin
end;
```

Powe…
```
begin event MouseUp(integer Button,integer Shift,long X,long Y)
end event MouseUp
```

VB.NET
```
Private Sub MouseUpEvent(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_MouseUpEvent) Handles MouseUpEvent
End Sub
```

VB6
```
Private Sub MouseUp(Button As Integer,Shift As Integer,X As Single,Y As Single)
End Sub
```

VBA
```
Private Sub MouseUp(ByVal Button As Integer,ByVal Shift As Integer,ByVal X As
Long,ByVal Y As Long)
End Sub
```

VFP
```
LPARAMETERS Button,Shift,X,Y
```

Xbas…
```
PROCEDURE OnMouseUp(oG2antt,Button,Shift,X,Y)
RETURN
```

Syntax for MouseUp event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="MouseUp(Button,Shift,X,Y)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function MouseUp(Button,Shift,X,Y)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComMouseUp Short llButton Short llShift OLE_XPOS_PIXELS llX
OLE_YPOS_PIXELS llY
    Forward Send OnComMouseUp llButton llShift llX llY
End_Procedure
```

Visual Objects
```
METHOD OCX_MouseUp(Button,Shift,X,Y) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_MouseUp(int _Button,int _Shift,int _X,int _Y)
{
}
```

XBasic
```
function MouseUp as v (Button as N,Shift as N,X as
OLE::Exontrol.G2antt.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.G2antt.1::OLE_YPOS_PIXELS)
end function
```

dBASE
```
function nativeObject_MouseUp(Button,Shift,X,Y)
return
```

The following Access sample prints the cell's caption where the mouse has been released:

```
Private Sub G2antt1_MouseUp(ByVal Button As Integer, ByVal Shift As Integer, ByVal X As Long, ByVal Y As Long)
    Dim h As HITEM
    Dim c As Long, hit As Long
    h = G2antt1.ItemFromPoint(-1, -1, c, hit)
    If Not (h = 0) Then
        MsgBox "The """ & G2antt1.Items.CellValue(h, c) & """ cell has been double clicked."
    End If
End Sub
```

The following VB sample prints the cell's caption where the mouse has been released:

```
Private Sub G2antt1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    ' Converts the container coordinates to client coordinates
    X = X / Screen.TwipsPerPixelX
    Y = Y / Screen.TwipsPerPixelY
    Dim h As HITEM
    Dim c As Long, hit as Long
    ' Gets the item from (X,Y)
    h = G2antt1.ItemFromPoint(X, Y, c, hit)
    If Not (h = 0) Then
        Debug.Print G2antt1.Items.CellValue(h, c)
    End If
End Sub
```

If you need to add a context menu based on the item you can use the MouseUp event, like in the following VB sample (the sample uses the Exontrol's ExPopupMenu Component):

```
Private Sub G2antt1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If (Button = 2) Then
        ' Converts the container coordinates to client coordinates
        X = X / Screen.TwipsPerPixelX
        Y = Y / Screen.TwipsPerPixelY
        Dim h As HITEM
        Dim c As Long, hit as Long
        ' Gets the item from (X,Y)
```

```
      h = G2antt1.ItemFromPoint(X, Y, c, hit)
      If Not (h = 0) Then
         Dim i As Long
         PopupMenu1.Items.Add G2antt1.Items.CellValue(h, c)
         i = PopupMenu1.ShowAtCursor
      End If
   End If
End Sub
```

The following VC sample displays the caption of the cell where the mouse is released:

```
#include "Items.h"

static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
   if ( pv )
   {
      if ( pv->vt == VT_ERROR )
         return szDefault;

      COleVariant vt;
      vt.ChangeType( VT_BSTR, pv );
      return V_BSTR( &vt );
   }
   return szDefault;
}

void OnMouseUpG2antt1(short Button, short Shift, long X, long Y)
{
   long c = 0, hit = 0, hItem = m_g2antt.GetItemFromPoint( X, Y, &c, &hit );
   if ( ( hItem != 0 ) || ( c != 0 ) )
   {
      CItems items = m_g2antt.GetItems();
      COleVariant vtItem( hItem ), vtColumn( c );
      CString strCaption = V2S( &items.GetCellValue( vtItem, vtColumn ) ), strOutput;
      strOutput.Format( "Cell: '%s', Hit = %08X\n", strCaption, hit );
      OutputDebugString( strOutput );
```

```
    }
}
```

The following VB.NET sample displays the caption of the cell where the mouse is released:

```
Private Sub AxG2antt1_MouseUpEvent(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_MouseUpEvent) Handles AxG2antt1.MouseUpEvent
    With AxG2antt1
        Dim i As Integer, c As Integer, hit As EXG2ANTTLib.HitTestInfoEnum
        i = .get_ItemFromPoint(e.x, e.y, c, hit)
        If (Not (i = 0) Or Not (c = 0)) Then
            Debug.WriteLine("Cell: " & .Items.CellValue(i, c) & " Hit: " & hit.ToString())
        End If
    End With
End Sub
```

The following C# sample displays the caption of the cell where the mouse is released:

```
private void axG2antt1_MouseUpEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_MouseUpEvent e)
{
    int c = 0;
    EXG2ANTTLib.HitTestInfoEnum hit;
    int i = axG2antt1.get_ItemFromPoint( e.x, e.y, out c,out hit );
    if ( ( i != 0 ) || ( c != 0 ) )
    {
        string s = axG2antt1.Items.get_CellValue( i,c ).ToString();
        s = "Cell: " + s + ", Hit: " + hit.ToString();
        System.Diagnostics.Debug.WriteLine( s );
    }
}
```

The following VFP sample displays the caption of the cell where the mouse is released:

```
*** ActiveX Control Event ***
LPARAMETERS button, shift, x, y

local c, hit
c = 0
```

```
hit = 0
with thisform.G2antt1
   .Items.DefaultItem = .ItemFromPoint( x, y, @c, @hit )
   if ( .Items.DefaultItem <> 0 ) or ( c <> 0 )
      wait window nowait .Items.CellValue( 0, c ) + " " + Str( hit )
   endif
endwith
```

event OffsetChanged (Horizontal as Boolean, NewVal as Long)

Occurs when the scroll position has been changed.

Type	Description
Horizontal as Boolean	A boolean expression that indicates whether the horizontal scroll bar has changed.
NewVal as Long	A long value that indicates the new scroll bar value in pixels.

If the control has no scroll bars the OffsetChanged and OversizeChanged events are not fired. Use the ScrollBars property of the control to determine which scroll bars are visible within the control. The OffsetChanged event is not fired when the user scrolls the chart's part of the control. In this case, the DateChange event is fired. The OffseyChanged event is fired only when the user scrolls horizontally the columns section of the control, or when the user scrolls vertically the items part of the control (including the chart part). The ScrollPos property can be used to programmatically scroll the control's content at giving position.

Syntax for OffsetChanged event, **/NET** version, on:

C#
```
private void OffsetChanged(object sender,bool Horizontal,int NewVal)
{
}
```

VB
```
Private Sub OffsetChanged(ByVal sender As System.Object,ByVal Horizontal As
Boolean,ByVal NewVal As Integer) Handles OffsetChanged
End Sub
```

Syntax for OffsetChanged event, **/COM** version, on:

C#
```
private void OffsetChanged(object sender,
AxEXG2ANTTLib._IG2anttEvents_OffsetChangedEvent e)
{
}
```

C++
```
void OnOffsetChanged(BOOL Horizontal,long NewVal)
{
}
```

C++ Builder
```
void __fastcall OffsetChanged(TObject *Sender,VARIANT_BOOL Horizontal,long
```

```
NewVal)
{
}
```

Delphi

```
procedure OffsetChanged(ASender: TObject; Horizontal : WordBool;NewVal :
Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure OffsetChanged(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_OffsetChangedEvent);
begin
end;
```

Powe…

```
begin event OffsetChanged(boolean Horizontal,long NewVal)
end event OffsetChanged
```

VB.NET

```
Private Sub OffsetChanged(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_OffsetChangedEvent) Handles OffsetChanged
End Sub
```

VB6

```
Private Sub OffsetChanged(ByVal Horizontal As Boolean,ByVal NewVal As Long)
End Sub
```

VBA

```
Private Sub OffsetChanged(ByVal Horizontal As Boolean,ByVal NewVal As Long)
End Sub
```

VFP

```
LPARAMETERS Horizontal,NewVal

```

Xbas…

```
PROCEDURE OnOffsetChanged(oG2antt,Horizontal,NewVal)
RETURN
```

Syntax for OffsetChanged event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="OffsetChanged(Horizontal,NewVal)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function OffsetChanged(Horizontal,NewVal)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComOffsetChanged Boolean llHorizontal Integer llNewVal
    Forward Send OnComOffsetChanged llHorizontal llNewVal
End_Procedure
```

Visual Objects

```
METHOD OCX_OffsetChanged(Horizontal,NewVal) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_OffsetChanged(boolean _Horizontal,int _NewVal)
{
}
```

XBasic

```
function OffsetChanged as v (Horizontal as L,NewVal as N)
end function
```

dBASE

```
function nativeObject_OffsetChanged(Horizontal,NewVal)
return
```

The following VB sample displays the new scroll position when user scrolls horizontally the control:

```
Private Sub G2antt1_OffsetChanged(ByVal Horizontal As Boolean, ByVal NewVal As Long)
    If (Horizontal) Then
        Debug.Print "The horizontal scroll bar has been moved to " & NewVal
    End If
End Sub
```

The following VC sample displays the new scroll position when the user scrolls vertically the control:

```
void OnOffsetChangedG2antt1(BOOL Horizontal, long NewVal)
{
    if ( !Horizontal )
```

```
    {
        CString strFormat;
        strFormat.Format( "NewPos = %i\n", NewVal );
        OutputDebugString( strFormat );
    }
}
```

The following VB.NET sample displays the new scroll position when the user scrolls vertically the control:

```
Private Sub AxG2antt1_OffsetChanged(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_OffsetChangedEvent) Handles AxG2antt1.OffsetChanged
    If (Not e.horizontal) Then
        Debug.WriteLine(e.newVal)
    End If
End Sub
```

The following C# sample displays the new scroll position when the user scrolls vertically the control:

```
private void axG2antt1_OffsetChanged(object sender,
AxEXG2ANTTLib._IG2anttEvents_OffsetChangedEvent e)
{
    if ( !e.horizontal )
        System.Diagnostics.Debug.WriteLine(e.newVal);
}
```

The following VFP sample displays the new scroll position when the user scrolls vertically the control:

```
*** ActiveX Control Event ***
LPARAMETERS horizontal, newval

if ( 0 # horizontal )
    wait window nowait str( newval )
endif
```

event OLECompleteDrag (Effect as Long)

Occurs when a source component is dropped onto a target component, informing the source component that a drag action was either performed or canceled

Type	Description
Effect as Long	A long set by the source object identifying the action that has been performed, thus allowing the source to take appropriate action if the component was moved (such as the source deleting data if it is moved from one component to another

The OLECompleteDrag event is the final event to be called in an OLE drag/drop operation. This event informs the source component of the action that was performed when the object was dropped onto the target component. The target sets this value through the effect parameter of the OLEDragDrop event. Based on this, the source can then determine the appropriate action it needs to take. For example, if the object was moved into the target (exDropEffectMove), the source needs to delete the object from itself after the move. The control supports only manual OLE drag and drop events. In order to enable OLE drag and drop feature into control you have to set the OLEDropMode and OLEDrag properties.

The settings for Effect are:

- exOLEDropEffectNone (0), Drop target cannot accept the data, or the drop operation was cancelled
- exOLEDropEffectCopy (1), Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
- exOLEDropEffectMove (2), Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

Syntax for OLECompleteDrag event, **/NET** version, on:

C#
```
// OLECompleteDrag event is not supported. Use the
DragEnter,DragLeave,DragOver, DragDrop ... events.
```

VB
```
// OLECompleteDrag event is not supported. Use the
DragEnter,DragLeave,DragOver, DragDrop ... events.
```

Syntax for OLECompleteDrag event, **/COM** version, on:

C#
```
private void OLECompleteDrag(object sender,
```

```
AxEXG2ANTTLib._IG2anttEvents_OLECompleteDragEvent e)
{
}
```

C++

```
void OnOLECompleteDrag(long Effect)
{
}
```

C++ Builder

```
void __fastcall OLECompleteDrag(TObject *Sender,long Effect)
{
}
```

Delphi

```
procedure OLECompleteDrag(ASender: TObject; Effect : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure OLECompleteDrag(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_OLECompleteDragEvent);
begin
end;
```

Powe…

```
begin event OLECompleteDrag(long Effect)
end event OLECompleteDrag
```

VB.NET

```
Private Sub OLECompleteDrag(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_OLECompleteDragEvent) Handles
OLECompleteDrag
End Sub
```

VB6

```
Private Sub OLECompleteDrag(ByVal Effect As Long)
End Sub
```

VBA

```
Private Sub OLECompleteDrag(ByVal Effect As Long)
End Sub
```

VFP

```
LPARAMETERS Effect

```

Xbas…

```
PROCEDURE OnOLECompleteDrag(oG2antt,Effect)
RETURN
```

Syntax for OLECompleteDrag event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="OLECompleteDrag(Effect)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function OLECompleteDrag(Effect)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComOLECompleteDrag Integer llEffect
    Forward Send OnComOLECompleteDrag llEffect
End_Procedure
```

Visual Objects

```
METHOD OCX_OLECompleteDrag(Effect) CLASS MainDialog
RETURN NIL
```

X++

```
// OLECompleteDrag event is not supported. Use the
DragEnter,DragLeave,DragOver, DragDrop ... events.
```

XBasic

```
function OLECompleteDrag as v (Effect as N)
end function
```

dBASE

```
function nativeObject_OLECompleteDrag(Effect)
return
```

event OLEDragDrop (Data as ExDataObject, Effect as Long, Button as Integer, Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occurs when a source component is dropped onto a target component when the source component determines that a drop can occur.

Type	Description
Data as ExDataObject	An ExDataObject object containing formats that the source will provide and, in addition, possibly the data for those formats. If no data is contained in the ExDataObject, it is provided when the control calls the GetData method. The SetData and Clear methods cannot be used here.
Effect as Long	A Long set by the target component identifying the action that has been performed (if any), thus allowing the source to take appropriate action if the component was moved (such as the source deleting the data). The possible values are listed in Remarks.
Button as Integer	An integer which acts as a bit field corresponding to the state of a mouse button when it is depressed. The left button is bit 0, the right button is bit 1, and the middle button is bit 2. These bits correspond to the values 1, 2, and 4, respectively. It indicates the state of the mouse buttons; some, all, or none of these three bits can be set, indicating that some, all, or none of the buttons are depressed.
Shift as Integer	An integer which acts as a bit field corresponding to the state of the SHIFT, CTRL, and ALT keys when they are depressed. The SHIFT key is bit 0, the CTRL key is bit 1, and the ALT key is bit 2. These bits correspond to the values 1, 2, and 4, respectively. The shift parameter indicates the state of these keys; some, all, or none of the bits can be set, indicating that some, all, or none of the keys are depressed. For example, if both the CTRL and ALT keys were depressed, the value of shift would be 6.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The X value is always expressed in container coordinates
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The Y value is always expressed in container coordinates.

In the /NET Assembly, you have to use the DragDrop event as explained here:

- *https://www.exontrol.com/sg.jsp?content=support/faq/net/#dragdrop*

The OLEDragDrop event is fired when the user has dropped files or clipboard information into the control. Use the OLEDropMode property on exOLEDropManual to enable OLE drop and drop support. Use the ItemFromPoint property to get the item from point. Use the ColumnFromPoint property to get the column from point. Use the AddItem method to add a new item to the control. Use the InsertItem method to insert a new child item. Use the ItemPosition property to specify the item's position.

The settings for Effect are:

- exOLEDropEffectNone (0), Drop target cannot accept the data, or the drop operation was cancelled
- exOLEDropEffectCopy (1), Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
- exOLEDropEffectMove (2), Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

Syntax for OLEDragDrop event, **/NET** version, on:

C#
```
// OLEDragDrop event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop ... events.
```

VB
```
// OLEDragDrop event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop ... events.
```

Syntax for OLEDragDrop event, **/COM** version, on:

C#
```
private void OLEDragDrop(object sender, AxEXG2ANTTLib._IG2anttEvents_OLEDragDropEvent e)
{
}
```

C++
```
void OnOLEDragDrop(LPDISPATCH Data,long FAR* Effect,short Button,short Shift,long X,long Y)
{
}
```

C++ Builder

```
void __fastcall OLEDragDrop(TObject *Sender,Exg2anttlib_tlb::IExDataObject *Data,long *
Effect,short Button,short Shift,int X,int Y)
{
}
```

Delphi

```
procedure OLEDragDrop(ASender: TObject; Data : IExDataObject;var Effect :
Integer;Button : Smallint;Shift : Smallint;X : Integer;Y : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure OLEDragDrop(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_OLEDragDropEvent);
begin
end;
```

Powe…

```
begin event OLEDragDrop(oleobject Data,long Effect,integer Button,integer
Shift,long X,long Y)
end event OLEDragDrop
```

VB.NET

```
Private Sub OLEDragDrop(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_OLEDragDropEvent) Handles OLEDragDrop
End Sub
```

VB6

```
Private Sub OLEDragDrop(ByVal Data As EXG2ANTTLibCtl.IExDataObject,Effect As
Long,ByVal Button As Integer,ByVal Shift As Integer,ByVal X As Single,ByVal Y As
Single)
End Sub
```

VBA

```
Private Sub OLEDragDrop(ByVal Data As Object,Effect As Long,ByVal Button As
Integer,ByVal Shift As Integer,ByVal X As Long,ByVal Y As Long)
End Sub
```

VFP

```
LPARAMETERS Data,Effect,Button,Shift,X,Y
```

Xbas…

```
PROCEDURE OnOLEDragDrop(oG2antt,Data,Effect,Button,Shift,X,Y)
```

```
RETURN
```

Syntax for OLEDragDrop event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="OLEDragDrop(Data,Effect,Button,Shift,X,Y)"
LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function OLEDragDrop(Data,Effect,Button,Shift,X,Y)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComOLEDragDrop Variant llData Integer llEffect Short llButton
Short llShift OLE_XPOS_PIXELS llX OLE_YPOS_PIXELS llY
    Forward Send OnComOLEDragDrop llData llEffect llButton llShift llX llY
End_Procedure
```

Visual Objects
```
METHOD OCX_OLEDragDrop(Data,Effect,Button,Shift,X,Y) CLASS MainDialog
RETURN NIL
```

X++
```
// OLEDragDrop event is not supported. Use the DragEnter,DragLeave,DragOver,
DragDrop ... events.
```

XBasic
```
function OLEDragDrop as v (Data as OLE::Exontrol.G2antt.1::IExDataObject,Effect
as N,Button as N,Shift as N,X as OLE::Exontrol.G2antt.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.G2antt.1::OLE_YPOS_PIXELS)
end function
```

dBASE
```
function nativeObject_OLEDragDrop(Data,Effect,Button,Shift,X,Y)
return
```

The following VB sample adds a new item when the user drags a file (Open the Windows Explorer, click and drag a file to the control) :

```
Private Sub G2antt1_OLEDragDrop(Index As Integer, ByVal Data As
EXG2ANTTLibCtl.IExDataObject, Effect As Long, ByVal Button As Integer, ByVal Shift As
```

