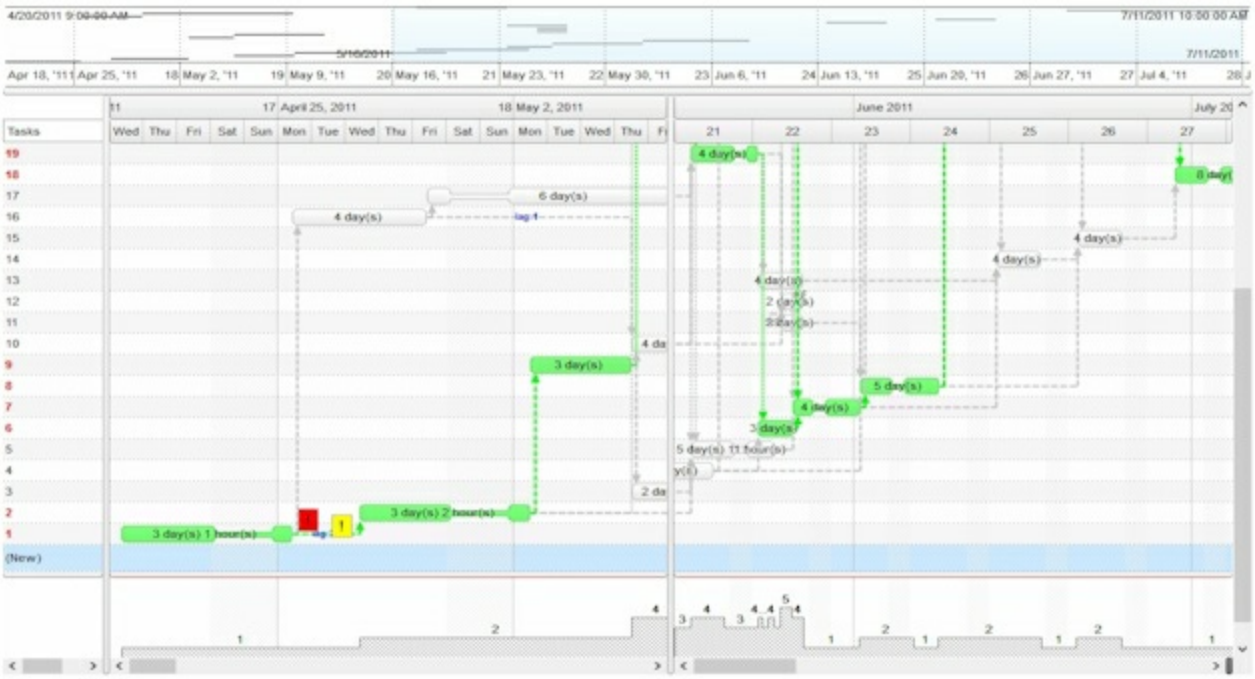
ExG2Host

The ExG2Host is an extension of the ExG2antt (Exontrol's Grid-Gantt component) with full database support (ADO, DAO, XML). In other words, the ExG2Host loads and saves automatically the host's data (including the hierarchy) to one or more databases. You can map a data field from the data-source, to a property of one object in the host/gantt control, and the control automatically updates the field when it is required. In the same manner, you can automatically save the control's layout, so next time the control is running it automatically get displayed the same way as it was closed. By default, the ExG2Host component let you add/remove/edit items, child-items, tasks, links and so on without having to code anything. The ExG2Host component provides all features of the ExG2antt component, it can be used as isolated, so no need for registration of any of the DLLs.

Features of eXG2Host include:

- Extends features of ExG2antt component
- ADO/DAO/XML full support
- AddNew, Delete and Update support
- Error support (ability to highlight the items with errors)
- Ability to map a field of the data-source with a property of an object



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

constants AppearanceEnum

The AppearanceEnum enumeration is used to specify the appearance of the control's border.

Name	Value	Description
None2	0	No border
Flat	1	Flat border
Sunken	2	Sunken border
Raised	3	Raised border
Etched	4	Etched border
Bump	5	Bump border

constants ContextHitTestInfoEnum

The ContextHitTestInfoEnum type defines areas of the host. The HitTest property determines the host' objects that can be accessed at current cursor location. For instance, if the HitTest property includes the exHTValidColumn flag, it indicates that the cursor hovers a column, and the Column property gives the index of the column from the cursor. The ContextHitTestInfoEnum type defines the following flags:

Name	Value	Description
exHTUnknown	0	The position of the cursor can not be determined as being one of the following.
exHTItemsArea	256	The cursor hovers the items/columns section of the control (left-panel).
exHTChartArea	512	The cursor hovers the chart/tasks section of the control (right-panel).
exHTEmptySpace	4096	The cursor hovers an empty part of the control (no item at the cursor position). This flag can be combined with exHTItemsArea or exHTChartArea, which indicates that the item hovers an empty area within the items/columns section of control, or hovers the empty area of the chart/tasks section of the control. If present, the exHTEmptySpace flag indicates that the EmptySpace property is valid (Returns the number of rows, between current cursor position and the the last fully-visible item of the control)
exHTValidMask	255	Specifies the mask for valid values.
exHTValidItem	1	If present, the exHTValidItem flag indicates that the Item property is valid (Returns the handle of the item from the current cursor location)
exHTValidColumn	2	If present, the exHTValidColumn flag indicates that the Column property is valid (Returns the index of the column from the current cursor location)
exHTValidDate	4	If present, the exHTValidDate flag indicates that the Date property is valid (Specifies the date-time from the current cursor location)
exHTValidBar	8	If present, the exHTValidBar flag indicates that the Bar property is valid (Returns the key of the bar from the current cursor location)

exHTValidLink	16	If present, the exHTValidLink flag indicates that the Link property is valid (Returns the key of the link from the current cursor location)

constants DefHostPropertyEnum

The DefHostPropertyEnum type defines properties whose default value can be changed. The HostDef property defines the default value for specified property. The DefHostPropertyEnum type supports the following values:

Name	Value	Description
exNew	1	By default, the exNew is "(New)". Retrieves or sets the caption to be displayed on the AddNew item. The HostDef(exNewColumn) property retrieves or sets the index of the column, where the (New) is being displayed. The (New) may not be displayed on formatted columns or on columns with drop down list editors. *(String expression)*
exNewColumn	2	By default, the exNewColumn is 0. Retrieves or sets the index of the column, where the (New) is being displayed. The HostDef(exNew) property retrieves or sets the caption to be displayed on the AddNew item. The (New) may not be displayed on formatted columns or on columns with drop down list editors. *(Long expression)*
exTaskName	3	By default, the exTaskName is "Task". Retrieves or sets the name of the task to be created / loaded. For instance, you can change the HostDef(exTaskName) property to "Progress" and so any new created bar will be of "Progress" type. *(String expression)*
		By default, the exNewTaskID is empty, if no data-source has been loaded. If the data-source has been specified it indicates the identifier of the next task that will be created. If data-source is assigned to the control, the HostDef(exNewTaskID) property defines the value of the next identifier that will be created once the user creates a new task. This property is automatically updated once the user

exNewTaskID	4	creates a new task, into a control with a data-source assigned. If no data-source is assigned, the user is responsible with changing this value. The HostDef(exNewTaskID) property defines the value of Key parameter of the AddBar method of the Items collection. *(Long expression or Variant expression)*
exErrorForeColor	5	By default, the exErrorForeColor property is RGB(255,0,0). Retrieves or sets the foreground-color to highlight items with errors. This property has effect only if the control is bounded to ADO / DAO source. It is important, that DataField(exItemsID) property to be set, so items with errors are being highlighted. If the HostDef(exErrorForeColor) property is -1, no foreground color is applied to any item with errors. *(Long/Color expression)*
exErrorBackColor	6	By default, the exErrorBackColor property is -1. Retrieves or sets the background-color to highlight items with errors. This property has effect only if the control is bounded to ADO / DAO source. It is important, that DataField(exItemsID) property to be set, so items with errors are being highlighted. If the HostDef(exErrorBackColor) property is -1, no background color is applied to any item with errors. *(Long/Color expression)*
exErrorClearOnChange	7	By default, the exErrorClearOnChange property is True, which indicates that no error will be highlighted once the user updates the control. Specifies whether the control keeps highlighting the errors once the changes occur. This property has effect only if the control is bounded to ADO / DAO source. It is important, that DataField(exItemsID) property to be set, so items with errors are being highlighted. *(Boolean expression)*

exNewLinkID	8	By default, the exNewLinkID is empty, if no data-source has been loaded. If the data-source has been specified it indicates the identifier of the next link that will be created. If data-source is assigned to the control, the HostDef(exNewLinkID) property defines the value of the next identifier that will be created once the user creates a new link. This property is automatically updated once the user creates a new link, into a control with a data-source assigned. If no data-source is assigned, the user is responsible with changing this value. The HostDef(exNewLinkID) property defines the value of LinkKey parameter of the AddLink method of the Items collection. *(Long expression or Variant expression)*
exNewNoteID	9	By default, the exNewNoteID is empty, if no data-source has been loaded. If the data-source has been specified it indicates the identifier of the next note that will be created. If data-source is assigned to the control, the HostDef(exNewNoteID) property defines the value of the next identifier that will be created once the user creates a new note. This property is NOT automatically updated once the user creates a new note, so the user is responsible with changing this value, once a new note has been added to the chart. The HostDef(exNewNoteID) property defines the value of ID parameter of the Add method of the Notes collection. *(Long expression or Variant expression)*

constants HostEventEnum

The HostEventEnum type holds identifiers of events being fired by the host. The HostEvent event notifies the application once the host fires an event. The HostEventParam(-2) gives a general information of the event. The HostEventEnum type supports the following values:

Name	Value	Description
exHostKeyDown	-602	Occurs when the user presses a key while an object has the focus. • The HostEventParam(-1) returns 2 • The HostEventParam(0) set/gets the value of the KeyCode parameter • The HostEventParam(1) gets the value of the Shift parameter This identifier handles the KeyDown event of the host.
exHostKeyUp	-604	Occurs when the user releases a key while an object has the focus. • The HostEventParam(-1) returns 2 • The HostEventParam(0) set/gets the value of the KeyCode parameter • The HostEventParam(1) gets the value of the Shift parameter This identifier handles the KeyUp event of the host.
exHostKeyPress	-603	Occurs when the user presses and releases an ANSI key. • The HostEventParam(-1) returns 1 • The HostEventParam(0) set/gets the value of the KeyAscii parameter This identifier handles the KeyPress event of the host.
exHostClick	-600	Occurs when the user presses and then releases the left mouse button over the control.

		• The HostEventParam(-1) returns 0 This identifier handles the Click event of the host.
exHostMouseMove	-606	Occurs when the user moves the mouse. • The HostEventParam(-1) returns 4 • The HostEventParam(0) gets the value of the Button parameter • The HostEventParam(1) gets the value of the Shift parameter • The HostEventParam(2) gets the value of the X parameter • The HostEventParam(3) gets the value of the Y parameter This identifier handles the MouseMove event of the host.
exHostMouseDown	-605	Occurs when the user presses a mouse button. • The HostEventParam(-1) returns 4 • The HostEventParam(0) gets the value of the Button parameter • The HostEventParam(1) gets the value of the Shift parameter • The HostEventParam(2) gets the value of the X parameter • The HostEventParam(3) gets the value of the Y parameter This identifier handles the MouseDown event of the host.
exHostMouseUp	-607	Occurs when the user releases a mouse button. • The HostEventParam(-1) returns 4 • The HostEventParam(0) gets the value of the Button parameter • The HostEventParam(1) gets the value of the Shift parameter • The HostEventParam(2) gets the value of the X parameter