```
Integer, ByVal X As Single, ByVal Y As Single)
   If Data.GetFormat(exCFFiles) Then
      Data.GetData (exCFFiles)
      Dim strFile As String
      strFile = Data.Files(0)
      'Adds a new item to the control
      G2antt1(Index).Visible = False
      With G2antt1(Index)
      .BeginUpdate
         Dim i As HITEM
         i = .Items.AddItem(strFile)
         .Items.EnsureVisibleItem i
      .EndUpdate
      End With
      G2antt1(Index).Visible = True
   End If
End Sub
```

The following VC sample inserts a child item for each file that user drags:

```
#import <exg2antt.dll> rename( "GetItems", "exGetItems" )

#include "Items.h"
void OnOLEDragDropG2antt1(LPDISPATCH Data, long FAR* Effect, short Button, short
Shift, long X, long Y)
{
   EXG2ANTTLib::IExDataObjectPtr spData( Data );
   if ( spData != NULL )
      if ( spData->GetFormat( EXG2ANTTLib::exCFFiles ) )
      {
         CItems items = m_g2antt.GetItems();
         // Gets the handle of the item where the files will be inserted
         long c = 0, h = 0, nParentItem = m_g2antt.GetItemFromPoint( X, Y, &c, &h );
         if ( nParentItem == 0 )
            if ( c != 0 )
               nParentItem = items.GetCellItem( c );
         EXG2ANTTLib::IExDataObjectFilesPtr spFiles( spData->Files );
```

```
        if ( spFiles->Count > 0 )
        {
            m_g2antt.BeginUpdate();
            COleVariant vtMissing; vtMissing.vt = VT_ERROR;
            for ( long i = 0; i < spFiles->Count; i++ )
                items.InsertItem( nParentItem, vtMissing, COleVariant( spFiles->GetItem( i
).operator const char *() ) );
            if ( nParentItem )
                items.SetExpandItem( nParentItem, TRUE );
            m_g2antt.EndUpdate();
        }

    }
}
```

The #import statement imports definition for the ExDataObject and ExDataObjectFiles objects. If the exg2antt.dll file is located in another folder than the system folder, the path to the file must be specified. The sample gets the item where the files were dragged and insert all files in that position, as child items, if case.

The following VB.NET sample inserts a child item for each file that user drags:

```
Private Sub AxG2antt1_OLEDragDrop(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_OLEDragDropEvent) Handles AxG2antt1.OLEDragDrop
    If e.data.GetFormat(EXG2ANTTLib.exClipboardFormatEnum.exCFFiles) Then
        If (e.data.Files.Count > 0) Then
            AxG2antt1.BeginUpdate()
            With AxG2antt1.Items
                Dim iParent As Integer, c As Integer, hit As EXG2ANTTLib.HitTestInfoEnum
                iParent = AxG2antt1.get_ItemFromPoint(e.x, e.y, c, hit)
                If iParent = 0 Then
                    If Not c = 0 Then
                        iParent = .CellItem(c)
                    End If
                End If
                Dim i As Long
                For i = 0 To e.data.Files.Count - 1
                    .InsertItem(iParent, , e.data.Files(i))
```

```
            Next
            If Not (iParent = 0) Then
               .ExpandItem(iParent) = True
            End If
         End With
         AxG2antt1.EndUpdate()
      End If
   End If
End Sub
```

The following C# sample inserts a child item for each file that user drags:

```
private void axG2antt1_OLEDragDrop(object sender,
AxEXG2ANTTLib._IG2anttEvents_OLEDragDropEvent e)
{
   if ( e.data.GetFormat(
Convert.ToInt16(EXG2ANTTLib.exClipboardFormatEnum.exCFFiles) ) )
      if ( e.data.Files.Count > 0 )
      {
         EXG2ANTTLib.HitTestInfoEnum hit;
         int c = 0, iParent = axG2antt1.get_ItemFromPoint( e.x, e.y, out c, out hit );
         if ( iParent == 0 )
            if ( c != 0 )
               iParent = axG2antt1.Items.get_CellItem( c );

         axG2antt1.BeginUpdate();
         for ( int i = 0; i < e.data.Files.Count; i++ )
            axG2antt1.Items.InsertItem( iParent,"", e.data.Files[i].ToString() );
         if ( iParent != 0 )
            axG2antt1.Items.set_ExpandItem( iParent, true );
         axG2antt1.EndUpdate();
      }
}
```

The following VFP sample inserts a child item for each file that user drags:

```
*** ActiveX Control Event ***
LPARAMETERS data, effect, button, shift, x, y
```

```
local c, hit, iParent
c = 0
hit = 0
if ( data.GetFormat( 15 ) ) && exCFFiles
   if ( data.Files.Count() > 0 )
      with thisform.G2antt1.Items
         iParent = thisform.G2antt1.ItemFromPoint( x, y, @c, @hit )

         thisform.G2antt1.BeginUpdate()
         for i = 0 to data.files.Count() - 1
            .InsertItem( iParent, "", data.files(i) )
         next
         if ( iParent != 0 )
            .DefaultItem = iParent
            .ExpandItem( 0 ) = .t.
         endif
         thisform.G2antt1.EndUpdate()
      endwith
   endif
endif
```

event OLEDragOver (Data as ExDataObject, Effect as Long, Button as Integer, Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS, State as Integer)

Occurs when one component is dragged over another.

Type	Description
Data as ExDataObject	An ExDataObject object containing formats that the source will provide and, in addition, possibly the data for those formats. If no data is contained in the ExDataObject, it is provided when the control calls the GetData method. The SetData and Clear methods cannot be used here
Effect as Long	A Long set by the target component identifying the action that has been performed (if any), thus allowing the source to take appropriate action if the component was moved (such as the source deleting the data). The possible values are listed in Remarks.
Button as Integer	An integer which acts as a bit field corresponding to the state of a mouse button when it is depressed. The left button is bit 0, the right button is bit 1, and the middle button is bit 2. These bits correspond to the values 1, 2, and 4, respectively. It indicates the state of the mouse buttons; some, all, or none of these three bits can be set, indicating that some, all, or none of the buttons are depressed.
Shift as Integer	An integer which acts as a bit field corresponding to the state of the SHIFT, CTRL, and ALT keys when they are depressed. The SHIFT key is bit 0, the CTRL key is bit 1, and the ALT key is bit 2. These bits correspond to the values 1, 2, and 4, respectively. The shift parameter indicates the state of these keys; some, all, or none of the bits can be set, indicating that some, all, or none of the keys are depressed. For example, if both the CTRL and ALT keys were depressed, the value of shift would be 6.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The X value is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The Y value is always expressed in container coordinates.

State as Integer	An integer that corresponds to the transition state of the control being dragged in relation to a target form or control. The possible values are listed in Remarks.

The settings for effect are:

- exOLEDropEffectNone (0), Drop target cannot accept the data, or the drop operation was cancelled
- exOLEDropEffectCopy (1), Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
- exOLEDropEffectMove (2), Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

The settings for state are:

- exOLEDragEnter (0), Source component is being dragged within the range of a target.
- exOLEDragLeave (1), Source component is being dragged out of the range of a target.
- exOLEOLEDragOver (2), Source component has moved from one position in the target to another.

Note If the state parameter is 1, indicating that the mouse pointer has left the target, then the x and y parameters will contain zeros.
The source component should always mask values from the effect parameter to ensure compatibility with future implementations of ActiveX components. As a precaution against future problems, drag sources and drop targets should mask these values appropriately before performing any comparisons.
For example, a source component should not compare an effect against, say, exOLEDropEffectCopy, such as in this manner:

If Effect = exOLEDropEffectCopy...
Instead, the source component should mask for the value or values being sought, such as this:
If Effect And exOLEDropEffectCopy = exOLEDropEffectCopy...
-or-
If (Effect And exOLEDropEffectCopy)...
This allows for the definition of new drop effects in future versions while preserving backwards compatibility with your existing code.
The control supports only manual OLE drag and drop events.

Syntax for OLEDragOver event, **/NET** version, on:

C#
```
// OLEDragOver event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop ... events.
```

VB
```
// OLEDragOver event is not supported. Use the DragEnter,DragLeave,DragOver,
DragDrop ... events.
```

Syntax for OLEDragOver event, **/COM** version, on:

C#
```
private void OLEDragOver(object sender,
AxEXG2ANTTLib._IG2anttEvents_OLEDragOverEvent e)
{
}
```

C++
```
void OnOLEDragOver(LPDISPATCH Data,long FAR* Effect,short Button,short
Shift,long X,long Y,short State)
{
}
```

C++ Builder
```
void __fastcall OLEDragOver(TObject *Sender,Exg2anttlib_tlb::IExDataObject
*Data,long * Effect,short Button,short Shift,int X,int Y,short State)
{
}
```

Delphi
```
procedure OLEDragOver(ASender: TObject; Data : IExDataObject;var Effect :
Integer;Button : Smallint;Shift : Smallint;X : Integer;Y : Integer;State : Smallint);
begin
end;
```

Delphi 8 (.NET only)
```
procedure OLEDragOver(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_OLEDragOverEvent);
begin
end;
```

Powe…
```
begin event OLEDragOver(oleobject Data,long Effect,integer Button,integer
Shift,long X,long Y,integer State)
end event OLEDragOver
```

VB.NET
```
Private Sub OLEDragOver(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_OLEDragOverEvent) Handles OLEDragOver
End Sub
```

VB6

```
Private Sub OLEDragOver(ByVal Data As EXG2ANTTLibCtl.IExDataObject,Effect As
Long,ByVal Button As Integer,ByVal Shift As Integer,ByVal X As Single,ByVal Y As
Single,ByVal State As Integer)
End Sub
```

VBA

```
Private Sub OLEDragOver(ByVal Data As Object,Effect As Long,ByVal Button As
Integer,ByVal Shift As Integer,ByVal X As Long,ByVal Y As Long,ByVal State As
Integer)
End Sub
```

VFP

```
LPARAMETERS Data,Effect,Button,Shift,X,Y,State
```

Xbas…

```
PROCEDURE OnOLEDragOver(oG2antt,Data,Effect,Button,Shift,X,Y,State)
RETURN
```

Syntax for OLEDragOver event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="OLEDragOver(Data,Effect,Button,Shift,X,Y,State)"
LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function OLEDragOver(Data,Effect,Button,Shift,X,Y,State)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComOLEDragOver Variant llData Integer llEffect Short llButton Short
llShift OLE_XPOS_PIXELS llX OLE_YPOS_PIXELS llY Short llState
    Forward Send OnComOLEDragOver llData llEffect llButton llShift llX llY llState
End_Procedure
```

Visual Objects

```
METHOD OCX_OLEDragOver(Data,Effect,Button,Shift,X,Y,State) CLASS MainDialog
RETURN NIL
```

X++

```
// OLEDragOver event is not supported. Use the DragEnter,DragLeave,DragOver,
```

DragDrop ... events.

XBasic

```
function OLEDragOver as v (Data as OLE::Exontrol.G2antt.1::IExDataObject,Effect as N,Button as N,Shift as N,X as OLE::Exontrol.G2antt.1::OLE_XPOS_PIXELS,Y as OLE::Exontrol.G2antt.1::OLE_YPOS_PIXELS,State as N)
end function
```

dBASE

```
function nativeObject_OLEDragOver(Data,Effect,Button,Shift,X,Y,State)
return
```

event OLEGiveFeedback (Effect as Long, DefaultCursors as Boolean)

Allows the drag source to specify the type of OLE drag-and-drop operation and the visual feedback.

Type	Description
Effect as Long	A long integer set by the target component in the OLEDragOver event specifying the action to be performed if the user drops the selection on it. This allows the source to take the appropriate action (such as giving visual feedback). The possible values are listed in Remarks.
DefaultCursors as Boolean	Boolean value that determines whether to use the default mouse cursor, or to use a user-defined mouse cursor.True (default) = use default mouse cursor.False = do not use default cursor. Mouse cursor must be set with the MousePointer property of the Screen object.

The settings for Effect are:

- exOLEDropEffectNone (0), Drop target cannot accept the data, or the drop operation was cancelled
- exOLEDropEffectCopy (1), Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
- exOLEDropEffectMove (2), Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

If there is no code in the OLEGiveFeedback event, or if the defaultcursors parameter is set to True, the mouse cursor will be set to the default cursor provided by the control. The source component should always mask values from the effect parameter to ensure compatibility with future implementations of ActiveX components. As a precaution against future problems, drag sources and drop targets should mask these values appropriately before performing any comparisons.

For example, a source component should not compare an effect against, say, exOLEDropEffectCopy, such as in this manner:
If Effect = exOLEDropEffectCopy...
Instead, the source component should mask for the value or values being sought, such as this:
If Effect And exOLEDropEffectCopy = exOLEDropEffectCopy...
-or-
If (Effect And exOLEDropEffectCopy)...
This allows for the definition of new drop effects in future versions while preserving backwards compatibility with your existing code.

The control supports only manual OLE drag and drop events.

Syntax for OLEGiveFeedback event, **/NET** version, on:

C#
```
// OLEGiveFeedback event is not supported. Use the
DragEnter,DragLeave,DragOver, DragDrop ... events.
```

VB
```
// OLEGiveFeedback event is not supported. Use the
DragEnter,DragLeave,DragOver, DragDrop ... events.
```

Syntax for OLEGiveFeedback event, **/COM** version, on:

C#
```
private void OLEGiveFeedback(object sender,
AxEXG2ANTTLib._IG2anttEvents_OLEGiveFeedbackEvent e)
{
}
```

C++
```
void OnOLEGiveFeedback(long Effect,BOOL FAR* DefaultCursors)
{
}
```

C++ Builder
```
void __fastcall OLEGiveFeedback(TObject *Sender,long Effect,VARIANT_BOOL *
DefaultCursors)
{
}
```

Delphi
```
procedure OLEGiveFeedback(ASender: TObject; Effect : Integer;var DefaultCursors
: WordBool);
begin
end;
```

Delphi 8 (.NET only)
```
procedure OLEGiveFeedback(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_OLEGiveFeedbackEvent);
begin
end;
```

Powe…
```
begin event OLEGiveFeedback(long Effect,boolean DefaultCursors)
end event OLEGiveFeedback
```

VB.NET

```
Private Sub OLEGiveFeedback(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_OLEGiveFeedbackEvent) Handles OLEGiveFeedback
End Sub
```

VB6

```
Private Sub OLEGiveFeedback(ByVal Effect As Long,DefaultCursors As Boolean)
End Sub
```

VBA

```
Private Sub OLEGiveFeedback(ByVal Effect As Long,DefaultCursors As Boolean)
End Sub
```

VFP

```
LPARAMETERS Effect,DefaultCursors

```

Xbas…

```
PROCEDURE OnOLEGiveFeedback(oG2antt,Effect,DefaultCursors)
RETURN
```

Syntax for OLEGiveFeedback event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="OLEGiveFeedback(Effect,DefaultCursors)"
LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function OLEGiveFeedback(Effect,DefaultCursors)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComOLEGiveFeedback Integer llEffect Boolean llDefaultCursors
    Forward Send OnComOLEGiveFeedback llEffect llDefaultCursors
End_Procedure
```

Visual Objects

```
METHOD OCX_OLEGiveFeedback(Effect,DefaultCursors) CLASS MainDialog
RETURN NIL
```

X++

```
// OLEGiveFeedback event is not supported. Use the
DragEnter,DragLeave,DragOver, DragDrop ... events.
```

XBasic

```
function OLEGiveFeedback as v (Effect as N,DefaultCursors as L)
end function
```

dBASE

```
function nativeObject_OLEGiveFeedback(Effect,DefaultCursors)
return
```

event OLESetData (Data as ExDataObject, Format as Integer)

Occurs on a drag source when a drop target calls the GetData method and there is no data in a specified format in the OLE drag-and-drop DataObject.

Type	Description
Data as ExDataObject	An ExDataObject object in which to place the requested data. The component calls the SetData method to load the requested format.
Format as Integer	An integer specifying the format of the data that the target component is requesting. The source component uses this value to determine what to load into the ExDataObject object.

The OLESetData is not currently supported.

Syntax for OLESetData event, **/NET** version, on:

C#
```
// OLESetData event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop ... events.
```

VB
```
// OLESetData event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop ... events.
```

Syntax for OLESetData event, **/COM** version, on:

C#
```
private void OLESetData(object sender,
AxEXG2ANTTLib._IG2anttEvents_OLESetDataEvent e)
{
}
```

C++
```
void OnOLESetData(LPDISPATCH Data,short Format)
{
}
```

C++ Builder
```
void __fastcall OLESetData(TObject *Sender,Exg2anttlib_tlb::IExDataObject *Data,short Format)
{
}
```

Delphi

```
procedure OLESetData(ASender: TObject; Data : IExDataObject;Format : Smallint);
begin
end;
```

Delphi 8 (.NET only)

```
procedure OLESetData(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_OLESetDataEvent);
begin
end;
```

Powe…

```
begin event OLESetData(oleobject Data,integer Format)
end event OLESetData
```

VB.NET

```
Private Sub OLESetData(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_OLESetDataEvent) Handles OLESetData
End Sub
```

VB6

```
Private Sub OLESetData(ByVal Data As EXG2ANTTLibCtl.IExDataObject,ByVal
Format As Integer)
End Sub
```

VBA

```
Private Sub OLESetData(ByVal Data As Object,ByVal Format As Integer)
End Sub
```

VFP

```
LPARAMETERS Data,Format

```

Xbas…

```
PROCEDURE OnOLESetData(oG2antt,Data,Format)
RETURN
```

Syntax for OLESetData event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="OLESetData(Data,Format)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function OLESetData(Data,Format)
End Function
```

```
</SCRIPT>
```

Visual Data…

```
Procedure OnComOLESetData Variant llData Short llFormat
    Forward Send OnComOLESetData llData llFormat
End_Procedure
```

Visual Objects

```
METHOD OCX_OLESetData(Data,Format) CLASS MainDialog
RETURN NIL
```

X++

```
// OLESetData event is not supported. Use the DragEnter,DragLeave,DragOver,
DragDrop ... events.
```

XBasic

```
function OLESetData as v (Data as OLE::Exontrol.G2antt.1::IExDataObject,Format as
N)
end function
```

dBASE

```
function nativeObject_OLESetData(Data,Format)
return
```

event OLEStartDrag (Data as ExDataObject, AllowedEffects as Long)

Occurs when the OLEDrag method is called.

Type	Description
Data as ExDataObject	An ExDataObject object containing formats that the source will provide and, optionally, the data for those formats. If no data is contained in the ExDataObject, it is provided when the control calls the GetData method. The programmer should provide the values for this parameter in this event. The SetData and Clear methods cannot be used here.
AllowedEffects as Long	A long containing the effects that the source component supports. The possible values are listed in Settings. The programmer should provide the values for this parameter in this event

In the /NET Assembly, you have to use the DragEnter event as explained here:

- *https://www.exontrol.com/sg.jsp?content=support/faq/net/#dragdrop*

Use the Background(exDragDropBefore) property to specify the visual appearance for the dragging items, before painting the items. Use the Background(exDragDropAfter) property to specify the visual appearance for the dragging items, after painting the items. Use the Background(exDragDropList) property to specify the graphic feedback for the item from the cursor, while the OLE drag and drop operation is running. Use the AutoDrag property to specify what the control does when the user clicks and drag the items.

The settings for AllowEffects are:

- exOLEDropEffectNone (0), Drop target cannot accept the data, or the drop operation was cancelled
- exOLEDropEffectCopy (1), Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
- exOLEDropEffectMove (2), Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

The source component should logically Or together the supported values and places the result in the AllowedEffects parameter. The target component can use this value to determine the appropriate action (and what the appropriate user feedback should be). You may wish to defer putting data into the ExDataObject object until the target component requests it. This allows the source component to save time. If the user does not load any formats into the ExDataObject, then the drag/drop operation is canceled. Use exCFFiles

and Files property to add files to the drag and drop data object.

The idea of drag and drop in exG2antt control is the same as in other controls. To start accepting drag and drop sources the exG2antt control should have the OLEDropMode to exOLEDropManual. Once that is is set, the exG2antt starts accepting any drag and drop sources.

The first step is if you want to be able to drag items from your exG2antt control to other controls the idea is to handle the OLE_StartDrag event. The event passes an object ExDataObject (Data) as argument. The Data and AllowedEffects can be changed only in the OLEStartDrag event. The OLE_StartDrag event is fired when user is about to drag items from the control. **The AllowedEffect parameter and SetData property must be set to continue drag and drop operation, as in the following samples**:

Syntax for OLEStartDrag event, **/NET** version, on:

C#
```
// OLEStartDrag event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop ... events.
```

VB
```
// OLEStartDrag event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop ... events.
```

Syntax for OLEStartDrag event, **/COM** version, on:

C#
```
private void OLEStartDrag(object sender, AxEXG2ANTTLib._IG2anttEvents_OLEStartDragEvent e)
{
}
```

C++
```
void OnOLEStartDrag(LPDISPATCH Data,long FAR* AllowedEffects)
{
}
```

C++ Builder
```
void __fastcall OLEStartDrag(TObject *Sender,Exg2anttlib_tlb::IExDataObject *Data,long * AllowedEffects)
{
}
```

Delphi
```
procedure OLEStartDrag(ASender: TObject; Data : IExDataObject;var AllowedEffects : Integer);
begin
```

```
end;
```

Delphi 8 (.NET only)

```
procedure OLEStartDrag(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_OLEStartDragEvent);
begin
end;
```

Powe…

```
begin event OLEStartDrag(oleobject Data,long AllowedEffects)
end event OLEStartDrag
```

VB.NET

```
Private Sub OLEStartDrag(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_OLEStartDragEvent) Handles OLEStartDrag
End Sub
```

VB6

```
Private Sub OLEStartDrag(ByVal Data As
EXG2ANTTLibCtl.IExDataObject,AllowedEffects As Long)
End Sub
```

VBA

```
Private Sub OLEStartDrag(ByVal Data As Object,AllowedEffects As Long)
End Sub
```

VFP

```
LPARAMETERS Data,AllowedEffects

```

Xbas…

```
PROCEDURE OnOLEStartDrag(oG2antt,Data,AllowedEffects)
RETURN
```

Syntax for OLEStartDrag event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="OLEStartDrag(Data,AllowedEffects)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function OLEStartDrag(Data,AllowedEffects)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComOLEStartDrag Variant llData Integer llAllowedEffects
    Forward Send OnComOLEStartDrag llData llAllowedEffects
End_Procedure
```

Visual Objects

```
METHOD OCX_OLEStartDrag(Data,AllowedEffects) CLASS MainDialog
RETURN NIL
```

X++

```
// OLEStartDrag event is not supported. Use the DragEnter,DragLeave,DragOver,
DragDrop ... events.
```

XBasic

```
function OLEStartDrag as v (Data as
OLE::Exontrol.G2antt.1::IExDataObject,AllowedEffects as N)
end function
```

dBASE

```
function nativeObject_OLEStartDrag(Data,AllowedEffects)
return
```

The following VB sample drags data from a control to another, by registering a new clipboard format:

```
Private Sub G2antt1_OLEStartDrag(Index As Integer, ByVal Data As
EXG2ANTTLibCtl.IExDataObject, AllowedEffects As Long)

    ' We are going to add two clipboard formats: text and "EXG2ANTT" clipboard format.
    ' We need to use RegisterClipboardFormat API function in order to register our
    ' clipboard format. One cliboard format is enough, but the sample shows
    ' how to filter in OLEDragDrop event the other clipboard formats

    ' Builds a string that contains each cell's caption on a new line
    Dim n As Long
    Dim s As String
    With G2antt1(Index)
        s = Index & vbCrLf ' Saves the source
        For n = 0 To .Columns.Count - 1
            s = s & .Items.CellValue(.Items.SelectedItem(0), n) & vbCrLf
```

```
    Next
  End With

  AllowedEffects = 0
  ' Checks whether the selected item has a parent
  If (G2antt1(Index).Items.ItemParent(G2antt1(Index).Items.SelectedItem(0)) <> 0) Then
    AllowedEffects = 1
  End If
  ' Sets the text clipboard format
  Data.SetData s, exCFText

  ' Builds an array of bytes, and copy there all characters in the s string.
  ' Passes the array to the SetData method.
  ReDim v(Len(s)) As Byte
  For n = 0 To Len(s) - 1
    v(n) = Asc(Mid(s, n + 1, 1))
  Next
  Data.SetData v, RegisterClipboardFormat("EXG2ANTT")

End Sub
```

The code fills data for two types of clipboard formats: text (CF_TEXT) and "EXG2ANTT" registered clipboard format. The registered clipboard format must be an array of bytes. As you can see we have used the RegisterClipboardFormat API function, and it should be declared like:

```
Private Declare Function RegisterClipboardFormat Lib "user32" Alias
"RegisterClipboardFormatA" (ByVal lpString As String) As Integer
```

The second step is accepting OLE drag and drop source objects. That means, if you would like to let your control accept drag and drop objects, you have to handle the OLEDragDrop event. It gets as argument an object Data that stores the drag and drop information. The next sample shows how handle the OLEDragDrop event:

```
Private Sub G2antt1_OLEDragDrop(Index As Integer, ByVal Data As
EXG2ANTTLibCtl.IExDataObject, Effect As Long, ByVal Button As Integer, ByVal Shift As
Integer, ByVal X As Single, ByVal Y As Single)
  ' Checks whether the clipboard format is our. Since we have registered the clipboard in
the
```

```
  ' OLEStartData format we now its format, so we can handle this type of clip formats.
  If (Data.GetFormat(RegisterClipboardFormat("EXG2ANTT"))) Then
    ' Builds the saved string from the array passed
    Dim s As String
    Dim v() As Byte
    Dim n As Integer
    v = Data.GetData(RegisterClipboardFormat("EXG2ANTT"))
    For n = LBound(v) To UBound(v)
      s = s + Chr(v(n))
    Next
    Debug.Print s

    'Adds a new item to the control, and sets the cells captions like we saved, line by line
    G2antt1(Index).Visible = False
    With G2antt1(Index)
      .BeginUpdate
      Dim i As HITEM
      Dim item As String
      Dim nCur As Long
      i = .Items.AddItem()
      nCur = InStr(1, s, vbCrLf) + Len(vbCrLf) ' Jumps the source
      For n = 0 To .Columns.Count - 1
        Dim nnCur As Long
        nnCur = InStr(nCur, s, vbCrLf)
        .Items.CellValue(i, n) = Mid(s, nCur, nnCur - nCur)
        nCur = nnCur + Len(vbCrLf)
      Next
      .Items.CellImage(i, "EmployeeID") = Int(.Items.CellValue(i, "EmployeeID"))
      .Items.SetParent i, h(Index, Int(.Items.CellValue(i, "EmployeeID")) - 1)
      .Items.EnsureVisibleItem i
      .EndUpdate
    End With
    G2antt1(Index).Visible = True
  End If
End Sub
```

The following VC sample copies the selected items to the clipboard, as soon as the user

starts dragging the items:

```
#import <exg2antt.dll> rename( "GetItems", "exGetItems" )

#include "Items.h"
#include "Columns.h"

static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
   if ( pv )
   {
      if ( pv->vt == VT_ERROR )
         return szDefault;

      COleVariant vt;
      vt.ChangeType( VT_BSTR, pv );
      return V_BSTR( &vt );
   }
   return szDefault;
}

void OnOLEStartDragG2antt1(LPDISPATCH Data, long FAR* AllowedEffects)
{
   CItems items = m_g2antt.GetItems();
   long nCount = items.GetSelectCount(), nColumnCount =
m_g2antt.GetColumns().GetCount();
   if ( nCount > 0 )
   {
      *AllowedEffects = /*exOLEDropEffectCopy */ 1;
      EXG2ANTTLib::IExDataObjectPtr spData( Data );
      if ( spData !=NULL )
      {
         CString strData;
         for ( long i = 0; i < nCount; i++ )
         {
            COleVariant vtItem( items.GetSelectedItem( i ) );
            for ( long j = 0; j < nColumnCount; j++ )
```

```
            strData += V2S( &items.GetCellValue( vtItem, COleVariant( j ) ) ) + "\t";
        }
        strData += "\r\n";
        spData->SetData( COleVariant( strData ), COleVariant(
(long)EXG2ANTTLib::exCFText) );
      }
   }
}
```

The sample saves data as CF_TEXT format (EXG2ANTTLib::exCFText). The data is a text, where each item is separated by "\r\n" (new line), and each cell is separated by "\t" (TAB charcater). Of course, data can be saved as you want. The sample only gives an idea of what and how it could be done. The sample uses the #import statement to import the control's type library, including definitions for ExDataObject and ExDataObjectFiles that are required to fill data to be dragged. If your exg2antt.dll file is located in another place than your system folder, the path to the exg2antt.dll file needs to be specified, else compiler errors occur.

The following VB.NET sample copies the selected items to the clipboard, as soon as the user starts dragging the items:

```
Private Sub AxG2antt1_OLEStartDrag(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_OLEStartDragEvent) Handles AxG2antt1.OLEStartDrag
   With AxG2antt1.Items
      If (.SelectCount > 0) Then
         e.allowedEffects = 1 'exOLEDropEffectCopy
         Dim i As Integer, j As Integer, strData As String, nColumnCount As Long =
AxG2antt1.Columns.Count
         For i = 0 To .SelectCount - 1
            For j = 0 To nColumnCount - 1
               strData = strData + .CellValue(.SelectedItem(i), j) + Chr(Keys.Tab)
            Next
         Next
         strData = strData + vbCrLf
         e.data.SetData(strData, EXG2ANTTLib.exClipboardFormatEnum.exCFText)
      End If
   End With
End Sub
```

The following C# sample copies the selected items to the clipboard, as soon as the user starts dragging the items:

```
private void axG2antt1_OLEStartDrag(object sender,
AxEXG2ANTTLib._IG2anttEvents_OLEStartDragEvent e)
{
   int nCount = axG2antt1.Items.SelectCount;
   if ( nCount > 0 )
   {
      int nColumnCount = axG2antt1.Columns.Count;
      e.allowedEffects = /*exOLEDropEffectCopy*/ 1;
      string strData = "";
      for ( int i =0 ; i < nCount; i++ )
      {
         for ( int j = 0; j < nColumnCount; j++ )
         {
            object strCell =
axG2antt1.Items.get_CellValue(axG2antt1.Items.get_SelectedItem(i), j);
            strData +=  ( strCell != null ? strCell.ToString() : "" ) + "\t";
         }
         strData += "\r\n";
      }
      e.data.SetData( strData, EXG2ANTTLib.exClipboardFormatEnum.exCFText );
   }
}
```

The following VFP sample copies the selected items to the clipboard, as soon as the user starts dragging the items:

```
*** ActiveX Control Event ***
LPARAMETERS data, allowedeffects

local sData, nColumnCount, i, j
with thisform.G2antt1.Items
   if ( .SelectCount() > 0 )
      allowedeffects = 1 && exOLEDropEffectCopy
      sData = ""
      nColumnCount = thisform.G2antt1.Columns.Count
```

```
    for i = 0 to .SelectCount - 1
      for j = 0 to nColumnCount
        sData = sData + .CellValue( .SelectedItem(i), j ) + chr(9)
      next
      sData = sData + chr(10)+ chr(13)
    next
    data.SetData( sData, 1 ) && exCFText
  endif
endwith
```

event OversizeChanged (Horizontal as Boolean, NewVal as Long)

Occurs when the right range of the scroll has been changed.

Type	Description
Horizontal as Boolean	A boolean expression that indicates whether the horizontal scroll bar has changed.
NewVal as Long	A long value that indicates the new scroll bar value.

If the control has no scroll bars the OffsetChanged and OversizeChanged events are not fired. When the scroll bar range is changed the OversizeChanged event is fired. Use the ScrollBars property of the control to determine which scroll bars are visible within the control. The control fires the LayoutChanged event when the user resizes a column, or change its position.

Syntax for OversizeChanged event, **/NET** version, on:

C#
```
private void OversizeChanged(object sender,bool Horizontal,int NewVal)
{
}
```

VB
```
Private Sub OversizeChanged(ByVal sender As System.Object,ByVal Horizontal As Boolean,ByVal NewVal As Integer) Handles OversizeChanged
End Sub
```

Syntax for OversizeChanged event, **/COM** version, on:

C#
```
private void OversizeChanged(object sender,
AxEXG2ANTTLib._IG2anttEvents_OversizeChangedEvent e)
{
}
```

C++
```
void OnOversizeChanged(BOOL Horizontal,long NewVal)
{
}
```

C++ Builder
```
void __fastcall OversizeChanged(TObject *Sender,VARIANT_BOOL Horizontal,long NewVal)
{
}
```

Delphi

```
procedure OversizeChanged(ASender: TObject; Horizontal : WordBool;NewVal :
Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure OversizeChanged(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_OversizeChangedEvent);
begin
end;
```

Powe…

```
begin event OversizeChanged(boolean Horizontal,long NewVal)
end event OversizeChanged
```

VB.NET

```
Private Sub OversizeChanged(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_OversizeChangedEvent) Handles
OversizeChanged
End Sub
```

VB6

```
Private Sub OversizeChanged(ByVal Horizontal As Boolean,ByVal NewVal As Long)
End Sub
```

VBA

```
Private Sub OversizeChanged(ByVal Horizontal As Boolean,ByVal NewVal As Long)
End Sub
```

VFP

```
LPARAMETERS Horizontal,NewVal

```

Xbas…

```
PROCEDURE OnOversizeChanged(oG2antt,Horizontal,NewVal)
RETURN
```

Syntax for OversizeChanged event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="OversizeChanged(Horizontal,NewVal)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function OversizeChanged(Horizontal,NewVal)
End Function
```

```
</SCRIPT>
```

Visual Data…

```
Procedure OnComOversizeChanged Boolean llHorizontal Integer llNewVal
    Forward Send OnComOversizeChanged llHorizontal llNewVal
End_Procedure
```

Visual Objects

```
METHOD OCX_OversizeChanged(Horizontal,NewVal) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_OversizeChanged(boolean _Horizontal,int _NewVal)
{
}
```

XBasic

```
function OversizeChanged as v (Horizontal as L,NewVal as N)
end function
```

dBASE

```
function nativeObject_OversizeChanged(Horizontal,NewVal)
return
```

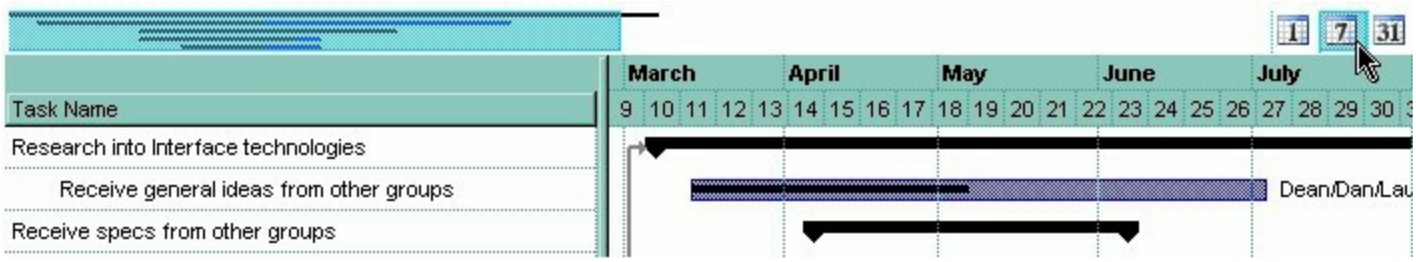
event OverviewZoom ()

Occurs once the user selects a new time scale unit in the overview zoom area.

Type	Description

The OverviewZoom event notifies your application once the user clicks or select a new time-scale in the overview-zoom area. The UnitScale property specifies the new selected time scale. Use the UnitWidth property to specify the width of the units in the chart area. Use the OverviewVisible property to show the control's overview area. Use the AllowOverviewZoom property to specify how the zoom scale is displayed on the control's overview area.