		• The HostEventParam(3) gets the value of the Y parameter This identifier handles the MouseUp event of the host.
exHostDblClick	-601	Occurs when the user dblclk the left mouse button over an object. • The HostEventParam(-1) returns 3 • The HostEventParam(0) gets the value of the Shift parameter • The HostEventParam(1) gets the value of the X parameter • The HostEventParam(2) gets the value of the Y parameter This identifier handles the DblClick event of the host.
exHostAddItem	1	Occurs after a new Item has been inserted to Items collection • The HostEventParam(-1) returns 1 • The HostEventParam(0) gets the value of the Item parameter This identifier handles the AddItem event of the host.
exHostRemoveItem	2	Occurs before removing an Item. • The HostEventParam(-1) returns 1 • The HostEventParam(0) gets the value of the Item parameter This identifier handles the RemoveItem event of the host.
exHostColumnClick	3	Fired after the user clicks on column's header. • The HostEventParam(-1) returns 1 • The HostEventParam(0) gets the value of the Column parameter

		This identifier handles the ColumnClick event of the host.
exHostCellImageClick	4	Occurs when the user clicks the cell's icon. • The HostEventParam(-1) returns 2 • The HostEventParam(0) gets the value of the Item parameter • The HostEventParam(1) gets the value of the ColIndex parameter This identifier handles the CellImageClick event of the host.
exHostCellStateChanged	5	Fired after cell's state has been changed. • The HostEventParam(-1) returns 2 • The HostEventParam(0) gets the value of the Item parameter • The HostEventParam(1) gets the value of the ColIndex parameter This identifier handles the CellStateChanged event of the host.
exHostSelectionChanged	6	Fired after a new item has been selected. • The HostEventParam(-1) returns 0 This identifier handles the SelectionChanged event of the host.
exHostAddColumn	7	Fired after a new column has been added. • The HostEventParam(-1) returns 1 • The HostEventParam(0) gets the value of the Column parameter This identifier handles the AddColumn event of the host.
		Fired before deleting a column.

exHostRemoveColumn	8	• The HostEventParam(-1) returns 1 • The HostEventParam(0) gets the value of the Column parameter This identifier handles the RemoveColumn event of the host.
exHostOversizeChanged	9	Occurs when the right range of the scroll has been changed. • The HostEventParam(-1) returns 2 • The HostEventParam(0) gets the value of the Horizontal parameter • The HostEventParam(1) gets the value of the NewVal parameter This identifier handles the OversizeChanged event of the host.
exHostOffsetChanged	10	Occurs when the scroll position has been changed. • The HostEventParam(-1) returns 2 • The HostEventParam(0) gets the value of the Horizontal parameter • The HostEventParam(1) gets the value of the NewVal parameter This identifier handles the OffsetChanged event of the host.
exHostRClick	11	Fired when right mouse button is clicked. • The HostEventParam(-1) returns 0 This identifier handles the RClick event of the host. The control fires the Context event, once the user right clicks the host.
		Fired when a cell requires to format its caption. • The HostEventParam(-1) returns 3 • The HostEventParam(0) gets the value of the Item parameter

exHostFormatColumn	15	• The HostEventParam(1) gets the value of the ColIndex parameter • The HostEventParam(2) sets / gets the value of the Value parameter This identifier handles the FormatColumn event of the host.
exHostBeforeExpandItem	16	Fired before an item is about to be expanded (collapsed). • The HostEventParam(-1) returns 2 • The HostEventParam(0) gets the value of the Item parameter • The HostEventParam(1) sets / gets the value of the Cancel parameter This identifier handles the BeforeExpandItem event of the host.
exHostAfterExpandItem	17	Fired after an item is expanded (collapsed). • The HostEventParam(-1) returns 1 • The HostEventParam(0) gets the value of the Item parameter This identifier handles the AfterExpandItem event of the host.
exHostItemOleEvent	18	Fired when an ActiveX control hosted by an item has fired an event. • The HostEventParam(-1) returns 2 • The HostEventParam(0) gets the value of the Item parameter • The HostEventParam(1) gets the value of the Ev parameter This identifier handles the ItemOleEvent event of the host.
		Occurs when column's position or column's size is changed.

exHostLayoutChanged	19	• The HostEventParam(-1) returns 0 This identifier handles the LayoutChanged event of the host.
exHostHyperLinkClick	20	Occurs when the user clicks on a hyperlink cell. • The HostEventParam(-1) returns 2 • The HostEventParam(0) gets the value of the Item parameter • The HostEventParam(1) gets the value of the ColIndex parameter This identifier handles the HyperLinkClick event of the host.
exHostFilterChange	21	Occurs when filter was changed. • The HostEventParam(-1) returns 0 This identifier handles the FilterChange event of the host.
exHostTooltip	22	Fired when the control prepares the object's tooltip. • The HostEventParam(-1) returns 7 • The HostEventParam(0) gets the value of the Item parameter • The HostEventParam(1) gets the value of the ColIndex parameter • The HostEventParam(2) sets / gets the value of the Visible parameter • The HostEventParam(3) sets / gets the value of the X parameter • The HostEventParam(4) sets / gets the value of the Y parameter • The HostEventParam(5) gets the value of the CX parameter • The HostEventParam(6) gets the value of the CY parameter This identifier handles the ToolTip event of the host.