The following screen shot shows the zoom area in the control's overview area (in the top-right corner):



Syntax for OverviewZoom event, **/NET** version, on:

C#

```
private void OverviewZoom(object sender)
{
}
```

VB

```
Private Sub OverviewZoom(ByVal sender As System.Object) Handles
OverviewZoom
End Sub
```

Syntax for OverviewZoom event, **/COM** version, on:

C#

```
private void OverviewZoom(object sender, EventArgs e)
{
}
```

C++

```
void OnOverviewZoom()
{
}
```

C++ Builder

```
void __fastcall OverviewZoom(TObject *Sender)
{
}
```

Delphi

```
procedure OverviewZoom(ASender: TObject; );
begin
end;
```

Delphi 8 (.NET only)

```
procedure OverviewZoom(sender: System.Object; e: System.EventArgs);
begin
end;
```

Powe…

```
begin event OverviewZoom()
end event OverviewZoom
```

VB.NET

```
Private Sub OverviewZoom(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles OverviewZoom
End Sub
```

VB6

```
Private Sub OverviewZoom()
End Sub
```

VBA

```
Private Sub OverviewZoom()
End Sub
```

VFP

```
LPARAMETERS nop
```

Xbas…

```
PROCEDURE OnOverviewZoom(oG2antt)
RETURN
```

Syntax for OverviewZoom event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="OverviewZoom()" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function OverviewZoom()
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComOverviewZoom
    Forward Send OnComOverviewZoom
End_Procedure
```

Visual Objects

```
METHOD OCX_OverviewZoom() CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_OverviewZoom()
{
}
```

XBasic

```
function OverviewZoom as v ()
end function
```

dBASE

```
function nativeObject_OverviewZoom()
return
```

event RClick ()

Fired when right mouse button is clicked.

Type	Description

Use the RClick event to add your context menu. The RClick event notifies your application when the user right clicks the control. Use the Click event to notify your application that the user clicks the control (using the left mouse button). Use the MouseDown or MouseUp event if you require the cursor position during the RClick event. Use the RClickSelect property to specify whether the user can select items by right clicking the mouse. Use the ItemFromPoint property to get the item from point. Use the ColumnFromPoint property to get the column from point. Use the AllowOverviewZoom property to specify whether the control displays the zooming scale on the overview area, when the user right clicks the overview area.

Syntax for RClick event, **/NET** version, on:

C#

```
private void RClick(object sender)
{
}
```

VB

```
Private Sub RClick(ByVal sender As System.Object) Handles RClick
End Sub
```

Syntax for RClick event, **/COM** version, on:

C#

```
private void RClick(object sender, EventArgs e)
{
}
```

C++

```
void OnRClick()
{
}
```

C++ Builder

```
void __fastcall RClick(TObject *Sender)
{
}
```

Delphi

```
procedure RClick(ASender: TObject; );
begin
```

```
end;
```

Delphi 8 (.NET only)

```
procedure RClick(sender: System.Object; e: System.EventArgs);
begin
end;
```

Powe…

```
begin event RClick()
end event RClick
```

VB.NET

```
Private Sub RClick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles RClick
End Sub
```

VB6

```
Private Sub RClick()
End Sub
```

VBA

```
Private Sub RClick()
End Sub
```

VFP

```
LPARAMETERS nop

```

Xbas…

```
PROCEDURE OnRClick(oG2antt)
RETURN
```

Syntax for RClick event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="RClick()" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function RClick()
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComRClick
   Forward Send OnComRClick
End_Procedure
```

Visual Objects

```
METHOD OCX_RClick() CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_RClick()
{
}
```

XBasic

```
function RClick as v ()
end function
```

dBASE

```
function nativeObject_RClick()
return
```

The following VB sample use Exontrol's ExPopupMenu Component to display a context menu when user has clicked the right mouse button in the control's client area:

```
Private Sub G2antt1_RClick()
   Dim i As Long
   i = PopupMenu1.ShowAtCursor
End Sub
```

If you need to add a context menu based on the item you can use the MouseUp event, like in the following VB sample:

```
Private Sub G2antt1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
   If (Button = 2) Then
      ' Converts the container coordinates to client coordinates
      X = X / Screen.TwipsPerPixelX
      Y = Y / Screen.TwipsPerPixelY
      Dim h As HITEM
      Dim c As Long, hit as Long
      ' Gets the item from (X,Y)
      h = G2antt1.ItemFromPoint(X, Y, c, hit)
      If Not (h = 0) Then
         Dim i As Long
         PopupMenu1.Items.Add G2antt1.Items.CellValue(h, c)
         i = PopupMenu1.ShowAtCursor
      End If
```

```
    End If
End Sub
```

The following VC sample displays the caption of the cell where the mouse is released:

```
#include "Items.h"

static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
    if ( pv )
    {
        if ( pv->vt == VT_ERROR )
            return szDefault;

        COleVariant vt;
        vt.ChangeType( VT_BSTR, pv );
        return V_BSTR( &vt );
    }
    return szDefault;
}

void OnMouseUpG2antt1(short Button, short Shift, long X, long Y)
{
    long c = 0, hit = 0, hItem = m_g2antt.GetItemFromPoint( X, Y, &c, &hit );
    if ( ( hItem != 0 ) || ( c != 0 ) )
    {
        CItems items = m_g2antt.GetItems();
        COleVariant vtItem( hItem ), vtColumn( c );
        CString strCaption = V2S( &items.GetCellValue( vtItem, vtColumn ) ), strOutput;
        strOutput.Format( "Cell: '%s', Hit = %08X\n", strCaption, hit );
        OutputDebugString( strOutput );
    }
}
```

The following VB.NET sample displays the caption of the cell where the mouse is released:

```
Private Sub AxG2antt1_MouseUpEvent(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_MouseUpEvent) Handles AxG2antt1.MouseUpEvent
```

```
    With AxG2antt1
        Dim i As Integer, c As Integer, hit As EXG2ANTTLib.HitTestInfoEnum
        i = .get_ItemFromPoint(e.x, e.y, c, hit)
        If (Not (i = 0) Or Not (c = 0)) Then
            Debug.WriteLine("Cell: " & .Items.CellValue(i, c) & " Hit: " & hit.ToString())
        End If
    End With
End Sub
```

The following C# sample displays the caption of the cell where the mouse is released:

```
private void axG2antt1_MouseUpEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_MouseUpEvent e)
{
    int c = 0;
    EXG2ANTTLib.HitTestInfoEnum hit;
    int i = axG2antt1.get_ItemFromPoint( e.x, e.y, out c,out hit );
    if ( ( i != 0 ) || ( c != 0 ) )
    {
        string s = axG2antt1.Items.get_CellValue( i,c ).ToString();
        s = "Cell: " + s + ", Hit: " + hit.ToString();
        System.Diagnostics.Debug.WriteLine( s );
    }
}
```

The following VFP sample displays the caption of the cell where the mouse is released:

```
*** ActiveX Control Event ***
LPARAMETERS button, shift, x, y

local c, hit
c = 0
hit = 0
with thisform.G2antt1
    .Items.DefaultItem = .ItemFromPoint( x, y, @c, @hit )
    if ( .Items.DefaultItem <> 0 ) or ( c <> 0 )
        wait window nowait .Items.CellValue( 0, c ) + " " + Str( hit )
    endif
```

endwith

event RemoveColumn (Column as Column)

Fired before deleting a column.

Type	Description
Column as Column	A Column object being removed.

The RemoveColumn event is invoked when the control is about to remove a column. Use the RemoveColumn event to release any extra data associated to the column. Use the Remove method to remove a specific column from Columns collection. Use the Clear method to clear the columns collection. Use the RemoveItem method to remove an item. Use the RemoveAllItems method to remove all items. Use the CellData property to assign an extra data to a cell. Use the ItemData property to assign an extra data to an item. Use the Data property to assign an extra data to a column.

Syntax for RemoveColumn event, **/NET** version, on:

C#

```
private void RemoveColumn(object sender,exontrol.EXG2ANTTLib.Column Column)
{
}
```

VB

```
Private Sub RemoveColumn(ByVal sender As System.Object,ByVal Column As exontrol.EXG2ANTTLib.Column) Handles RemoveColumn
End Sub
```

Syntax for RemoveColumn event, **/COM** version, on:

C#

```
private void RemoveColumn(object sender, AxEXG2ANTTLib._IG2anttEvents_RemoveColumnEvent e)
{
}
```

C++

```
void OnRemoveColumn(LPDISPATCH Column)
{
}
```

C++ Builder

```
void __fastcall RemoveColumn(TObject *Sender,Exg2anttlib_tlb::IColumn *Column)
{
}
```

Delphi
```
procedure RemoveColumn(ASender: TObject; Column : IColumn);
begin
end;
```

Delphi 8 (.NET only)
```
procedure RemoveColumn(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_RemoveColumnEvent);
begin
end;
```

Powe…
```
begin event RemoveColumn(oleobject Column)
end event RemoveColumn
```

VB.NET
```
Private Sub RemoveColumn(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_RemoveColumnEvent) Handles RemoveColumn
End Sub
```

VB6
```
Private Sub RemoveColumn(ByVal Column As EXG2ANTTLibCtl.IColumn)
End Sub
```

VBA
```
Private Sub RemoveColumn(ByVal Column As Object)
End Sub
```

VFP
```
LPARAMETERS Column
```

Xbas…
```
PROCEDURE OnRemoveColumn(oG2antt,Column)
RETURN
```

Syntax for RemoveColumn event, ***/COM*** version (others), on:

Java…
```
<SCRIPT EVENT="RemoveColumn(Column)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function RemoveColumn(Column)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComRemoveColumn Variant llColumn
    Forward Send OnComRemoveColumn llColumn
End_Procedure
```

Visual Objects

```
METHOD OCX_RemoveColumn(Column) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_RemoveColumn(COM _Column)
{
}
```

XBasic

```
function RemoveColumn as v (Column as OLE::Exontrol.G2antt.1::IColumn)
end function
```

dBASE

```
function nativeObject_RemoveColumn(Column)
return
```

event RemoveItem (Item as HITEM)

Occurs before removing an Item.

Type	Description
Item as HITEM	A long expression that indicates the handle of the item being removed.

Use the RemoveItem to release any extra data that you might have used. The control fires the RemoveItem event before removing the item. Use the RemoveItem method to remove an item from Items collection. Use the RemoveAllItems method to clear the items collection. Use the Remove method to remove a column. Use the Clear method to clear the columns collection. Use the CellData property to assign an extra data to a cell. Use the ItemData property to assign an extra data to an item. Use the Data property to assign an extra data to a column.

Syntax for RemoveItem event, **/NET** version, on:

C#

```
private void RemoveItem(object sender,int Item)
{
}
```

VB

```
Private Sub RemoveItem(ByVal sender As System.Object,ByVal Item As Integer)
Handles RemoveItem
End Sub
```

Syntax for RemoveItem event, **/COM** version, on:

C#

```
private void RemoveItem(object sender,
AxEXG2ANTTLib._IG2anttEvents_RemoveItemEvent e)
{
}
```

C++

```
void OnRemoveItem(long Item)
{
}
```

C++ Builder

```
void __fastcall RemoveItem(TObject *Sender,Exg2anttlib_tlb::HITEM Item)
{
}
```

Delphi
```
procedure RemoveItem(ASender: TObject; Item : HITEM);
begin
end;
```

Delphi 8 (.NET only)
```
procedure RemoveItem(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_RemoveItemEvent);
begin
end;
```

Powe…
```
begin event RemoveItem(long Item)
end event RemoveItem
```

VB.NET
```
Private Sub RemoveItem(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_RemoveItemEvent) Handles RemoveItem
End Sub
```

VB6
```
Private Sub RemoveItem(ByVal Item As EXG2ANTTLibCtl.HITEM)
End Sub
```

VBA
```
Private Sub RemoveItem(ByVal Item As Long)
End Sub
```

VFP
```
LPARAMETERS Item

```

Xbas…
```
PROCEDURE OnRemoveItem(oG2antt,Item)
RETURN
```

Syntax for RemoveItem event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="RemoveItem(Item)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function RemoveItem(Item)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComRemoveItem HITEM llItem
   Forward Send OnComRemoveItem llItem
End_Procedure
```

Visual Objects

```
METHOD OCX_RemoveItem(Item) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_RemoveItem(int _Item)
{
}
```

XBasic

```
function RemoveItem as v (Item as OLE::Exontrol.G2antt.1::HITEM)
end function
```

dBASE

```
function nativeObject_RemoveItem(Item)
return
```

event ScrollButtonClick (ScrollBar as ScrollBarEnum, ScrollPart as ScrollPartEnum)

Occurs when the user clicks a button in the scrollbar.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBarEnum expression that specifies the scroll bar being clicked.
ScrollPart as ScrollPartEnum	A ScrollPartEnum expression that indicates the part of the scroll being clicked.

Use the ScrollButtonClick event to notify your application that the user clicks a button in the control's scrollbar. The ScrollButtonClick event is fired when the user clicks and releases the mouse over an enabled part of the scroll bar. Use the ScrollBars property to specify the visible scrollbars in the control. Use the ScrollPartVisible property to add or remove buttons/parts in the control's scrollbar. Use the ScrollPartEnable property to specify enable or disable parts in the control's scrollbar. Use the ScrolPartCaption property to specify the caption of the scroll's part. Use the OffsetChanged event to notify your application that the scroll position is changed. Use the OversizeChanged event to notify your application whether the range for a specified scroll bar is changed. Use the ScrollPos property to specify the position for the control's scroll bar. Use the Background property to change the visual appearance for any part in the control's scroll bar.

Syntax for ScrollButtonClick event, **/NET** version, on:

C#
```
private void ScrollButtonClick(object
sender,exontrol.EXG2ANTTLib.ScrollBarEnum
ScrollBar,exontrol.EXG2ANTTLib.ScrollPartEnum ScrollPart)
{
}
```

VB
```
Private Sub ScrollButtonClick(ByVal sender As System.Object,ByVal ScrollBar As
exontrol.EXG2ANTTLib.ScrollBarEnum,ByVal ScrollPart As
exontrol.EXG2ANTTLib.ScrollPartEnum) Handles ScrollButtonClick
End Sub
```

Syntax for ScrollButtonClick event, **/COM** version, on:

C#
```
private void ScrollButtonClick(object sender,
AxEXG2ANTTLib._IG2anttEvents_ScrollButtonClickEvent e)
{
```

```
}
```

C++

```
void OnScrollButtonClick(long ScrollBar,long ScrollPart)
{
}
```

C++ Builder

```
void __fastcall ScrollButtonClick(TObject *Sender,Exg2anttlib_tlb::ScrollBarEnum
ScrollBar,Exg2anttlib_tlb::ScrollPartEnum ScrollPart)
{
}
```

Delphi

```
procedure ScrollButtonClick(ASender: TObject; ScrollBar :
ScrollBarEnum;ScrollPart : ScrollPartEnum);
begin
end;
```

Delphi 8 (.NET only)

```
procedure ScrollButtonClick(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_ScrollButtonClickEvent);
begin
end;
```

Powe…

```
begin event ScrollButtonClick(long ScrollBar,long ScrollPart)
end event ScrollButtonClick
```

VB.NET

```
Private Sub ScrollButtonClick(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_ScrollButtonClickEvent) Handles
ScrollButtonClick
End Sub
```

VB6

```
Private Sub ScrollButtonClick(ByVal ScrollBar As
EXG2ANTTLibCtl.ScrollBarEnum,ByVal ScrollPart As
EXG2ANTTLibCtl.ScrollPartEnum)
End Sub
```

VBA

```
Private Sub ScrollButtonClick(ByVal ScrollBar As Long,ByVal ScrollPart As Long)
End Sub
```

VFP

```
LPARAMETERS ScrollBar,ScrollPart
```

```
```

Xbas…
```
PROCEDURE OnScrollButtonClick(oG2antt,ScrollBar,ScrollPart)
RETURN
```

Syntax for ScrollButtonClick event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="ScrollButtonClick(ScrollBar,ScrollPart)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function ScrollButtonClick(ScrollBar,ScrollPart)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComScrollButtonClick OLEScrollBarEnum llScrollBar
OLEScrollPartEnum llScrollPart
    Forward Send OnComScrollButtonClick llScrollBar llScrollPart
End_Procedure
```

Visual Objects
```
METHOD OCX_ScrollButtonClick(ScrollBar,ScrollPart) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_ScrollButtonClick(int _ScrollBar,int _ScrollPart)
{
}
```

XBasic
```
function ScrollButtonClick as v (ScrollBar as
OLE::Exontrol.G2antt.1::ScrollBarEnum,ScrollPart as
OLE::Exontrol.G2antt.1::ScrollPartEnum)
end function
```

dBASE
```
function nativeObject_ScrollButtonClick(ScrollBar,ScrollPart)
return
```

The following VB sample displays the identifier of the scroll's button being clicked:

```
With G2antt1
   .BeginUpdate
      .ScrollBars = exDisableBoth
      .ScrollPartVisible(exVScroll, exLeftB1Part Or exRightB1Part) = True
      .ScrollPartCaption(exVScroll, exLeftB1Part) = "<img></img>1"
      .ScrollPartCaption(exVScroll, exRightB1Part) = "<img></img>2"
   .EndUpdate
End With
```

```
Private Sub G2antt1_ScrollButtonClick(ByVal ScrollPart As
EXG2ANTTLibCtl.ScrollPartEnum)
   MsgBox (ScrollPart)
End Sub
```

The following VB.NET sample displays the identifier of the scroll's button being clicked:

```
With AxG2antt1
   .BeginUpdate()
   .ScrollBars = EXG2ANTTLib.ScrollBarsEnum.exDisableBoth
   .set_ScrollPartVisible(EXG2ANTTLib.ScrollBarEnum.exVScroll,
EXG2ANTTLib.ScrollPartEnum.exLeftB1Part Or
EXG2ANTTLib.ScrollPartEnum.exRightB1Part, True)
   .set_ScrollPartCaption(EXG2ANTTLib.ScrollBarEnum.exVScroll,
EXG2ANTTLib.ScrollPartEnum.exLeftB1Part, "<img></img>1")
   .set_ScrollPartCaption(EXG2ANTTLib.ScrollBarEnum.exVScroll,
EXG2ANTTLib.ScrollPartEnum.exRightB1Part, "<img></img>2")
   .EndUpdate()
End With
```

```
Private Sub AxG2antt1_ScrollButtonClick(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_ScrollButtonClickEvent) Handles
AxG2antt1.ScrollButtonClick
      MessageBox.Show( e.scrollPart.ToString())
End Sub
```

The following C# sample displays the identifier of the scroll's button being clicked:

```
axG2antt1.BeginUpdate();
```

```
axG2antt1.ScrollBars = EXG2ANTTLib.ScrollBarsEnum.exDisableBoth;
axG2antt1.set_ScrollPartVisible(EXG2ANTTLib.ScrollBarEnum.exVScroll,
EXG2ANTTLib.ScrollPartEnum.exLeftB1Part | EXG2ANTTLib.ScrollPartEnum.exRightB1Part,
true);
axG2antt1.set_ScrollPartCaption(EXG2ANTTLib.ScrollBarEnum.exVScroll,
EXG2ANTTLib.ScrollPartEnum.exLeftB1Part , "<img></img>1");
axG2antt1.set_ScrollPartCaption(EXG2ANTTLib.ScrollBarEnum.exVScroll,
EXG2ANTTLib.ScrollPartEnum.exRightB1Part, "<img></img>2");
axG2antt1.EndUpdate();
```

```
private void axG2antt1_ScrollButtonClick(object sender,
AxEXG2ANTTLib._IG2anttEvents_ScrollButtonClickEvent e)
{
   MessageBox.Show(e.scrollPart.ToString());
}
```

The following C++ sample displays the identifier of the scroll's button being clicked:

```
m_g2antt.BeginUpdate();
m_g2antt.SetScrollBars( 15 /*exDisableBoth*/ );
m_g2antt.SetScrollPartVisible( 0 /*exVScroll*/, 32768  /*exLeftB1Part*/ | 32
/*exRightB1Part*/, TRUE );
m_g2antt.SetScrollPartCaption( 0 /*exVScroll*/, 32768  /*exLeftB1Part*/ , _T("<img>
</img>1") );
m_g2antt.SetScrollPartCaption( 0 /*exVScroll*/, 32 /*exRightB1Part*/ , _T("<img>
</img>2") );
m_g2antt.EndUpdate();
```

```
void OnScrollButtonClickG2antt1(long ScrollPart)
{
   CString strFormat;
   strFormat.Format( _T("%i"), ScrollPart );
   MessageBox( strFormat );
}
```

The following VFP sample displays the identifier of the scroll's button being clicked:

```
With thisform.G2antt1
```

```
    .BeginUpdate
        .ScrollBars = 15
        .ScrollPartVisible(0, bitor(32768,32)) = .t.
        .ScrollPartCaption(0,32768) = "<img></img>1"
        .ScrollPartCaption(0, 32) = "<img></img>2"
    .EndUpdate
EndWith
```

```
*** ActiveX Control Event ***
LPARAMETERS scrollpart

wait window nowait ltrim(str(scrollpart))
```

event SelectionChanged ()

Fired after a new item has been selected.

Type	Description

Use the SelectionChanged event to notify your application that the user selects an item (that's selectable). Use the SelectableItem property to specify the user can select an item. The control supports single or multiple selection as well. When an item is selected or unselected the control fires the SelectionChanged event. Use the SingleSel property to specify if your control supports single or multiple selection. Use the SelectCount property to get the number of selected items. Use the SelectedItem property to get the selected item. Use the SelectItem to select or unselect a specified item. Use the FocusItem property to get the focused item. If the control supports only single selection, you can use the FocusItem property to get the selected/focused item because they are always the same. Use the SelForeColor and SelBackColor properties to specify colors for selected items. The AllowSelectObjects property allows users to select at runtime the bars and links in the chart area. Use the ItemBar(exBarSelected) property to select or unselect programmatically a bar. Use the Link(exLinkSelected) property to select or unselect programmatically a link. Use the SelectOnClick property to disable selecting new items when the user clicks the chart area.