exHostFilterChanging	24	Notifies your application that the filter is about to change. • The HostEventParam(-1) returns 0 This identifier handles the FilterChanging event of the host.
exHostAddGroupItem	31	Occurs after a new Group Item has been inserted to Items collection. • The HostEventParam(-1) returns 1 • The HostEventParam(0) gets the value of the Item parameter This identifier handles the AddGroupItem event of the host.
exHostCellStateChanging	32	Fired before cell's state is about to be changed. • The HostEventParam(-1) returns 3 • The HostEventParam(0) gets the value of the Item parameter • The HostEventParam(1) gets the value of the ColIndex parameter • The HostEventParam(2) sets / gets the value of the NewState parameter This identifier handles the CellStateChanging event of the host.
exHostDateChange	50	Occurs when the first visible date is changed. • The HostEventParam(-1) returns 0 This identifier handles the DateChange event of the host.
exHostChartStartChanging	51	Occurs when the chart is about to be changed. • The HostEventParam(-1) returns 1 • The HostEventParam(0) gets the value of the Operation parameter

		This identifier handles the ChartStartChanging event of the host.
exHostChartEndChanging	52	Occurs when the chart is about to be changed. • The HostEventParam(-1) returns 1 • The HostEventParam(0) gets the value of the Operation parameter This identifier handles the ChartEndChanging event of the host.
exHostInsideZoom	53	Notifies your application that a date is about to be magnified. • The HostEventParam(-1) returns 1 • The HostEventParam(0) gets the value of the DateTime parameter This identifier handles the InsideZoom event of the host.
exHostBeforeDrawPart	54	Occurs just before drawing a part of the control. • The HostEventParam(-1) returns 7 • The HostEventParam(0) gets the value of the Part parameter • The HostEventParam(1) gets the value of the hDC parameter • The HostEventParam(2) gets the value of the X parameter • The HostEventParam(3) gets the value of the Y parameter • The HostEventParam(4) gets the value of the Width parameter • The HostEventParam(5) gets the value of the Height parameter • The HostEventParam(6) gets the value of the Cancel parameter This identifier handles the BeforeDrawPart event of the host.

exHostAfterDrawPart	55	Occurs right after drawing the part of the control. • The HostEventParam(-1) returns 6 • The HostEventParam(0) gets the value of the Part parameter • The HostEventParam(1) gets the value of the hDC parameter • The HostEventParam(2) gets the value of the X parameter • The HostEventParam(3) gets the value of the Y parameter • The HostEventParam(4) gets the value of the Width parameter • The HostEventParam(5) gets the value of the Height parameter This identifier handles the AfterDrawPart event of the host.
exHostSort	100	Occurs when the control sorts a column. • The HostEventParam(-1) returns 0 This identifier handles the Sort event of the host.
exHostEdit	101	Occurs just before editing the focused cell. • The HostEventParam(-1) returns 3 • The HostEventParam(0) gets the value of the Item parameter • The HostEventParam(1) gets the value of the ColIndex parameter • The HostEventParam(2) sets / gets the value of the Cancel parameter This identifier handles the Edit event of the host.
		Occurs when user clicks on the cell's button. • The HostEventParam(-1) returns 3 • The HostEventParam(0) gets the value of the Item parameter • The HostEventParam(1) gets the value of the

exHostButtonClick	102	ColIndex parameter • The HostEventParam(2) gets the value of the Key parameter This identifier handles the ButtonClick event of the host.
exHostChange	103	Occurs when the user changes the cell's content. • The HostEventParam(-1) returns 3 • The HostEventParam(0) gets the value of the Item parameter • The HostEventParam(1) gets the value of the ColIndex parameter • The HostEventParam(2) sets / gets the value of the NewValue parameter This identifier handles the Change event of the host.
exHostError	104	Fired when an internal error occurs. • The HostEventParam(-1) returns 2 • The HostEventParam(0) gets the value of the Error parameter • The HostEventParam(1) gets the value of the Description parameter This identifier handles the Error event of the host.
exHostValidateValue	105	Occurs before user changes the cell's value. • The HostEventParam(-1) returns 4 • The HostEventParam(0) gets the value of the Item parameter • The HostEventParam(1) gets the value of the ColIndex parameter • The HostEventParam(2) gets the value of the NewValue parameter • The HostEventParam(3) sets / gets the value of the Cancel parameter This identifier handles the ValidateValue event of the host.

exHostUserEditorOpen	106	Occurs when an user editor is about to be opened. • The HostEventParam(-1) returns 3 • The HostEventParam(0) gets the value of the Object parameter • The HostEventParam(1) gets the value of the Item parameter • The HostEventParam(2) gets the value of the ColIndex parameter This identifier handles the UserEditorOpen event of the host.
exHostUserEditorClose	107	Occurs when an user editor is about to be closed. • The HostEventParam(-1) returns 3 • The HostEventParam(0) gets the value of the Object parameter • The HostEventParam(1) gets the value of the Item parameter • The HostEventParam(2) gets the value of the ColIndex parameter This identifier handles the UserEditorClose event of the host.
exHostUserEditorOleEvent	108	Occurs when an user editor fires an event. • The HostEventParam(-1) returns 5 • The HostEventParam(0) gets the value of the Object parameter • The HostEventParam(1) gets the value of the Ev parameter • The HostEventParam(2) gets the value of the CloseEditor parameter • The HostEventParam(3) gets the value of the Item parameter • The HostEventParam(4) gets the value of the ColIndex parameter This identifier handles the UserEditorOleEvent event of the host.