Syntax for SelectionChanged event, **/NET** version, on:

C#
```
private void SelectionChanged(object sender)
{
}
```

VB
```
Private Sub SelectionChanged(ByVal sender As System.Object) Handles
SelectionChanged
End Sub
```

Syntax for SelectionChanged event, **/COM** version, on:

C#
```
private void SelectionChanged(object sender, EventArgs e)
{
}
```

C++
```
void OnSelectionChanged()
{
}
```

C++ Builder
```
void __fastcall SelectionChanged(TObject *Sender)
{
}
```

Delphi
```
procedure SelectionChanged(ASender: TObject; );
begin
end;
```

Delphi 8 (.NET only)
```
procedure SelectionChanged(sender: System.Object; e: System.EventArgs);
begin
end;
```

Powe…
```
begin event SelectionChanged()
end event SelectionChanged
```

VB.NET
```
Private Sub SelectionChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles SelectionChanged
End Sub
```

VB6
```
Private Sub SelectionChanged()
End Sub
```

VBA
```
Private Sub SelectionChanged()
End Sub
```

VFP
```
LPARAMETERS nop
```

Xbas…
```
PROCEDURE OnSelectionChanged(oG2antt)
RETURN
```

Syntax for SelectionChanged event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="SelectionChanged()" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function SelectionChanged()
```

```
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComSelectionChanged
    Forward Send OnComSelectionChanged
End_Procedure
```

Visual Objects

```
METHOD OCX_SelectionChanged() CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_SelectionChanged()
{
}
```

XBasic

```
function SelectionChanged as v ()
end function
```

dBASE

```
function nativeObject_SelectionChanged()
return
```

The following VB sample displays the selected items:

```
Private Sub G2antt1_SelectionChanged()
   On Error Resume Next
   Dim h As HITEM
   Dim i As Long, j As Long, nCols As Long, nSels As Long
   nCols = G2antt1.Columns.Count
   With G2antt1.Items
   nSels = .SelectCount
   For i = 0 To nSels - 1
      Dim s As String
      For j = 0 To nCols - 1
         s = s + .CellValue(.SelectedItem(i), j) + Chr(9)
      Next
   Debug.Print s
   Next
   End With
```

```
End Sub
```

The following VB sample expands programmatically items when the selection is changed:

```
Private Sub G2antt1_SelectionChanged()
   G2antt1.Items.ExpandItem(G2antt1.Items.SelectedItem()) = True
End Sub
```

The following VB sample displays the selected items:

```
Private Sub G2antt1_SelectionChanged()
   Dim i As Long
   With G2antt1.Items
      For i = 0 To .SelectCount - 1
         Debug.Print .CellValue(.SelectedItem(i), 0)
      Next
   End With
End Sub
```

The following VC sample displays the selected items:

```
#include "Items.h"

static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
   if ( pv )
   {
      if ( pv->vt == VT_ERROR )
         return szDefault;

      COleVariant vt;
      vt.ChangeType( VT_BSTR, pv );
      return V_BSTR( &vt );
   }
   return szDefault;
}

void OnSelectionChangedG2antt1()
{
```

```
    CItems items = m_g2antt.GetItems();
    for ( long i = 0; i < items.GetSelectCount(); i++ )
    {
        COleVariant vtItem( items.GetSelectedItem( i ) );
        CString strOutput;
        strOutput.Format( "%s\n", V2S( &items.GetCellValue( vtItem, COleVariant( (long)0 ) ) )
);
        OutputDebugString( strOutput );
    }
}
```

The following VB.NET sample displays the selected items:

```
Private Sub AxG2antt1_SelectionChanged(ByVal sender As Object, ByVal e As
System.EventArgs) Handles AxG2antt1.SelectionChanged
    With AxG2antt1.Items
        Dim i As Integer
        For i = 0 To .SelectCount - 1
            Debug.WriteLine(.CellValue(.SelectedItem(i), 0))
        Next
    End With
End Sub
```

The following C# sample displays the selected items:

```
private void axG2antt1_SelectionChanged(object sender, System.EventArgs e)
{
    for ( int i = 0; i < axG2antt1.Items.SelectCount - 1; i++ )
    {
        object cell = axG2antt1.Items.get_CellValue( axG2antt1.Items.get_SelectedItem( i), 0 );
        System.Diagnostics.Debug.WriteLine( cell != null ? cell.ToString() : "" );
    }
}
```

The following VFP sample displays the selected items:

```
*** ActiveX Control Event ***

with thisform.G2antt1.Items
```

```
    for i = 0 to .SelectCount - 1
        .DefaultItem = .SelectedItem( i )
        wait window nowait .CellValue( 0, 0 )
    next
endwith
```

event Sort ()

Occurs when the control sorts a column.

Type	Description

The control fires the Sort event when the control sorts a column (the user clicks the column's head) or when the sorting position is changed in the control's sort bar. Use the SortOnClick property to specify the action that control executes when the user clicks the column's head. Use the SortBarVisible property to show the control's sort bar. Use the SortOrder property to sorts a column at runtime. Use the SortPosition property to determine the position of the column in the sorting columns collection. Use the ItemBySortPosition property to access a column giving its position in the sorting columns collection. Use the Sort event to sort the data when the SortOnClk property is exUserSort. Use the SingleSort property to allow sorting by single or multiple columns.

Syntax for Sort event, **/NET** version, on:

C#
```
private void Sort(object sender)
{
}
```

VB
```
Private Sub Sort(ByVal sender As System.Object) Handles Sort
End Sub
```

Syntax for Sort event, **/COM** version, on:

C#
```
private void Sort(object sender, EventArgs e)
{
}
```

C++
```
void OnSort()
{
}
```

C++ Builder
```
void __fastcall Sort(TObject *Sender)
{
}
```

Delphi
```
procedure Sort(ASender: TObject; );
begin
```

```
end;
```

Delphi 8 (.NET only)

```
procedure Sort(sender: System.Object; e: System.EventArgs);
begin
end;
```

Powe…

```
begin event Sort()
end event Sort
```

VB.NET

```
Private Sub Sort(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Sort
End Sub
```

VB6

```
Private Sub Sort()
End Sub
```

VBA

```
Private Sub Sort()
End Sub
```

VFP

```
LPARAMETERS nop

```

Xbas…

```
PROCEDURE OnSort(oG2antt)
RETURN
```

Syntax for Sort event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="Sort()" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function Sort()
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComSort
    Forward Send OnComSort
End_Procedure
```

Visual Objects

```
METHOD OCX_Sort() CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_Sort()
{
}
```

XBasic

```
function Sort as v ()
end function
```

dBASE

```
function nativeObject_Sort()
return
```

The following VB sample displays the list of columns being sorted:

```
Private Sub G2antt1_Sort()
    Dim s As String, i As Long, c As Column
    i = 0
    With G2antt1.Columns
        Set c = .ItemBySortPosition(i)
        While (Not c Is Nothing)
            s = s + """" & c.Caption & """ " & IIf(c.SortOrder = SortAscending, "A", "D") & " "
            i = i + 1
            Set c = .ItemBySortPosition(i)
        Wend
    End With
    s = "Sort: " & s
    Debug.Print s
End Sub
```

The following VC sample displays the list of columns being sorted:

```
void OnSortG2antt1()
{
    CString strOutput;
    CColumns columns = m_g2antt.GetColumns();
    long i = 0;
    CColumn column = columns.GetItemBySortPosition( COleVariant( i ) );
```

```
   while ( column.m_lpDispatch )
   {
      strOutput += "\"" + column.GetCaption() + "\" " + ( column.GetSortOrder() == 1 ?
"A" : "D" ) + " ";
      i++;
      column = columns.GetItemBySortPosition( COleVariant( i ) );
   }
   OutputDebugString( strOutput );
}
```

The following VB.NET sample displays the list of columns being sorted:

```
Private Sub AxG2antt1_Sort(ByVal sender As Object, ByVal e As System.EventArgs) Handles
AxG2antt1.Sort
   With AxG2antt1
      Dim s As String, i As Integer, c As EXG2ANTTLib.Column
      i = 0
      With AxG2antt1.Columns
         c = .ItemBySortPosition(i)
         While (Not c Is Nothing)
            s = s + """" & c.Caption & """ " & IIf(c.SortOrder =
EXG2ANTTLib.SortOrderEnum.SortAscending, "A", "D") & " "
            i = i + 1
            c = .ItemBySortPosition(i)
         End While
      End With
      s = "Sort: " & s
      Debug.WriteLine(s)
   End With
End Sub
```

The following C# sample displays the list of columns being sorted:

```
private void axG2antt1_Sort(object sender, System.EventArgs e)
{
   string strOutput = "";
   int i = 0;
   EXG2ANTTLib.Column column = axG2antt1.Columns.get_ItemBySortPosition( i );
```

```
   while ( column != null )
   {
      strOutput += column.Caption + " " + ( column.SortOrder ==
EXG2ANTTLib.SortOrderEnum.SortAscending ? "A" : "D" ) + " ";
      column = axG2antt1.Columns.get_ItemBySortPosition( ++i );
   }
   Debug.WriteLine( strOutput );
}
```

The following VFP sample displays the list of columns being sorted (the code is listed in the Sort event) :

```
local s, i, c
i = 0
s = ""
With thisform.G2antt1.Columns
   c = .ItemBySortPosition(i)
   do While (!isnull(c))
      with c
         s = s + "'" + .Caption
         s = s + "' " + Iif(.SortOrder = 1, "A", "D") + " "
         i = i + 1
      endwith
      c = .ItemBySortPosition(i)
   enddo
endwith
s = "Sort: " + s
wait window nowait s
```

event ToolTip (Item as HITEM, ColIndex as Long, Visible as Boolean, X as Long, Y as Long, CX as Long, CY as Long)

Fired when the control prepares the object's tooltip.

Type	Description
Item as HITEM	A long expression that indicates the item's handle or 0 if the cursor is not over the cell.
ColIndex as Long	A long expression that indicates the column's index. If positive (including 0) it indicates the index of the column. If negative it indicates one of the following: • -1, if the mouse pointer hovers the levels of the chart (Level.ToolTip property) • -2, if the mouse pointer hovers the bars of the chart (ItemBarPropertyEnum,exBarToolTip property) • -3, if the mouse pointer hovers the links of the chart (LinkPropertyEnum,exLinkToolTip property) • -4, if the mouse pointer hovers the notes of the chart (Note.ToolTip property) • -5, if the mouse pointer hovers the overview section of the chart (Chart.OverviewToolTip property)
Visible as Boolean	A boolean expression that indicates whether the object's tooltip is visible.
X as Long	A long expression that indicates the left location of the tooltip window. The x values is always expressed in screen coordinates.
Y as Long	A long expression that indicates the top location of the tooltip window. The y values is always expressed in screen coordinates.
CX as Long	A long expression that indicates the width of the tooltip window.
CY as Long	A long expression that indicates the height of the tooltip window.

The ToolTip event notifies your application that the control prepares the tooltip for a cell or column. Use the ToolTip event to change the default position of the tooltip window. Use the CellToolTip property to specify the cell's tooltip. Use the Tooltip property to assign a tooltip to a column. Use the ToolTipWidth property to specify the width of the tooltip window.

Syntax for ToolTip event, **/NET** version, on:

C#

```
private void ToolTip(object sender,int Item,int ColIndex,ref bool Visible,ref int X,ref int Y,int CX,int CY)
{
}
```

VB

```
Private Sub ToolTip(ByVal sender As System.Object,ByVal Item As Integer,ByVal ColIndex As Integer,ByRef Visible As Boolean,ByRef X As Integer,ByRef Y As Integer,ByVal CX As Integer,ByVal CY As Integer) Handles ToolTip
End Sub
```

Syntax for ToolTip event, **/COM** version, on:

C#

```
private void ToolTip(object sender, AxEXG2ANTTLib._IG2anttEvents_ToolTipEvent e)
{
}
```

C++

```
void OnToolTip(long Item,long ColIndex,BOOL FAR* Visible,long FAR* X,long FAR* Y,long CX,long CY)
{
}
```

C++ Builder

```
void __fastcall ToolTip(TObject *Sender,Exg2anttlib_tlb::HITEM Item,long ColIndex,VARIANT_BOOL * Visible,long * X,long * Y,long CX,long CY)
{
}
```

Delphi

```
procedure ToolTip(ASender: TObject; Item : HITEM;ColIndex : Integer;var Visible : WordBool;var X : Integer;var Y : Integer;CX : Integer;CY : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure ToolTip(sender: System.Object; e: AxEXG2ANTTLib._IG2anttEvents_ToolTipEvent);
begin
end;
```

Powe…
```
begin event ToolTip(long Item,long ColIndex,boolean Visible,long X,long Y,long CX,long CY)
end event ToolTip
```

VB.NET
```
Private Sub ToolTip(ByVal sender As System.Object, ByVal e As AxEXG2ANTTLib._IG2anttEvents_ToolTipEvent) Handles ToolTip
End Sub
```

VB6
```
Private Sub ToolTip(ByVal Item As EXG2ANTTLibCtl.HITEM,ByVal ColIndex As Long,Visible As Boolean,X As Long,Y As Long,ByVal CX As Long,ByVal CY As Long)
End Sub
```

VBA
```
Private Sub ToolTip(ByVal Item As Long,ByVal ColIndex As Long,Visible As Boolean,X As Long,Y As Long,ByVal CX As Long,ByVal CY As Long)
End Sub
```

VFP
```
LPARAMETERS Item,ColIndex,Visible,X,Y,CX,CY

```

Xbas…
```
PROCEDURE OnToolTip(oG2antt,Item,ColIndex,Visible,X,Y,CX,CY)
RETURN
```

Syntax for ToolTip event, ***/COM*** version (others), on:

Java…
```
<SCRIPT EVENT="ToolTip(Item,ColIndex,Visible,X,Y,CX,CY)"
LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function ToolTip(Item,ColIndex,Visible,X,Y,CX,CY)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComToolTip HITEM llItem Integer llColIndex Boolean llVisible Integer llX Integer llY Integer llCX Integer llCY
    Forward Send OnComToolTip llItem llColIndex llVisible llX llY llCX llCY
End_Procedure
```

Visual Objects

```
METHOD OCX_ToolTip(Item,ColIndex,Visible,X,Y,CX,CY) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_ToolTip(int _Item,int _ColIndex,COMVariant /*bool*/
_Visible,COMVariant /*long*/ _X,COMVariant /*long*/ _Y,int _CX,int _CY)
{
}
```

XBasic

```
function ToolTip as v (Item as OLE::Exontrol.G2antt.1::HITEM,ColIndex as N,Visible
as L,X as N,Y as N,CX as N,CY as N)
end function
```

dBASE

```
function nativeObject_ToolTip(Item,ColIndex,Visible,X,Y,CX,CY)
return
```

event UserEditorClose (Object as Object, Item as HITEM, ColIndex as Long)

Fired the user editor is about to be opened.

Type	Description
Object as Object	An object created by UserEditor property.
Item as HITEM	A long expression that determines the item's handle. If the Item parameter is 0, and the ColIndex property is different than zero, the ColIndex indicates the handle of the cell where the state is changed.
ColIndex as Long	A long expression that indicates the column's index, if the Item parameter is not zero, a long expression that indicates the handle of the cell if the Item parameter is 0.

Use the UserEditorClose event to notify your application when the user editor is hidden. Use the UserEditorClose event to update the cell's value when user editor is hidden. The control fires UserEditorOleEvent event each time when a an user editor object fires an event.

Syntax for UserEditorClose event, **/NET** version, on:

C#
```
private void UserEditorClose(object sender,object Obj,int Item,int ColIndex)
{
}
```

VB
```
Private Sub UserEditorClose(ByVal sender As System.Object,ByVal Obj As Object,ByVal Item As Integer,ByVal ColIndex As Integer) Handles UserEditorClose
End Sub
```

Syntax for UserEditorClose event, **/COM** version, on:

C#
```
private void UserEditorClose(object sender,
AxEXG2ANTTLib._IG2anttEvents_UserEditorCloseEvent e)
{
}
```

C++
```
void OnUserEditorClose(LPDISPATCH Object,long Item,long ColIndex)
{
}
```

C++ Builder

```
void __fastcall UserEditorClose(TObject *Sender,IDispatch *Object,Exg2anttlib_tlb::HITEM Item,long ColIndex)
{
}
```

Delphi

```
procedure UserEditorClose(ASender: TObject; Object : IDispatch;Item : HITEM;ColIndex : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure UserEditorClose(sender: System.Object; e: AxEXG2ANTTLib._IG2anttEvents_UserEditorCloseEvent);
begin
end;
```

Powe…

```
begin event UserEditorClose(oleobject Object,long Item,long ColIndex)
end event UserEditorClose
```

VB.NET

```
Private Sub UserEditorClose(ByVal sender As System.Object, ByVal e As AxEXG2ANTTLib._IG2anttEvents_UserEditorCloseEvent) Handles UserEditorClose
End Sub
```

VB6

```
Private Sub UserEditorClose(ByVal Object As Object,ByVal Item As EXG2ANTTLibCtl.HITEM,ByVal ColIndex As Long)
End Sub
```

VBA

```
Private Sub UserEditorClose(ByVal Object As Object,ByVal Item As Long,ByVal ColIndex As Long)
End Sub
```

VFP

```
LPARAMETERS Object,Item,ColIndex

```

Xbas…

```
PROCEDURE OnUserEditorClose(oG2antt,Object,Item,ColIndex)
RETURN
```

Syntax for UserEditorClose event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="UserEditorClose(Object,Item,ColIndex)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function UserEditorClose(Object,Item,ColIndex)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComUserEditorClose Variant llObject HITEM llItem Integer llColIndex
    Forward Send OnComUserEditorClose llObject llItem llColIndex
End_Procedure
```

Visual Objects
```
METHOD OCX_UserEditorClose(Object,Item,ColIndex) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_UserEditorClose(COM _Object,int _Item,int _ColIndex)
{
}
```

XBasic
```
function UserEditorClose as v (Object as P,Item as
OLE::Exontrol.G2antt.1::HITEM,ColIndex as N)
end function
```

dBASE
```
function nativeObject_UserEditorClose(Object,Item,ColIndex)
return
```

The following VB sample updates the cell's value when the user editor is hidden (the sample handles the event for an exMaskEdit inside):

```
Private Sub G2antt1_UserEditorClose(ByVal Object As Object, ByVal Item As
EXG2ANTTLibCtl.HITEM, ByVal ColIndex As Long)
    With G2antt1.Items
        .CellValue(Item, ColIndex) = Object.Text
    End With
End Sub
```

The following C++ sample updates the cell's value when the user editor is hidden (the sample handles the event for an exMaskEdit inside):

```
#import <maskedit.dll>

#include "Items.h"

void OnUserEditorCloseG2antt1(LPDISPATCH Object, long Item, long ColIndex)
{
   MaskEditLib::IMaskEditPtr spMaskEdit( Object );
   if ( spMaskEdit != NULL )
   {
      COleVariant vtNewValue(spMaskEdit->GetText());
      m_g2antt.GetItems().SetCellValue( COleVariant( Item ), COleVariant( ColIndex ),
vtNewValue );
   }
}
```

where the #import <maskedit.dll> defines the type library of the exMaskEdit component, in the MaskEditLib namespace. The V2S function converts a VARIANT value to a string value and may look like follows:

```
static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
   if ( pv )
   {
      if ( pv->vt == VT_ERROR )
         return szDefault;

      COleVariant vt;
      vt.ChangeType( VT_BSTR, pv );
      return V_BSTR( &vt );
   }
   return szDefault;
}
```

The following VB.NET sample updates the cell's value when the user editor is hidden (the sample handles the event for an exMaskEdit inside):

```
Private Sub AxG2antt1_UserEditorClose(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_UserEditorCloseEvent) Handles
AxG2antt1.UserEditorClose
   With AxG2antt1.Items
      .CellValue(e.item, e.colIndex) = e.object.Text
   End With
End Sub
```

The following C# sample updates the cell's value when the user editor is hidden (the sample handles the event for an exMaskEdit inside):

```
private void axG2antt1_UserEditorClose(object sender,
AxEXG2ANTTLib._IG2anttEvents_UserEditorCloseEvent e)
{
   MaskEditLib.MaskEdit maskEdit = e.@object as MaskEditLib.MaskEdit;
   if (maskEdit != null)
      axG2antt1.Items.set_CellValue(e.item, e.colIndex, maskEdit.Text);
}
```

where the MaskEditLib class is defined by adding a new reference to the ExMaskEdit component to your project.

The following VFP sample updates the cell's value when the user editor is hidden (the sample handles the event for an exMaskEdit inside):

```
*** ActiveX Control Event ***
LPARAMETERS object, item, colindex

with thisform.G2antt1.Items
   .DefaultItem = item
   .CellValue( 0, colindex ) = object.Text()
endwith
```

event UserEditorOleEvent (Object as Object, Ev as OleEvent, CloseEditor as Boolean, Item as HITEM, ColIndex as Long)

Occurs when an user editor fires an event.

Type	Description
Object as Object	An object created by the UserEditor property.
Ev as OleEvent	An OleEvent object that holds information about the event
CloseEditor as Boolean	A boolean expression that indicates whether the control should close the user editor.
Item as HITEM	A long expression that determines the item's handle. If the Item parameter is 0, and the ColIndex property is different than zero, the ColIndex indicates the handle of the cell where the state is changed.
ColIndex as Long	A long expression that indicates the column's index, if the Item parameter is not zero, a long expression that indicates the handle of the cell if the Item parameter is 0.

The UserEditorOleEvent is fired every time when an user editor object fires an event. The information about fired event is stored in the Ev parameter. The CloseEditor parameter is useful to inform the control when the editor should be hidden, on certain events. The control fires the UserEditorOpen event when a ActiveX editor is about to be shown. The control fires the UserEditorClose event when the user editor is hidden.

Syntax for UserEditorOleEvent event, **/NET** version, on:

C#
```
private void UserEditorOleEvent(object sender,object
Obj,exontrol.EXG2ANTTLib.OleEvent Ev,ref bool CloseEditor,int Item,int ColIndex)
{
}
```

VB
```
Private Sub UserEditorOleEvent(ByVal sender As System.Object,ByVal Obj As
Object,ByVal Ev As exontrol.EXG2ANTTLib.OleEvent,ByRef CloseEditor As
Boolean,ByVal Item As Integer,ByVal ColIndex As Integer) Handles
UserEditorOleEvent
End Sub
```

Syntax for UserEditorOleEvent event, **/COM** version, on:

C#
```
private void UserEditorOleEvent(object sender,
```

```
AxEXG2ANTTLib._IG2anttEvents_UserEditorOleEventEvent e)
{
}
```

C++

```
void OnUserEditorOleEvent(LPDISPATCH Object,LPDISPATCH Ev,BOOL FAR*
CloseEditor,long Item,long ColIndex)
{
}
```

C++ Builder

```
void __fastcall UserEditorOleEvent(TObject *Sender,IDispatch
*Object,Exg2anttlib_tlb::IOleEvent *Ev,VARIANT_BOOL *
CloseEditor,Exg2anttlib_tlb::HITEM Item,long ColIndex)
{
}
```

Delphi

```
procedure UserEditorOleEvent(ASender: TObject; Object : IDispatch;Ev :
IOleEvent;var CloseEditor : WordBool;Item : HITEM;ColIndex : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure UserEditorOleEvent(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_UserEditorOleEventEvent);
begin
end;
```

Powe…

```
begin event UserEditorOleEvent(oleobject Object,oleobject Ev,boolean
CloseEditor,long Item,long ColIndex)
end event UserEditorOleEvent
```

VB.NET

```
Private Sub UserEditorOleEvent(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_UserEditorOleEventEvent) Handles
UserEditorOleEvent
End Sub
```

VB6

```
Private Sub UserEditorOleEvent(ByVal Object As Object,ByVal Ev As
EXG2ANTTLibCtl.IOleEvent,CloseEditor As Boolean,ByVal Item As
EXG2ANTTLibCtl.HITEM,ByVal ColIndex As Long)
End Sub
```

VBA
```
Private Sub UserEditorOleEvent(ByVal Object As Object,ByVal Ev As
Object,CloseEditor As Boolean,ByVal Item As Long,ByVal ColIndex As Long)
End Sub
```

VFP
```
LPARAMETERS Object,Ev,CloseEditor,Item,ColIndex

```

Xbas…
```
PROCEDURE OnUserEditorOleEvent(oG2antt,Object,Ev,CloseEditor,Item,ColIndex)
RETURN
```

Syntax for UserEditorOleEvent event, ***/COM*** version (others), on:

Java…
```
<SCRIPT EVENT="UserEditorOleEvent(Object,Ev,CloseEditor,Item,ColIndex)"
LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function UserEditorOleEvent(Object,Ev,CloseEditor,Item,ColIndex)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComUserEditorOleEvent Variant llObject Variant llEv Boolean
llCloseEditor HITEM llItem Integer llColIndex
    Forward Send OnComUserEditorOleEvent llObject llEv llCloseEditor llItem
llColIndex
End_Procedure
```

Visual Objects
```
METHOD OCX_UserEditorOleEvent(Object,Ev,CloseEditor,Item,ColIndex) CLASS
MainDialog
RETURN NIL
```

X++
```
void onEvent_UserEditorOleEvent(COM _Object,COM _Ev,COMVariant /*bool*/
_CloseEditor,int _Item,int _ColIndex)
{
}
```

XBasic
```
function UserEditorOleEvent as v (Object as P,Ev as
OLE::Exontrol.G2antt.1::IOleEvent,CloseEditor as L,Item as
```

```
OLE::Exontrol.G2antt.1::HITEM,ColIndex as N)
end function
```

dBASE

```
function nativeObject_UserEditorOleEvent(Object,Ev,CloseEditor,Item,ColIndex)
return
```

The following VB sample closes the editor and focus a new column when the user presses the TAB key:

```
Private Sub Grid1_UserEditorOleEvent(ByVal Object As Object, ByVal Ev As
EXGRIDLibCtl.IOleEvent, CloseEditor As Boolean, ByVal Item As EXGRIDLibCtl.HITEM, ByVal
ColIndex As Long)
    If (Ev.Name = "KeyDown") Then
        Dim iKey As Long
        iKey = Ev(0).Value
        If iKey = vbKeyTab Then
            With Grid1
                CloseEditor = True
                .FocusColumnIndex = .FocusColumnIndex + 1
                .SearchColumnIndex = .FocusColumnIndex
            End With
        End If
    End If
End Sub
```

The following VB sample closes the Exontrol.ComboBox user editor when the user selects a new value, or when it presses the Escape key. Also the sample changes the value of the cell in the control:

```
Private Sub G2antt1_UserEditorOleEvent(ByVal Object As Object, ByVal Ev As
EXG2ANTTLibCtl.IOleEvent, CloseEditor As Boolean, ByVal Item As
EXG2ANTTLibCtl.HITEM, ByVal ColIndex As Long)
    ' Closes the Exontrol.ComboBox when user changes the value in the control
    If nEvents = 0 Then
        If (Ev.Name = "Change") Then
            With G2antt1
                .BeginUpdate
                With .Items
```

```
            .CellValue(Item, ColIndex) = Object.Select(1)
            .CellValueFormat(Item, ColIndex) = exHTML
            .CellValue(Item, ColIndex) = .CellValue(Item, ColIndex) + " <fgcolor=FF0000>
[<b>changed</b>]</fgcolor>"
          End With
          .EndUpdate
        End With
        CloseEditor = True
      End If

      If (Ev.Name = "KeyPress") Then
        Dim l As Long
        l = Ev(0).Value
        If l = vbKeyEscape Then
          CloseEditor = True
        End If
      End If
    End If
End Sub
```

The following VB sample displays the event and its parameters when an user editor object fires an event:

```
Private Sub G2antt1_UserEditorOleEvent(ByVal Object As Object, ByVal Ev As
EXG2ANTTLibCtl.IOleEvent, CloseEditor As Boolean, ByVal Item As
EXG2ANTTLibCtl.HITEM, ByVal ColIndex As Long)
  Debug.Print "Event name:" & Ev.Name
  If (Ev.CountParam = 0) Then
    Debug.Print "The event has no arguments."
  Else
    Debug.Print "The event has the following arguments:"
    Dim i As Long
    For i = 0 To Ev.CountParam - 1
      Debug.Print Ev(i).Name; " = " & Ev(i).Value
    Next
  End If
End Sub
```

The following C++ sample displays the event and its parameters when an user editor object fires an event:

```
#import <exg2antt.dll> rename( "GetItems", "exGetItems" )

void OnUserEditorOleEventG2antt1(LPDISPATCH Object, LPDISPATCH Ev, BOOL FAR*
CloseEditor, long Item, long ColIndex)
{
   EXG2ANTTLib::IOleEventPtr spEvent( Ev );
   CString strOutput;
   strOutput.Format( "Event's name: %s\n", spEvent->Name.operator const char *() );
   OutputDebugString( strOutput );
   if ( spEvent->CountParam == 0 )
      OutputDebugString( "The event has no parameters." );
   else
   {
      for ( long i = 0; i < spEvent->CountParam; i++ )
      {
         EXG2ANTTLib::IOleEventParamPtr spParam = spEvent->GetParam( COleVariant( i ) );
         strOutput.Format( "Name: %s, Value: %s\n", spParam->Name.operator const char *
(), V2S( &spParam->Value ) );
         OutputDebugString( strOutput );
      }
   }
   OutputDebugString( "" );
}
```

where the #import<g2antt.dll> defines the EXG2ANTTLib namespace that exports definitions for the OleEvent and OleEventParam objects. The V2S function converts a VARIANT value to a string value and may look like follows:

```
static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
   if ( pv )
   {
      if ( pv->vt == VT_ERROR )
         return szDefault;
```

```
        COleVariant vt;
        vt.ChangeType( VT_BSTR, pv );
        return V_BSTR( &vt );
    }
    return szDefault;
}
```

The following VB.NET sample displays the event and its parameters when an user editor object fires an event:

```
Private Sub AxG2antt1_UserEditorOleEvent(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_UserEditorOleEventEvent) Handles
AxG2antt1.UserEditorOleEvent
    Debug.WriteLine("Event's name: " & e.ev.Name)
    Dim i As Long
    For i = 0 To e.ev.CountParam - 1
        Dim eP As EXG2ANTTLib.OleEventParam
        eP = e.ev(i)
        Debug.WriteLine("Name: " & e.ev.Name & " Value: " & eP.Value)
    Next
End Sub
```

The following C# sample displays the event and its parameters when an user editor object fires an event:

```
private void axG2antt1_UserEditorOleEvent(object sender,
AxEXG2ANTTLib._IG2anttEvents_UserEditorOleEventEvent e)
{
    System.Diagnostics.Debug.WriteLine("Event's name: " + e.ev.Name.ToString());
    for (int i = 0; i < e.ev.CountParam; i++)
    {
        EXG2ANTTLib.IOleEventParam evP = e.ev[i];
        System.Diagnostics.Debug.WriteLine("Name: " + evP.Name.ToString() + ", Value: " +
evP.Value.ToString());
    }
}
```

The following VFP sample displays the event and its parameters when an user editor object fires an event:

```
*** ActiveX Control Event ***
LPARAMETERS object, ev, closeeditor, item, colindex

local s
s = "Event's name: " + ev.Name
for i = 0 to ev.CountParam - 1
	s = s +  "Name: " + ev.Param(i).Name + " ,Value: " + Str(ev.Param(i).Value)
endfor
wait window nowait s
```

event UserEditorOpen (Object as Object, Item as HITEM, ColIndex as Long)

Occurs when an user editor is about to be opened.

Type	Description
Object as Object	An object created by UserEditor property
Item as HITEM	A long expression that determines the item's handle. If the Item parameter is 0, and the ColIndex property is different than zero, the ColIndex indicates the handle of the cell where the state is changed.
ColIndex as Long	A long expression that indicates the column's index, if the Item parameter is not zero, a long expression that indicates the handle of the cell if the Item parameter is 0.

The control supports custom ActiveX editors support. The control fires the UserEditorOpen event when an user editor is shown. Use the UserEditorOpen event to initialize the user editor when it is shown. For instance, if you have a custom maskedit control you can initialize the mask and the value based on the cell's value property. Use the CellValue property to access the cell's value. The control fires the UserEditorOleEvent event each time when an user editor fires an event. The control fires the UserEditorClose event when an user editor is hidden.

Syntax for UserEditorOpen event, **/NET** version, on:

C#
```
private void UserEditorOpen(object sender,object Obj,int Item,int ColIndex)
{
}
```

VB
```
Private Sub UserEditorOpen(ByVal sender As System.Object,ByVal Obj As Object,ByVal Item As Integer,ByVal ColIndex As Integer) Handles UserEditorOpen
End Sub
```

Syntax for UserEditorOpen event, **/COM** version, on:

C#
```
private void UserEditorOpen(object sender,
AxEXG2ANTTLib._IG2anttEvents_UserEditorOpenEvent e)
{
}
```

C++

```
void OnUserEditorOpen(LPDISPATCH Object,long Item,long ColIndex)
{
}
```

C++ Builder

```
void __fastcall UserEditorOpen(TObject *Sender,IDispatch
*Object,Exg2anttlib_tlb::HITEM Item,long ColIndex)
{
}
```

Delphi

```
procedure UserEditorOpen(ASender: TObject; Object : IDispatch;Item :
HITEM;ColIndex : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure UserEditorOpen(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_UserEditorOpenEvent);
begin
end;
```

Powe…

```
begin event UserEditorOpen(oleobject Object,long Item,long ColIndex)
end event UserEditorOpen
```

VB.NET

```
Private Sub UserEditorOpen(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_UserEditorOpenEvent) Handles UserEditorOpen
End Sub
```

VB6

```
Private Sub UserEditorOpen(ByVal Object As Object,ByVal Item As
EXG2ANTTLibCtl.HITEM,ByVal ColIndex As Long)
End Sub
```

VBA

```
Private Sub UserEditorOpen(ByVal Object As Object,ByVal Item As Long,ByVal
ColIndex As Long)
End Sub
```

VFP

```
LPARAMETERS Object,Item,ColIndex
```

Xbas…
```
PROCEDURE OnUserEditorOpen(oG2antt,Object,Item,ColIndex)
RETURN
```

Syntax for UserEditorOpen event, **/COM** version (others), on:

Java…
```
<SCRIPT EVENT="UserEditorOpen(Object,Item,ColIndex)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function UserEditorOpen(Object,Item,ColIndex)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComUserEditorOpen Variant llObject HITEM llItem Integer llColIndex
    Forward Send OnComUserEditorOpen llObject llItem llColIndex
End_Procedure
```

Visual Objects
```
METHOD OCX_UserEditorOpen(Object,Item,ColIndex) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_UserEditorOpen(COM _Object,int _Item,int _ColIndex)
{
}
```

XBasic
```
function UserEditorOpen as v (Object as P,Item as
OLE::Exontrol.G2antt.1::HITEM,ColIndex as N)
end function
```

dBASE
```
function nativeObject_UserEditorOpen(Object,Item,ColIndex)
return
```

The following VB sample selects an item into an user editor of EXCOMBOBOXLibCtl.ComboBox type (the sample uses the Exontrol's ExComboBox Component):

```
Private Sub G2antt1_UserEditorOpen(ByVal Object As Object, ByVal Item As
EXG2ANTTLibCtl.HITEM, ByVal ColIndex As Long)
  On Error Resume Next
```

```
    nEvents = nEvents + 1
    'Selects the value in the combo box
    With Object          ' Points to an EXCOMBOBOXLibCtl.ComboBox object
        Dim sID As String
        sID = G2antt1.Items.CellValue(Item, ColIndex)
        If (G2antt1.Items.CellValueFormat(Item, ColIndex) = exHTML) Then
            sID = Mid(sID, 1, InStr(1, sID, " ", vbTextCompare) - 1)
        End If
        .Select(1) = sID
        If .Items.SelectCount > 0 Then
            .Items.EnsureVisibleItem .Items.SelectedItem(0)
        End If
    End With
    nEvents = nEvents - 1
End Sub
```

The following samples use the Exontrol's ExMaskEdit Component to mask floating point numbers using digit grouping.

The following VB sample initializes the mask's value when user editor is shown (the sample calls the Text property of the exMaskEdit component):

```
Private Sub G2antt1_UserEditorOpen(ByVal Object As Object, ByVal Item As
EXG2ANTTLibCtl.HITEM, ByVal ColIndex As Long)
    With G2antt1.Items
        Object.Text = .CellValue(Item, ColIndex)
    End With
End Sub
```

The following C++ sample initializes the mask's value when user editor is shown:

```
#import <maskedit.dll>

#include "Items.h"

void OnUserEditorOpenG2antt1(LPDISPATCH Object, long Item, long ColIndex)
{
    MaskEditLib::IMaskEditPtr spMaskEdit( Object );
    if ( spMaskEdit != NULL )
```

```
    {
        CString strValue = V2S( &m_g2antt.GetItems().GetCellValue( COleVariant( Item ),
COleVariant( ColIndex ) ) );
        spMaskEdit->PutText( strValue.AllocSysString() );
    }
}
```

where the #import <maskedit.dll> defines the type library of the exMaskEdit component, in the MaskEditLib namespace. The V2S function converts a VARIANT value to a string value and may look like follows:

```
static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
    if ( pv )
    {
        if ( pv->vt == VT_ERROR )
            return szDefault;

        COleVariant vt;
        vt.ChangeType( VT_BSTR, pv );
        return V_BSTR( &vt );
    }
    return szDefault;
}
```

The following VB.NET sample initializes the mask's value when user editor is shown:

```
Private Sub AxG2antt1_UserEditorOpen(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_UserEditorOpenEvent) Handles
AxG2antt1.UserEditorOpen
    With AxG2antt1.Items
        e.object.Text = .CellValue(e.item, e.colIndex)
    End With
End Sub
```

The following C# sample initializes the mask's value when user editor is shown:

```
private void axG2antt1_UserEditorOpen(object sender,
AxEXG2ANTTLib._IG2anttEvents_UserEditorOpenEvent e)
```

```
{
    MaskEditLib.MaskEdit maskEdit = e.@object as MaskEditLib.MaskEdit;
    if (maskEdit != null)
    {
        object cellValue = axG2antt1.Items.get_CellValue(e.item, e.colIndex);
        maskEdit.Text = (cellValue != null ? cellValue.ToString() : "");
    }
}
```

where the MaskEditLib class is defined by adding a new reference to the ExMaskEdit component to your project.

The following VFP sample initializes the mask's value when user editor is shown:

```
*** ActiveX Control Event ***
LPARAMETERS object, item, colindex

with thisform.G2antt1.Items
    .DefaultItem = item
    object.Text = .CellValue( 0, colindex )
endwith
```

event ValidateValue (Item as HITEM, ColIndex as Long, NewValue as Variant, Cancel as Boolean)

Occurs before user changes the cell's value.

Type	Description
Item as HITEM	A long expression that determines the item's handle. If the Item parameter is 0, and the ColIndex property is different than zero, the ColIndex indicates the handle of the cell where the change occurs.
ColIndex as Long	A long expression that indicates the column's index, if the Item parameter is not zero, a long expression that indicates the handle of the cell if the Item parameter is 0.
NewValue as Variant	A Variant value that indicates the value being validated.
Cancel as Boolean	A boolean expression that indicates whether the value is valid or not. By default, the Cancel parameter is False, and so the NewValue parameter is valid. If the Cancel parameter is set on True, the control considers the NewValue being a non valid value, so the Change event is not fired.

The ValidateValue event notifies your application that the user is about to change the cell's value using the control's UI. Use the ValidateValue event to prevent users enter wrong values to the cells. The ValidateValue event is fired **only** if the CauseValidateValue property is not zero and the user alters the focused value. The validation can be done per cell or per item, in other words, the validation can be made if the user leaves the focused cell, or focused item. If the Cancel parameter is True, the user can't move the focus to a new cell/item, until the Cancel parameter is False. If the Cancel parameter is False the control fires the Change event to notify your application that the cell's value is changed. Use the Edit method to programmatically edit the focused cell. Call the DiscardValidateValue method to restore back the values being changed during the validation.

During ValidateValue event, the Items.CellValue(Item,ColIndex) and Items.CellCaption(Item,ColIndex) properties retrieve the original value/caption of the cell. You can access the modified value for any cell in the validating item using the Items.CellValue(-1,ColIndex) and Items.CellCaption(-1,ColIndex), or uses the -1 identifier for the Item parameter of the Items.CellValue and Items.CellCaption properties.

During the validation you may have the following order of the events:

- Edit - prevent showing the editor for specified cell.
- EditOpen - indicates that the editor for the focused cell is being opened.

- EditClose - indicates that the editor for the focused cell is being closed.
- ValidateValue - notifies your application that the value must be validated (Cancel parameter on False)
- Change - notifies the application once the user validates the newly value. In case the control is bounded to a database, the change is performed to the database too.
- Error - notifies the application for any error (for instance, if the change is not supported by the database, the Error indicates the error being issued).

The ValidateValue event is not fired if the CellValue property is called during the event.

Syntax for ValidateValue event, **/NET** version, on:

C#
```
private void ValidateValue(object sender,int Item,int ColIndex,object NewValue,ref bool Cancel)
{
}
```

VB
```
Private Sub ValidateValue(ByVal sender As System.Object,ByVal Item As Integer,ByVal ColIndex As Integer,ByVal NewValue As Object,ByRef Cancel As Boolean) Handles ValidateValue
End Sub
```

Syntax for ValidateValue event, **/COM** version, on:

C#
```
private void ValidateValue(object sender, AxEXG2ANTTLib._IG2anttEvents_ValidateValueEvent e)
{
}
```

C++
```
void OnValidateValue(long Item,long ColIndex,VARIANT NewValue,BOOL FAR* Cancel)
{
}
```

C++ Builder
```
void __fastcall ValidateValue(TObject *Sender,Exg2anttlib_tlb::HITEM Item,long ColIndex,Variant NewValue,VARIANT_BOOL * Cancel)
{
}
```

Delphi
```
procedure ValidateValue(ASender: TObject; Item : HITEM;ColIndex :
```

```
Integer;NewValue : OleVariant;var Cancel : WordBool);
begin
end;
```

Delphi 8 (.NET only)

```
procedure ValidateValue(sender: System.Object; e:
AxEXG2ANTTLib._IG2anttEvents_ValidateValueEvent);
begin
end;
```

Powe…

```
begin event ValidateValue(long Item,long ColIndex,any NewValue,boolean Cancel)
end event ValidateValue
```

VB.NET

```
Private Sub ValidateValue(ByVal sender As System.Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_ValidateValueEvent) Handles ValidateValue
End Sub
```

VB6

```
Private Sub ValidateValue(ByVal Item As EXG2ANTTLibCtl.HITEM,ByVal ColIndex As
Long,ByVal NewValue As Variant,Cancel As Boolean)
End Sub
```

VBA

```
Private Sub ValidateValue(ByVal Item As Long,ByVal ColIndex As Long,ByVal
NewValue As Variant,Cancel As Boolean)
End Sub
```

VFP

```
LPARAMETERS Item,ColIndex,NewValue,Cancel

```

Xbas…

```
PROCEDURE OnValidateValue(oG2antt,Item,ColIndex,NewValue,Cancel)
RETURN
```

Syntax for ValidateValue event, ***/COM*** version (others), on:

Java…

```
<SCRIPT EVENT="ValidateValue(Item,ColIndex,NewValue,Cancel)"
LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function ValidateValue(Item,ColIndex,NewValue,Cancel)
```

```
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComValidateValue HITEM llItem Integer llColIndex Variant
llNewValue Boolean llCancel
    Forward Send OnComValidateValue llItem llColIndex llNewValue llCancel
End_Procedure
```

Visual Objects

```
METHOD OCX_ValidateValue(Item,ColIndex,NewValue,Cancel) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_ValidateValue(int _Item,int _ColIndex,COMVariant
_NewValue,COMVariant /*bool*/ _Cancel)
{
}
```

XBasic

```
function ValidateValue as v (Item as OLE::Exontrol.G2antt.1::HITEM,ColIndex as
N,NewValue as A,Cancel as L)
end function
```

dBASE

```
function nativeObject_ValidateValue(Item,ColIndex,NewValue,Cancel)
return
```

The following VB sample asks the user to validate the value for **each** cell that's edited:

```
Private Sub G2antt_ValidateValue(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal ColIndex As
Long, ByVal NewValue As Variant, Cancel As Boolean)
   Cancel = True ' Causes all cells to be invalid
   If MsgBox("The ValidateValue event just occurs. Do the change with this value '" &
NewValue & "'?", vbYesNo) = vbYes Then
      Cancel = False ' Only cells where user selects the Yes button, are valid
   End If
   G2antt.Edit ' Continue editing a cell.
End Sub
```

The following C++ sample asks the user to validate the value for **each** cell that's edited:

```
CString V2S( const VARIANT* pvtValue )
```

```
{
   COleVariant vt;
   vt.ChangeType( VT_BSTR, (LPVARIANT)pvtValue );
   return V_BSTR( &vt );
}

void OnValidateValueG2antt1(long Item, long ColIndex, const VARIANT FAR& NewValue,
BOOL FAR* Cancel)
{
   *Cancel = TRUE;// Causes all cells to be invalid
   if ( MessageBox( "The ValidateValue event just occurs. Do the change with this value '" +
V2S( &NewValue ) + "'?", "Information", MB_YESNO ) == IDYES )
      *Cancel = FALSE; // Only cells where user selects the Yes button, are valid
   COleVariant vtOptional; V_VT(&vtOptional) = VT_ERROR;
   m_g2antt.Edit( vtOptional ); // Continue editing a cell.
}
```

The following C++ sample asks the user to enter a value greater than 10 on the first column, if the value is less than 10:

```
Private Sub G2antt_ValidateValue(ByVal Item As EXG2ANTTLibCtl.HITEM, ByVal ColIndex As
Long, ByVal NewValue As Variant, Cancel As Boolean)
   If (ColIndex = 0) Then
      If (NewValue < 10) Then
         MsgBox "Enter a value greater than 10."
         Cancel = True ' Cancels only the cells with the value less than 10.
         G2antt.Edit ' Continue editing a cell.
      End If
   End If
End Sub
```

The following VB.NET sample asks the user to validate the values on the first column:

```
Private Sub AxG2antt1_ValidateValue(ByVal sender As Object, ByVal e As
AxEXG2ANTTLib._IG2anttEvents_ValidateValueEvent) Handles AxG2antt1.ValidateValue
   If (e.colIndex = 0) Then
      e.cancel = True
      Dim strMessage As String = "The ValidateValue event just occurs. Do the change with
```

```
this value '" & e.newValue.ToString() & "'?"
      If (MessageBox.Show(strMessage, "Question", MessageBoxButtons.YesNoCancel,
MessageBoxIcon.Question) = Windows.Forms.DialogResult.Yes) Then
         e.cancel = False
      End If
   End If
End Sub
```

The following C# sample asks the user to validate the values on the first column:

```
private void axG2antt1_ValidateValue(object sender,
AxEXG2ANTTLib._IG2anttEvents_ValidateValueEvent e)
{
   if (e.colIndex == 0)
   {
      e.cancel = true;
      string strMessage = "The ValidateValue event just occurs. Do the change with this
value '" + e.newValue.ToString() + "'?";
      if ( MessageBox.Show(strMessage, "Question", MessageBoxButtons.YesNoCancel,
MessageBoxIcon.Question) == DialogResult.Yes )
         e.cancel = false;
   }
}
```

The following VFP sample asks the user to validate the values on the first column:

```
*** ActiveX Control Event ***
LPARAMETERS item, colindex, newvalue, cancel

with thisform.G2antt1.Items
   if ( colindex = 0 )
      cancel = .t.
      local strMessage
      strMessage = "The ValidateValue event just occurs. Do the change with this value '" +
newvalue + "'?"
      if ( MessageBox( strMessage, 3 ) = 6 )
         cancel = .f.
      endif
```