exHostFocusChanged	109	Occurs when a new cell is focused. • The HostEventParam(-1) returns 0 This identifier handles the FocusChanged event of the host.
exHostEditOpen	110	Occurs when the edit operation starts. • The HostEventParam(-1) returns 0 This identifier handles the EditOpen event of the host.
exHostEditClose	111	Occurs when the edit operation ends. • The HostEventParam(-1) returns 0 This identifier handles the EditClose event of the host.
exHostBarResize	120	Occurs when the bar is moved or resized. • The HostEventParam(-1) returns 2 • The HostEventParam(0) gets the value of the Item parameter • The HostEventParam(1) gets the value of the Key parameter This identifier handles the BarResize event of the host.
exHostCreateBar	121	Fired when the user creates a new bar. • The HostEventParam(-1) returns 3 • The HostEventParam(0) gets the value of the Item parameter • The HostEventParam(1) gets the value of the DateStart parameter • The HostEventParam(2) gets the value of the DateEnd parameter This identifier handles the CreateBar event of the

		host.
exHostAddLink	122	Occurs when the user links two bars using the mouse. • The HostEventParam(-1) returns 1 • The HostEventParam(0) gets the value of the LinkKey parameter This identifier handles the AddLink event of the host.
exHostChartSelectionChanged	123	Occurs when the user selects objects in the chart area. • The HostEventParam(-1) returns 0 This identifier handles the ChartSelectionChanged event of the host.
exHostDateTimeChanged	124	Notifies your application that the current time is changed. • The HostEventParam(-1) returns 1 • The HostEventParam(0) gets the value of the DateTime parameter This identifier handles the DateTimeChanged event of the host.
exHostBarParentChange	125	Occurs just before moving a bar from current item to another item. • The HostEventParam(-1) returns 4 • The HostEventParam(0) gets the value of the Item parameter • The HostEventParam(1) gets the value of the Key parameter • The HostEventParam(2) gets the value of the NewItem parameter • The HostEventParam(3) sets / gets the value of the Cancel parameter This identifier handles the BarParentChange event

		of the host.
exHostOverviewZoom	126	Occurs once the user selects a new time scale unit in the overview zoom area. • The HostEventParam(-1) returns 0 This identifier handles the OverviewZoom event of the host.
exHostHistogramBoundsChanged	127	Occurs when the location and the size of the histogram is changed. • The HostEventParam(-1) returns 4 • The HostEventParam(0) gets the value of the X parameter • The HostEventParam(1) gets the value of the Y parameter • The HostEventParam(2) gets the value of the Width parameter • The HostEventParam(3) gets the value of the Height parameter This identifier handles the HistogramBoundsChanged event of the host.
exHostAllowLink	128	Notifies at runtime when a link between two bars is possible. • The HostEventParam(-1) returns 6 • The HostEventParam(0) gets the value of the StartItem parameter • The HostEventParam(1) gets the value of the StartBarKey parameter • The HostEventParam(2) gets the value of the EndItem parameter • The HostEventParam(3) gets the value of the EndBarKey parameter • The HostEventParam(4) gets the value of the LinkKey parameter • The HostEventParam(5) sets / gets the value of the Cancel parameter

		This identifier handles the AllowLink event of the host.
exHostBarResizing	129	Occurs when a bar is moving or resizing. • The HostEventParam(-1) returns 2 • The HostEventParam(0) gets the value of the Item parameter • The HostEventParam(1) gets the value of the Key parameter This identifier handles the BarResizing event of the host.
exHostAllowAutoDrag	130	Occurs when the user drags the item between InsertA and InsertB as child of NewParent • The HostEventParam(-1) returns 6 • The HostEventParam(0) gets the value of the Item parameter • The HostEventParam(1) gets the value of the NewParent parameter • The HostEventParam(2) gets the value of the InsertA parameter • The HostEventParam(3) gets the value of the InsertB parameter • The HostEventParam(4) sets / gets the value of the Cancel parameter This identifier handles the AllowAutoDrag event of the host.
exHostScrollButtonClick	200	Occurs when the user clicks a button in the scrollbar. • The HostEventParam(-1) returns 2 • The HostEventParam(0) gets the value of the ScrollBar parameter • The HostEventParam(1) gets the value of the ScrollPart parameter This identifier handles the ScrollButtonClick event of the host.

exHostAnchorClick	450	Occurs when an anchor element is clicked. • The HostEventParam(-1) returns 2 • The HostEventParam(0) gets the value of the AnchorID parameter • The HostEventParam(1) gets the value of the Options parameter This identifier handles the AnchorClick event of the host.

constants HostObjectFieldEnum

The HostObjectFieldEnum type specifies the properties you can map to a field into a data-source. The DataField property associates a data field to a property of the object of the host. The HostObjectFieldEnum type supports the following values:

Name	Value	Description
exItemsDataSource	0	Specifies the Name of the data source to load/save the rows of the host. The DataSource(Name)/DataMember(Name) defines the data source for the Items/Columns section of the host.
exItemsID	1	Specifies the name of the field from the items data source, that specifies the index/identifier of the row/item. This field should be numeric and unique. The CellValue property specifies the cell's value.
exItemsParentID	2	Specifies the name of the field from the items data source, that specifies the index/identifier of the parent's row/item. The ItemParent property specifies the handle of the item's parent.
exItemsPosition	3	Specifies the name of the field from the items data source, that specifies the position of the row/item. If set, this field is associated with the ItemPosition property of the Items object.
exItemsHeight	4	Specifies the name of the field from the items data source, that specifies the height of the row/item. If set, this field is associated with the ItemHeight property of the Items object.
exItemsBackColor	5	Specifies the name of the field from the items data source, that specifies the background color the row/item. If set, this field is associated with the ItemBackColor property of the Items object.
exItemsForeColor	6	Specifies the name of the field from the items data source, that specifies the foreground color the row/item. If set, this field is associated with the ItemForeColor property of the Items object.
		Specifies the Name of the data source to load/save the tasks of the host. The DataSource(Name)/DataMember(Name) defines the data source for the Tasks/Chart section of the