```
    endif
endwith
```

Expressions

An expression is a string which defines a formula or criteria, that's evaluated at runtime. The expression may be a combination of variables, constants, strings, dates and operators/functions. For instance 1000 format `` gets 1,000.00 for US format, while 1.000,00 is displayed for German format.

The Exontrol's **eXPression** component is a syntax-editor that helps you to define, view, edit and evaluate expressions. Using the eXPression component you can easily view or check if the expression you have used is syntactically correct, and you can evaluate what is the result you get giving different values to be tested. The Exontrol's eXPression component can be used as an user-editor, to configure your applications.

Usage examples:

- 100 + 200, adds two numbers and returns 300
- "100" + 200, concatenates the strings, and returns "100200"
- currency(1000) displays the value in currency format based on the current regional setting, such as "$1,000.00" for US format.
- 1000 format `` gets 1,000.00 for English format, while 1.000,00 is displayed for German format
- 1000 format `2|.|3|,` always gets 1,000.00 no matter of settings in the control panel.
- date(value) format `MMM d, yyyy` , returns the date such as Sep 2, 2023, for English format
- upper("string") converts the giving string in uppercase letters, such as "STRING"
- date(dateS('3/1/' + year(9:=#1/1/2018#)) + ((1:=(((255 - 11 * (year(=:9) mod 19)) - 21) mod 30) + 21) + (=:1 > 48 ? -1 : 0) + 6 - ((year(=:9) + int(year(=:9) / 4)) + =:1 + (=:1 > 48 ? -1 : 0) + 1) mod 7)) returns the date the Easter Sunday will fall, for year 2018. In this case the expression returns #4/1/2018#. If #1/1/2018# is replaced with #1/1/2019#, the expression returns #4/21/2019#.

Listed bellow are all predefined constants, operators and functions the general-expression supports:

The constants can be represented as:

- numbers in **decimal** format (where dot character specifies the decimal separator). For instance: -1, 100, 20.45, .99 and so on
- numbers in **hexa-decimal** format (preceded by **0x** or **0X** sequence), uses sixteen distinct symbols, most often the symbols 0-9 to represent values zero to nine, and A, B, C, D, E, F (or alternatively a, b, c, d, e, f) to represent values ten to fifteen. Hexadecimal numerals are widely used by computer system designers and

programmers. As each hexadecimal digit represents four binary digits (bits), it allows a more human-friendly representation of binary-coded values. For instance, 0xFF, 0x00FF00, and so so.
- **date-time** in format **#mm/dd/yyyy hh:mm:ss#**, For instance, #1/31/2001 10:00# means the January 31th, 2001, 10:00 AM
- **string**, if it starts / ends with any of the **'** or ` or **"** characters. If you require the starting character inside the string, it should be escaped ( preceded by a \ character ). For instance, `Mihai`, "Filimon", 'has', "\"a quote\"", and so on

The predefined constants are:

- **bias** (BIAS constant), defines the difference, in minutes, between Coordinated Universal Time (UTC) and local time. For example, Middle European Time (MET, GMT+01:00) has a time zone bias of "-60" because it is one hour ahead of UTC. Pacific Standard Time (PST, GMT-08:00) has a time zone bias of "+480" because it is eight hours behind UTC. For instance, date(value - bias/24/60) converts the UTC time to local time, or date(date('now') + bias/24/60) converts the current local time to UTC time. For instance, "date(value - bias/24/60)" converts the value date-time from UTC to local time, while "date(value + bias/24/60)" converts the local-time to UTC time.
- **dpi** (DPI constant), specifies the current DPI setting. and it indicates the minimum value between **dpix** and **dpiy** constants. For instance, if current DPI setting is 100%, the dpi constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpi returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpix** (DPIX constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpix constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpix returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpiy** (DPIY constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpiy constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpiy returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%

The supported binary arithmetic operators are:

- ***** (multiplicity operator), priority 5
- **/** (divide operator), priority 5
- **mod** (reminder operator), priority 5
- **+** (addition operator), priority 4 (concatenates two strings, if one of the operands is of string type)
- **-** (subtraction operator), priority 4

The supported unary boolean operators are:

- **not** (not operator), priority 3 (high priority)

The supported binary boolean operators are:

- **or** (or operator), priority 2
- **and** (or operator), priority 1

The supported binary boolean operators, all these with the same priority 0, are :

- **<** (less operator)
- **<=** (less or equal operator)
- **=** (equal operator)
- **!=** (not equal operator)
- **>=** (greater or equal operator)
- **>** (greater operator)

The supported binary range operators, all these with the same priority 5, are :

- a **MIN** b (min operator), indicates the minimum value, so a **MIN** b returns the value of a, if it is less than b, else it returns b. For instance, the expression value MIN 10 returns always a value greater than 10.
- a **MAX** b (max operator), indicates the maximum value, so a **MAX** b returns the value of a, if it is greater than b, else it returns b. For instance, the expression value MAX 100 returns always a value less than 100.

The supported binary operators, all these with the same priority 0, are :

- **:= (Store operator),** stores the result of expression to variable. The syntax for := operator is

  variable := expression

  where variable is a integer between 0 and 9. You can use the **=:** operator to restore any stored variable (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

- **=: (Restore operator),** restores the giving variable (previously saved using the store operator). The syntax for =: operator is

  =: variable

where variable is a integer between 0 and 9. You can use the **:=** operator to store the value of any expression (please make the difference between := and =:). For instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

The supported ternary operators, all these with the same priority 0, are :

- **?** (**Immediate If operator**), returns and executes one of two expressions, depending on the evaluation of an expression. The syntax for *?* operator is

expression ? true_part : false_part

, while it executes and returns the true_part if the expression is true, else it executes and returns the false_part. For instance, the *%0 = 1 ? 'One' : (%0 = 2 ? 'Two' : 'not found')* returns 'One' if the value is 1, 'Two' if the value is 2, and 'not found' for any other value. A n-ary equivalent operation is the case() statement, which is available in newer versions of the component.

The supported n-ary operators are (with priority 5):

- ***array*** *(at operator),* returns the element from an array giving its index (0 base). The *array* operator returns empty if the element is found, else the associated element in the collection if it is found. The syntax for *array* operator is

expression array (c1,c2,c3,...cn)

, where the c1, c2, ... are constant elements. The constant elements could be numeric, date or string expressions. For instance the *month(value)-1 array ('J','F','M','A','M','Jun','J','A','S','O','N','D')* is equivalent with *month(value)-1 case (default:''; 0:'J';1:'F';2:'M';3:'A';4:'M';5:'Jun';6:'J';7:'A';8:'S';9:'O';10:'N';11:'D')*.

- ***in*** *(include operator),* specifies whether an element is found in a set of constant elements. The *in* operator returns -1 (True) if the element is found, else 0 (false) is retrieved. The syntax for *in* operator is

expression in (c1,c2,c3,...cn)

, where the c1, c2, ... are constant elements. The constant elements could be numeric, date or string expressions. For instance the *value in (11,22,33,44,13)* is equivalent with *(expression = 11) or (expression = 22) or (expression = 33) or (expression = 44) or (expression = 13)*. The *in* operator is not a time consuming as the equivalent *or* version is, so when you have large number of constant elements it is recommended using the

in operator. Shortly, if the collection of elements has 1000 elements the *in* operator could take up to 8 operations in order to find if an element fits the set, else if the *or* statement is used, it could take up to 1000 operations to check, so by far, the *in* operator could save time on finding elements within a collection.

- ***switch*** *(switch operator),* returns the value being found in the collection, or a predefined value if the element is not found (default). The syntax for *switch* operator is

expression switch (default,c1,c2,c3,...,cn)

, where the c1, c2, ... are constant elements, and the default is a constant element being returned when the element is not found in the collection. The constant elements could be numeric, date or string expressions. The equivalent syntax is "%0 = c 1 ? c 1 : (%0 = c 2 ? c 2 : (... ? . : default))". The *switch* operator is very similar with the *in* operator excepts that the first element in the switch is always returned by the statement if the element is not found, while the returned value is the value itself instead -1. For instance, the *%0 switch ('not found',1,4,7,9,11)* gets 1, 4, 7, 9 or 11, or 'not found' for any other value. As the *in* operator the *switch* operator uses binary searches for fitting the element, so it is quicker that iif (immediate if operator) alterative.

- ***case()*** *(case operator)* returns and executes one of n expressions, depending on the evaluation of the expression (IIF - immediate IF operator is a binary case() operator). The syntax for *case()* operator is:

expression case ([default : default_expression ;] c1 : expression1 ; c2 : expression2 ; c3 : expression3 ;....)

If the default part is missing, the case() operator returns the value of the expression if it is not found in the collection of cases (c1, c2, ...). For instance, if the value of expression is not any of c1, c2, ... the default_expression is executed and returned. If the value of the expression is c1, then the *case()* operator executes and returns the *expression1.* The *default, c1, c2, c3, ...* must be constant elements as numbers, dates or strings. For instance, the *date(shortdate(value)) case (default:0 ; #1/1/2002#:1 ; #2/1/2002#:1; #4/1/2002#:1; #5/1/2002#:1)* indicates that only *#1/1/2002#, #2/1/2002#, #4/1/2002# and #5/1/2002#* dates returns 1, since the others returns 0. For instance the following sample specifies the hour being non-working for specified dates: *date(shortdate(value)) case(default:0;#4/1/2009# : hour(value) >= 6 and hour(value) <= 12 ; #4/5/2009# : hour(value) >= 7 and hour(value) <= 10 or hour(value) in(15,16,18,22); #5/1/2009# : hour(value) <= 8)* statement indicates the working hours for dates as follows:

  - #4/1/2009#, from hours 06:00 AM to 12:00 PM
  - #4/5/2009#, from hours 07:00 AM to 10:00 AM and hours 03:00PM,

04:00PM, 06:00PM and 10:00PM
- #5/1/2009#, from hours 12:00 AM to 08:00 AM

The *in*, *switch* and *case()* use binary search to look for elements so they are faster then using iif and or expressions. Obviously, the priority of the operations inside the expression is determined by ( ) parenthesis and the priority for each operator.

The supported conversion unary operators are:

- **type** (unary operator) retrieves the type of the object. The type operator may return any of the following: 0 - empty (not initialized), 1 - null, 2 - short, 3 - long, 4 - float, 5 - double, 6 - currency, **7 - date**, **8 - string**, 9 - object, 10 - error, **11 - boolean**, 12 - variant, 13 - any, 14 - decimal, 16 - char, 17 - byte, 18 - unsigned short, 19 - unsigned long, 20 - long on 64 bits, 21 - unsigned long on 64 bites. For instance *type(%1) = 8* specifies the cells (on the column with the index 1) that contains string values.
- **str** (unary operator) converts the expression to a string. The str operator converts the expression to a string. For instance, the *str(-12.54)* returns the string "-12.54".
- **dbl** (unary operator) converts the expression to a number. The dbl operator converts the expression to a number. For instance, the *dbl("12.54")* returns 12.54
- **date** (unary operator) converts the expression to a date, based on your regional settings. For instance, the *date(``)* gets the current date (no time included), the *date(`now`)* gets the current date-time, while the *date("01/01/2001")* returns #1/1/2001#
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS. For instance, the *dateS("01/01/2001 14:00:00")* returns #1/1/2001 14:00:00#
- **hex** (unary operator) converts the giving string from hexa-representation to a numeric value, or converts the giving numeric value to hexa-representation as string. For instance, hex(`FF`) returns 255, while the hex(255) or hex(0xFF) returns the `FF` string. The hex(hex(`FFFFFFFF`)) always returns `FFFFFFFF` string, as the second hex call converts the giving string to a number, and the first hex call converts the returned number to string representation (hexa-representation).

The bitwise operators for numbers are:

- a **bitand** b (binary operator) computes the AND operation on bits of a and b, and returns the unsigned value. For instance, 0x01001000 bitand 0x10111000 returns 0x00001000.
- a **bitor** b (binary operator) computes the OR operation on bits of a and b, and returns the unsigned value. For instance, 0x01001000 bitor 0x10111000 returns 0x11111000.
- a **bitxor** b (binary operator) computes the XOR (exclusive-OR) operation on bits of a and b, and returns the unsigned value. For instance, 0x01110010 bitxor 0x10101010 returns 0x11011000.

- a **bitshift** (b) (binary operator) shifts every bit of a value to the left if b is negative, or to the right if b is positive, for b times, and returns the unsigned value. For instance, 128 bitshift 1 returns 64 (dividing by 2) or 128 bitshift (-1) returns 256 (multiplying by 2)
- **bitnot** (unary operator) flips every bit of x, and returns the unsigned value. For instance, bitnot(0x00FF0000) returns 0xFF00FFFF.

The operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number. For instance, the *int(12.54)* returns 12
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2. For instance, the *round(12.54)* returns 13
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument. For instance, the *floor(12.54)* returns 12
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2. For instance, the *abs(-12.54)* returns 12.54
- **sin** (unary operator) returns the sine of an angle of x radians. For instance, the *sin(3.14)* returns 0.001593.
- **cos** (unary operator) returns the cosine of an angle of x radians. For instance, the *cos(3.14)* returns -0.999999.
- **asin** (unary operator) returns the principal value of the arc sine of x, expressed in radians. For instance, the *2*asin(1)* returns the value of PI.
- **acos** (unary operator) returns the principal value of the arc cosine of x, expressed in radians. For instance, the *2*acos(0)* returns the value of PI
- **sqrt** (unary operator) returns the square root of x. For instance, the *sqrt(81)* returns 9.
- **currency** (unary operator) formats the giving number as a currency string, as indicated by the control panel. For instance, *currency(value)* displays the value using the current format for the currency ie, 1000 gets displayed as $1,000.00, for US format.
- value **format** 'flags' (binary operator) formats a numeric value with specified flags. The format method formats numeric or date expressions (depends on the type of the value, explained at operators for dates). If flags is empty, the number is displayed as shown in the field "Number" in the "Regional and Language Options" from the Control Panel. For instance the "*1000 format ""* displays 1,000.00 for English format, while 1.000,00 is displayed for German format. "1000 format '2|.|3|,'" will always displays 1,000.00 no matter of the settings in your control panel. If formatting the number fails for some invalid parameter, the value is displayed with no formatting.

  The ' flags' for format operator is a list of values separated by | character such as '*NumDigits*|*DecimalSep*|*Grouping*|*ThousandSep*|*NegativeOrder*|*LeadingZero*' with the following meanings:

  - *NumDigits* - specifies the number of fractional digits, If the flag is missing, the

field "No. of digits after decimal" from "Regional and Language Options" is using.
    - *DecimalSep* - specifies the decimal separator. If the flag is missing, the field "Decimal symbol" from "Regional and Language Options" is using.
    - Grouping - indicates the number of digits in each group of numbers to the left of the decimal separator. Values in the range 0 through 9 and 32 are valid. The most significant grouping digit indicates the number of digits in the least significant group immediately to the left of the decimal separator. Each subsequent grouping digit indicates the next significant group of digits to the left of the previous group. If the last value supplied is not 0, the remaining groups repeat the last group. Typical examples of settings for this member are: 0 to group digits as in 123456789.00; 3 to group digits as in 123,456,789.00; and 32 to group digits as in 12,34,56,789.00. If the flag is missing, the field "Digit grouping" from "Regional and Language Options" indicates the grouping flag.
    - *ThousandSep* - specifies the thousand separator. If the flag is missing, the field "Digit grouping symbol" from "Regional and Language Options" is using.
    - *NegativeOrder* - indicates the negative number mode. If the flag is missing, the field "Negative number format" from "Regional and Language Options" is using. The valid values are 0, 1, 2, 3 and 4 with the following meanings:
        - 0 - Left parenthesis, number, right parenthesis; for example, (1.1)
        - 1 - Negative sign, number; for example, -1.1
        - 2 - Negative sign, space, number; for example, - 1.1
        - 3 - Number, negative sign; for example, 1.1-
        - 4 - Number, space, negative sign; for example, 1.1 -
    - *LeadingZero* - indicates if leading zeros should be used in decimal fields. If the flag is missing, the field "Display leading zeros" from "Regional and Language Options" is using. The valid values are 0, 1

The operators for strings are:

- **len** (unary operator) retrieves the number of characters in the string. For instance, the *len("Mihai")* returns 5.
- **lower** (unary operator) returns a string expression in lowercase letters. For instance, the *lower("MIHAI")* returns "mihai"
- **upper** (unary operator) returns a string expression in uppercase letters. For instance, the *upper("mihai")* returns "MIHAI"
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names. For instance, the *proper("mihai")* returns "Mihai"
- **ltrim** (unary operator) removes spaces on the left side of a string. For instance, the *ltrim(" mihai")* returns "mihai"
- **rtrim** (unary operator) removes spaces on the right side of a string. For instance, the *rtrim("mihai ")* returns "mihai"

- **trim** (unary operator) removes spaces on both sides of a string. For instance, the *trim(" mihai ")* returns "mihai"
- **reverse** (unary operator) reverses the order of the characters in the string a. For instance, the *reverse("Mihai")* returns "iahiM"
- a **startwith** b (binary operator) specifies whether a string starts with specified string (0 if not found, -1 if found). For instance *"Mihai" startwith "Mi"* returns -1
- a **endwith** b (binary operator) specifies whether a string ends with specified string (0 if not found, -1 if found). For instance *"Mihai" endwith "ai"* returns -1
- a **contains** b (binary operator) specifies whether a string contains another specified string (0 if not found, -1 if found). For instance *"Mihai" contains "ha"* returns -1
- a **left** b (binary operator) retrieves the left part of the string. For instance *"Mihai" left 2* returns "Mi".
- a **right** b (binary operator) retrieves the right part of the string. For instance *"Mihai" right 2* returns "ai"
- a **lfind** b (binary operator) The a lfind b (binary operator) searches the first occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" lfind "C"* returns 2
- a **rfind** b (binary operator) The a rfind b (binary operator) searches the last occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" rfind "C"* returns 5.
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on). For instance *"Mihai" mid 2* returns "ihai"
- a **count** b (binary operator) retrieves the number of occurrences of the b in a. For instance *"Mihai" count "i"* returns 2.
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result. For instance, the *"Mihai" replace "i" with ""* returns "Mha" string, as it replaces all "i" with nothing.
- a **split** b (binary operator) splits the a using the separator b, and returns an array. For instance, the *weekday(value) array 'Sun Mon Thu Wed Thu Fri Sat' split ' '* gets the weekday as string. This operator can be used with the array.
- a **like** b (binary operator) compares the string a against the pattern b. The pattern b may contain wild-characters such as *, ?, # or [] and can have multiple patterns separated by space character. In order to have the space, or any other wild-character inside the pattern, it has to be escaped, or in other words it should be preceded by a \ character. For instance value like `F*e` matches all strings that start with F and ends on e, or value like `a* b*` indicates any strings that start with a or b character.
- a **lpad** b (binary operator) pads the value of a to the left with b padding pattern. For instance, 12 lpad "0000" generates the string "0012".
- a **rpad** b (binary operator) pads the value of a to the right with b padding pattern. For instance, 12 lpad "____" generates the string "12__".
- a **concat** b (binary operator) concatenates the a (as string) for b times. For instance, "x" concat 5, generates the string "xxxxx".

The operators for dates are:

- **time** (unary operator) retrieves the time of the date in string format, as specified in the control's panel. For instance, the *time(#1/1/2001 13:00#)* returns "1:00:00 PM"
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance, the *timeF(#1/1/2001 13:00#)* returns "13:00:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel. For instance, the *shortdate(#1/1/2001 13:00#)* returns "1/1/2001"
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance, the *shortdateF(#1/1/2001 13:00#)* returns "01/01/2001"
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format. For instance, the *dateF(#01/01/2001 14:00:00#)* returns #01/01/2001 14:00:00#
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel. For instance, the *longdate(#1/1/2001 13:00#)* returns "Monday, January 01, 2001"
- **year** (unary operator) retrieves the year of the date (100,...,9999). For instance, the *year(#12/31/1971 13:14:15#)* returns 1971
- **month** (unary operator) retrieves the month of the date (1, 2,...,12). For instance, the *month(#12/31/1971 13:14:15#)* returns 12.
- **day** (unary operator) retrieves the day of the date (1, 2,...,31). For instance, the *day(#12/31/1971 13:14:15#)* returns 31
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365). For instance, the *yearday(#12/31/1971 13:14:15#)* returns 365
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday). For instance, the *weekday(#12/31/1971 13:14:15#)* returns 5.
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23). For instance, the *hour(#12/31/1971 13:14:15#)* returns 13
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59). For instance, the *min(#12/31/1971 13:14:15#)* returns 14
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59). For instance, the *sec(#12/31/1971 13:14:15#)* returns 15
- value **format** 'flags' (binary operator) formats a date expression with specified flags. The format method formats numeric (depends on the type of the value, explained at operators for numbers) or date expressions. If not supported, the value is formatted as a number (the date format is supported by newer version only). The flags specifies the format picture string that is used to form the date. Possible values for the format picture string are defined below. For instance, the *date(value) format `MMM d, yyyy`*

returns "Sep 2, 2023"

The following table defines the format types used to represent days:

- d, day of the month as digits without leading zeros for single-digit days (8)
- dd, day of the month as digits with leading zeros for single-digit days (08)
- ddd, abbreviated day of the week as specified by the current locale ("Mon" in English)
- dddd, day of the week as specified by the current locale ("Monday" in English)

The following table defines the format types used to represent months:

- M, month as digits without leading zeros for single-digit months (4)
- MM, month as digits with leading zeros for single-digit months (04)
- MMM, abbreviated month as specified by the current locale ("Nov" in English)
- MMMM, month as specified by the current locale ("November" for English)

The following table defines the format types used to represent years:

- y, year represented only by the last digit (3)
- yy, year represented only by the last two digits. A leading zero is added for single-digit years (03)
- yyy, year represented by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed.
- yyyy, behaves identically to "yyyy"

The expression supports also **immediate if** (similar with iif in visual basic, or ? : in C++) ie cond ? value_true : value_false, which means that once that cond is true the value_true is used, else the value_false is used. Also, it supports variables, up to 10 from 0 to 9. For instance, 0:="Abc" means that in the variable 0 is "Abc", and =:0 means retrieves the value of the variable 0. For instance, the len(%0) ? (0:=(%1+%2) ? currency(=:0) else `` ) : `` gets the sum between second and third column in currency format if it is not zero, and only if the first column is not empty. As you can see you can use the variables to avoid computing several times the same thing (in this case the sum %1 and %2 .

OrderID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	Freight	ShipName
10248	5	8/4/1994	9/1/1994	8/16/1994	3	32.38	Vins et alcools Chevalie
10249	6	8/5/1994	9/16/1994	8/10/1994	1	11.61	Toms Spezialitäten
10250	4	8/8/1994	9/5/1994	8/12/1994	2	65.83	Hanari Carnes
10251	3	8/8/1994	9/5/1994	8/15/1994	1	41.34	Victuailles en stock
10252	4	8/9/1994	9/6/1994	8/11/1994	2	51.3	Suprêmes délices
10253	3	8/10/1994	8/24/1994	8/16/1994	2	58.17	Hanari Carnes
10254	5	8/11/1994	9/8/1994	8/23/1994	2	22.98	Chop-suey Chinese
10255	9	8/12/1994	9/9/1994	8/15/1994	3	148.33	Richter Supermarkt
10256	3	8/15/1994	9/12/1994	8/17/1994	2	13.97	Wellington Importadora
10257	4	8/16/1994	9/13/1994	8/22/1994	3	81.91	HILARIÓN-Abastos
10258	1	8/17/1994	9/14/1994	8/23/1994	1	140.51	Ernst Handel
10259	4	8/18/1994	9/15/1994	8/25/1994	3	3.25	Centro comercial Moct
10260	4	8/19/1994	9/16/1994	8/29/1994	1	55.09	Ottilies Käseladen
10261	4	8/19/1994	9/16/1994	8/30/1994	2	3.05	Que Delícia
10262	8	8/22/1994	9/19/1994	8/25/1994	3	48.29	Rattlesnake Canyon Gro
10263	9	8/23/1994	9/20/1994	8/31/1994	3	146.06	Ernst Handel
10264	6	8/24/1994	9/21/1994	9/23/1994	3	3.67	Folk och fä HB
10265	2	8/25/1994	9/22/1994	9/12/1994	1	55.28	Blondel pčre et fils
10266	3	8/26/1994	10/7/1994	8/31/1994	3	25.73	Wartian Herkku
10267	4	8/29/1994	9/26/1994	9/6/1994	1	208.58	Frankenversand
10268	8	8/30/1994	9/27/1994	9/2/1994	3	66.29	GROSELLA-Restaurant
10269	5	8/31/1994	9/14/1994	9/9/1994	1	4.56	White Clover Markets
10270	1	9/1/1994	9/29/1994	9/2/1994	1	136.54	Wartian Herkku
10271	6	9/1/1994	9/29/1994	9/30/1994	2	4.54	Split Rail Beer & Ale
10272	6	9/2/1994	9/30/1994	9/6/1994	2	98.03	Rattlesnake Canyon Gro
10273	3	9/5/1994	10/3/1994	9/12/1994	3	76.07	QUICK-Stop
10274	6	9/6/1994	10/4/1994	9/16/1994	1	6.01	Vins et alcools Chevalie
10275	1	9/7/1994	10/5/1994	9/9/1994	1	26.93	Magazzini Alimentari R
10276	8	9/8/1994	9/22/1994	9/14/1994	3	13.84	Tortuga Restaurante
10277	2	9/9/1994	10/7/1994	9/13/1994	3	125.77	Morgenstern Gesundkos
10278	8	9/12/1994	10/10/1994	9/16/1994	2	92.69	Berglunds snabbköp
10279	8	9/13/1994	10/11/1994	9/16/1994	2	25.83	Lehmanns Marktstand
10280	2	9/14/1994	10/12/1994	10/13/1994	1	8.98	Berglunds snabbköp
10281	4	9/14/1994	9/28/1994	9/21/1994	1	2.94	Romero y tomillo
10282	4	9/15/1994	10/13/1994	9/21/1994	1	12.69	Romero y tomillo
10283	3	9/16/1994	10/14/1994	9/23/1994	3	84.81	LILA-Supermercado
10284	4	9/19/1994	10/17/1994	9/27/1994	1	76.56	Lehmanns Marktstand
10285	1	9/20/1994	10/18/1994	9/26/1994	2	76.83	QUICK-Stop
10286	8	9/21/1994	10/19/1994	9/30/1994	3	229.24	QUICK-Stop

10287	8	9/22/1994	10/20/1994	9/28/1994	3	12.76	Ricardo Adocicados
10288	4	9/23/1994	10/21/1994	10/4/1994	1	7.45	Reggiani Caseifici
10289	7	9/26/1994	10/24/1994	9/28/1994	3	22.77	B's Beverages
10290	8	9/27/1994	10/25/1994	10/4/1994	1	79.7	Comércio Mineiro
10291	6	9/27/1994	10/25/1994	10/5/1994	2	6.4	Que Delícia
10292	1	9/28/1994	10/26/1994	10/3/1994	2	1.35	Tradição Hipermercado
10293	1	9/29/1994	10/27/1994	10/12/1994	3	21.18	Tortuga Restaurante
10294	4	9/30/1994	10/28/1994	10/6/1994	2	147.26	Rattlesnake Canyon Gro
10295	2	10/3/1994	10/31/1994	10/11/1994	2	1.15	Vins et alcools Chevalie
10296	6	10/4/1994	11/1/1994	10/12/1994	1	0.12	LILA-Supermercado
10297	5	10/5/1994	11/16/1994	10/11/1994	2	5.74	Blondel pčre et fils
10298	6	10/6/1994	11/3/1994	10/12/1994	2	168.22	Hungry Owl All-Night G
10299	4	10/7/1994	11/4/1994	10/14/1994	2	29.76	Ricardo Adocicados
10300	2	10/10/1994	11/7/1994	10/19/1994	2	17.68	Magazzini Alimentari R
10301	8	10/10/1994	11/7/1994	10/18/1994	2	45.08	Die Wandernde Kuh
10302	4	10/11/1994	11/8/1994	11/9/1994	2	6.27	Supręmes délices
10303	7	10/12/1994	11/9/1994	10/19/1994	2	107.83	Godos Cocina Típica
10304	1	10/13/1994	11/10/1994	10/18/1994	2	63.79	Tortuga Restaurante
10305	8	10/14/1994	11/11/1994	11/9/1994	3	257.62	Old World Delicatessen
10306	1	10/17/1994	11/14/1994	10/24/1994	3	7.56	Romero y tomillo
10307	2	10/18/1994	11/15/1994	10/26/1994	2	0.56	Lonesome Pine Restaura
10308	7	10/19/1994	11/16/1994	10/25/1994	3	1.61	Ana Trujillo Emparedad
10309	3	10/20/1994	11/17/1994	11/23/1994	1	47.3	Hungry Owl All-Night G
10310	8	10/21/1994	11/18/1994	10/28/1994	2	17.52	The Big Cheese
10311	1	10/21/1994	11/4/1994	10/27/1994	3	24.69	Du monde entier