exTasksDataSource	7	host. If the DataField(exTasksDataSource) is DataField(exItemsDataSource) (items and tasks from the same table), only one task per item / row is allowed.
exTasksItemID	8	Specifies the name of the field from the tasks data source, that specifies the index/identifier of the row/item. This field is not required if the DataField(exTasksDataSource) is DataField(exItemsDataSource) (items and tasks from the same table).
exTasksStart	9	Specifies the name of the field from the tasks data source, that specifies the start date-time of the task. The ItemBar(,,exBarStart) property specifies the date-start of the task.
exTasksEnd	10	Specifies the name of the field from the tasks data source, that specifies the end date-time of the task. The ItemBar(,,exBarEnd) property specifies the date-end of the task.
exTasksName	11	Specifies the name of the field from the tasks data source, that specifies the name of the task. The ItemBar(,,exBarName) property specifies the name of the task.
exTasksColor	12	Specifies the name of the field from the tasks data source, that specifies the color of the task. The ItemBar(,,exBarColor) property specifies the color of the task.
exTasksCaption	13	Specifies the name of the field from the tasks data source, that specifies the caption of the task. The ItemBar(,,exBarCaption) property specifies the caption of the task.
exTasksID	14	Specifies the name of the field from the tasks data source, that specifies the unique key/identifier of bar inside the row/item. The ItemBar(,,exBarKey) property specifies the ID/Key of the task. This field should be numeric and unique.
exLinksDataSource	15	Specifies the Name of the data source to load/save the links of the host. The DataSource(Name)/DataMember(Name) defines the data source for the Links/Chart section of the

		host.
exLinksStart	16	Specifies the name of the field from the links data source, that specifies the start of the link. This could be the identifier of the item / key where the link starts.
exLinksEnd	17	Specifies the name of the field from the links data source, that specifies the end of the link. This could be the identifier of the item / key where the link ends.
exLinksColor	18	Specifies the name of the field from the links data source, that specifies the color of the link. The Link(,exLinkColor) property specifies the color of the link.
exLinksCaption	19	Specifies the name of the field from the links data source, that specifies the caption of the link. The Link(,exLinkText) property specifies the caption of the link.
exLinksID	20	Specifies the name of the field from the links data source, that specifies the unique key/identifier of the link. This field should be numeric and unique.
exNotesDataSource	21	Specifies the Name of the data source to load/save the notes of the host. The DataSource(Name)/DataMember(Name) defines the data source for the Notes collection of the host.
exNotesID	22	Specifies the name of the field from the notes data source, that specifies the identifier of the note within the host. This value is equivalent with the ID parameter of the Add method of the Notes collection. The ID property of the Note object specifies the unique identifier of the note to be shown on the chart. If the associated field is of Numeric type, the host automatically prefixes the number with a "N" character, so the notes will be serialized as "N1", "N2" and so on/. The HostDef (exNewNoteID) property defines the identifier of the new note to be created. This property is NOT automatically updated once the user creates a new note, so the user is responsible with changing this value, once a new note has been added to the chart.

exNotesItemID	23	Specifies the name of the field from the notes data source, that specifies the index/identifier of the row/item associated with the note. The Item property of the Note specifies the handle of the item associated with the note.
exNotesKey	24	Specifies the name of the field from the notes data source, that specifies the object (date, bar) related to the note. The Key property of the Note specifies the date or bar to be associated with the note. The Key parameter of the Add method of the Notes collection specifies the DATE or the BAR to be related with the newly added note.
exNotesStartCaption	25	Specifies the name of the field from the notes data source, that specifies the text/caption of the start part of the note. The PartText(exNoteStart) property specifies the HTML caption being shown in the start part of the note.
exNotesEndCaption	26	Specifies the name of the field from the notes data source, that specifies the text/caption of the end part of the note. The PartText(exNoteEnd) property specifies the HTML caption being shown in the end part of the note.
exNotesShowLink	27	Specifies the name of the field from the notes data source, that defines the link between parts of the note. The ShowLink property retrieves or sets a value that indicates the link between parts of the note
exNotesStartCanMove	28	Specifies the name of the field from the notes data source, that indicates whether the start part of the note can move. The PartCanMove(exNoteStart) property specifies whether the user can move the start part of the note.
exNotesEndCanMove	29	Specifies the name of the field from the notes data source, that indicates whether the end part of the note can move. The PartCanMove(exNoteEnd) property specifies whether the user can move the end part of the note.
exNotesRelativePosition	30	Specifies the name of the field from the notes data source, that defines the position of the note relative to associated object. The RelativePosition property

		specifies the position of the note relative to associated object
exNotesStartHOffset	31	Specifies the name of the field from the notes data source, that indicates the horizontal offset to display the start part of the note. The PartHOffset(exNoteStart) property specifies the horizontal offset to display the start part of the note.
exNotesEndHOffset	32	Specifies the name of the field from the notes data source, that indicates the horizontal offset to display the end part of the note. The PartHOffset(exNoteEnd) property specifies the horizontal offset to display the end part of the note.
exNotesStartVOffset	33	Specifies the name of the field from the notes data source, that indicates the vertical offset to display the start part of the note. The PartVOffset(exNoteStart) property specifies the vertical offset to display the start part of the note.
exNotesEndVOffset	34	Specifies the name of the field from the notes data source, that indicates the vertical offset to display the end part of the note. The PartVOffset(exNoteEnd) property specifies the vertical offset to display the end part of the note.
exNotesStartBackColor	35	Specifies the name of the field from the notes data source, that defines the background color of the start part of the note. The PartBackColor(exNoteStart) property specifies the background color to show the start part of the note.
exNotesEndBackColor	36	Specifies the name of the field from the notes data source, that defines the background color of the end part of the note. The PartBackColor(exNoteEnd) property specifies the background color to show the end part of the note.
exNotesStartForeColor	37	Specifies the name of the field from the notes data source, that defines the foreground color of the start part of the note. The PartForeColor(exNoteStart) property specifies the foreground color to show the start part of the note.
exNotesEndForeColor	38	Specifies the name of the field from the notes data source, that defines the foreground color of the end part of the note. The PartForeColor(exNoteEnd)

		property specifies the foreground color to show the end part of the note.
exNotesStartShadow	39	Specifies the name of the field from the notes data source, that indicates whether the start part of the note shows its shadow. The PartShadow(exNoteStart) property specifies whether the start part of the note shows a shadow border.
exNotesEndShadow	40	Specifies the name of the field from the notes data source, that indicates whether the end part of the note shows its shadow. The PartShadow(exNoteEnd) property specifies whether the end part of the note shows a shadow border.
exLinksStartPos	41	Specifies the name of the field from the links data source, that specifies where the link starts. The Link(exLinkStartPos) property specifies the position where the link starts in the source item.
exLinksEndPos	42	Specifies the name of the field from the links data source, that specifies where the link ends. The Link(exLinkEndPos) property specifies the position where the link ends in the target item.
exTasksPercent	43	Specifies the name of the field from the tasks data source, that specifies the percent of the task. The ItemBar(exBarPercent) property 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%).
exTasksShowPercentCaption	44	Specifies the name of the field from the tasks data source, that specifies whether the percent is shown on the task. The ItemBar(exBarShowPercentCaption) property specifies whether the percent is displayed as caption on the bar.
exTasksPercentCaptionFormat	45	Specifies the name of the field from the tasks data source, that specifies the format to displays the percent on the task. The ItemBar(exBarPercentCaptionFormat) property specifies the HTML format to be displayed as percent.

exTasksCanResize	46	Specifies the name of the field from the tasks data source, that specifies whether the task can be resized at runtime. The ItemBar(exBarCanResize) property specifies whether the bar can be resized.
exTasksCanMove	47	Specifies the name of the field from the tasks data source, that specifies whether the task can be moved at runtime. The ItemBar(exBarCanMove) property specifies whether the bar can be moved.
exTasksCanMoveToAnother	48	Specifies the name of the field from the tasks data source, that specifies whether the task can be moved to another item. The ItemBar(exBarCanMoveToAnother) property specifies whether the bar can be moved to another item.
exTasksPattern	49	Specifies the name of the field from the tasks data source, that specifies the pattern of the task. The ItemBar(exBarPattern) property specifies bar's individual pattern

constants HostReadOnlyEnum

The HostReadOnlyEnum type specifies whether AddNew, Delete or Update are supported by the Host. The HostReadOnly property retrieves or sets a value that indicates whether the host is readonly. The HostReadOnlyEnum type supports the following flags:

Name	Value	Description
exHostReadOnly	0	The host is read-only.
exHostAllowAddNewItem	1	The host allows adding new items (it displays the (New) row)
exHostAllowAddEmptyItem	2	The host allows adding new items if the user clicks the empty part of the control.
exHostAllowAddNewTask	4	The host allow creating new tasks.
exHostAllowAddNewLink	8	The host allows linking the tasks.
exHostAllowAddNew	13	The host allows adding new items, tasks or links.
exHostAllowDeleteItem	16	The host allows deleting the item. The user can delete the object at runtime, by pressing the Delete key.
exHostAllowDeleteTask	32	The host allows deleting the task. The user can delete the object at runtime, by pressing the Delete key.
exHostAllowDeleteLink	64	The host allows deleting the link. The user can delete the object at runtime, by pressing the Delete key.
exHostAllowDelete	112	The host allows deleting new items, tasks or links.
exHostAllowUpdate	128	The host allows updating.
exHostReadWrite	253	The host is read-write. The exHostReadWrite is a combination of exHostAllowAddNew, exHostAllowDelete and exHostAllowUpdate. You can use the exHostAllowAddEmptyItem to allow user create new items, once he clicks the empty part of the control.

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.

G2Host 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: {DFE195F7-4F43-482A-A14B-0C97B032A5E1}. The object's program identifier is: "Exontrol.G2Host". The /COM object module is: "ExG2Host.dll"

The ExG2Host is an extension of the ExG2antt (Exontrol's Grid-Gantt component) with full database support (ADO, DAO, XML). In other words, the ExG2Host loads and saves automatically the host's data (including the hierarchy) to one or more databases. You can map a data field from the data-source, to a property of one object in the host/gantt control, and the control automatically updates the field when it is required. In the same manner, you can automatically save the control's layout, so next time the control is running it automatically get displayed the same way as it was closed. By default, the ExG2Host component let you add/remove/edit items, child-items, tasks, links and so on without having to code anything. The ExG2Host component provides all features of the ExG2antt component, it can be used as isolated, so no need for registration of any of the DLLs. The G2Host object supports the following properties and methods:

Name	Description
AttachTemplate	Attaches a script to the current object, including the events, from a string, file, a safe array of bytes.
AutoSave	Specifies the time in ms to perform saving, once the user changes the host.
DataField	Associates a data field to a property of the object of the host.
DataMember	Specifies the host's data member.
DataSource	Specifies the host's data source.
DataTechnology	Specifies the host's data technology, like ADO or/and DAO.
ExecuteTemplate	Executes a template and returns the result.
Host	Specifies the reference to the host.
HostContext	Gets the host's context at the current cursor position.
HostDef	Defines the default value for specified property.
HostDirty	You can use the HostDirty property to determine whether the current host has been modified since it was last saved.
HostEventParam	Retrieves or sets a value that indicates the current's event parameter.
HostInitTemplate	Specifies the template to init the host.
	Retrieves or sets a value that indicates whether the host is

HostReadOnly	readonly.
hWnd	Retrieves the control's window handle.
Refresh	Refreshes the host.
Reload	Reloads the entire host.
Save	Saves the host, if it was modified.
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.
Version	Retrieves the control's version.

method G2Host.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 G2Host1_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 G2Host.AutoSave as Long

Specifies the time in ms to perform saving, once the user changes the host.

Type	Description
Long	A long expression that defines the time in ms to perform saving, once the user changes the host.

By default, the AutoSave property is 1000 ms, which indicates that the changes are automatically saved after 1 second. If AutoSave property is 0, the control's save the data on closing. The Save method saves the control's data, while the HostDirty property is True. If no change occurs since last save, nothing will be saved.

Once the HostDirty property is set, the Save will be performed if:

- calling programmatically the Save method
- after elapsed time, if the AutoSave property is positive
- once the user closes the form/dialog

Once the Save method is performed, the HostDirty property is set on False.

property G2Host.DataField(Field as HostObjectFieldEnum) as String

Associates a data field to a property of the object of the host.

Type	Description
Field as HostObjectFieldEnum	A HostObjectFieldEnum expression that defines the property to be associated
String	A String expression that defines a data-source name, or a field to be mapped.

When you are bound the control to a data-source you can automatically loads/saves the items, tasks and links of the host. The following properties are used together to provide data-source for the control:

- DataSource property, specifies the data-source for the host.
- DataMember property, specifies the query or the name of the table from the DataSource that's bounded with a specified part of the host. This property should be used if the DataSource property refers a file, else the property is ignored
- DataField property, associates a data field to a property of the object of the host.
- DataTechnology property specifies the host's data technology, like XML, ADO or/and DAO to be used.

Based on the DataTechnology and DataSource properties, the control can use any of the following technologies:

- **XML (Extensible Markup Language)**, if the DataTechnology property includes the "MSXML.DOMDocument" and the DataSource property points to a XML file, or the DataSource property refers to an object of MSXML.DOMDocument type.
- **ADO (ActiveX Data Objects)**, if the DataTechnology property includes the "ADODB.Recordset" or "ADOR.Recordset" and the DataSource property points to a ACCDB / MDB / DBF file, or the DataSource property refers to an object of ADODB.Recordset or ADOR.Recordset type.
- **DAO (Data Access Object)**, if the DataTechnology property includes the "DAO.DBEngine.120" or "DAO.DBEngine.36" and the DataSource property points to a ACCDB / MDB, or the DataSource property refers to an object of DAO.DBEngine.36 or DAO.DBEngine.120.

to bound the host to a data-source.

If using the **XML (Extensible Markup Language)** technology, only following properties are required:

- DataSource property, specifies the data-source for the host. It could be the XML file, or an object of MSXML.DOMDocument type

- DataField(exItemsDataSource) property, which specifies the name of the data-source to be bounded with the control.

The following sample bounds the control **XML (Extensible Markup Language)** way:

```
With G2Host1
  .DataSource("Items") = "C:\Program Files\Exontrol\ExG2Host\Sample\sample.xml"
  .DataField(exItemsDataSource) = "Items"
End With
```

When using the **XML (Extensible Markup Language)** technology the control uses the Host's LoadXML and SaveXML to perform savings.

When using the **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)** technology, the following properties are required (**items/columns** section):

- DataSource property, specifies the data-source for the host. It could be the ACCDB/MDB/DBF file, or an object of ADODB.Recordset or ADOR.Recordset type
- DataMember property, specifies the query or the table definition to be bounded with the items section of the host, if the DataSource property is of string type, else it is ignored.
- DataField(exItemsDataSource) property, which specifies the name of the data-source / data-member which bounds the items of the host to the data-source

The following sample bounds the items **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)** way:

```
With G2Host1
  .DataSource("Items") = "C:\Program Files\Exontrol\ExG2Host\Sample\sample.accdb"
  .DataMember("Items") = "Employees"
  .DataField(exItemsDataSource) = "Items"
End With
```

A) When using the **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)**, the tasks can be loaded/saved to the same data-source as items, and so only one task is possible per item, so in this case the following properties are required (**chart/tasks** section):

- DataField(exTasksDataSource) property must be the same as DataField(exItemsDataSource) property
- DataField(exTasksStart) property, specifies the name of the field from the tasks data source, that specifies the start date-time of the task.
- DataField(exTasksEnd) property, specifies the name of the field from the tasks data

source, that specifies the end date-time of the task.

The following sample bounds the items and tasks (same table) **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)** way:

```
With G2Host1
   .DataSource("Items") = "C:\Program Files\Exontrol\ExG2Host\Sample\sample.accdb"
   .DataMember("Items") = "Employees"
   .DataField(exItemsDataSource) = "Items"
   .DataField(exTasksDataSource) = .DataField(exItemsDataSource)
   .DataField(exTasksStart) = "BirthDate"
   .DataField(exTasksEnd) = "HireDate"
End With
```

B) When using the **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)**, the tasks can be loaded/saved from a different data-source, the following properties are required (**chart/tasks** section):

- DataSource property, specifies the data-source for the host. It could be the ACCDB/MDB/DBF file, or an object of ADODB.Recordset or ADOR.Recordset type
- DataMember property, specifies the query or the table definition to be bounded with the chart/tasks section of the host, if the DataSource property is of string type, else it is ignored.
- DataField(exTasksDataSource) property, which specifies the name of the data-source / data-member which bounds the chart/tasks of the host to the data-source
- DataField(exTasksItemID) property, specifies the name of the field from the tasks data source, that specifies the index/identifier of the row/item.
- DataField(exTasksStart) property, specifies the name of the field from the tasks data source, that specifies the start date-time of the task.
- DataField(exTasksEnd) property, specifies the name of the field from the tasks data source, that specifies the end date-time of the task.

in addition for items/columns section, the following is required too:

- DataField(exItemsID) property, specifies the name of the field from the items data source, that specifies the index/identifier of the row/item.

The following sample bounds the items and tasks (different tables) **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)** way:

```
With G2Host1
   .DataSource("Items") = "C:\Program Files\Exontrol\ExG2Host\Sample\sample.accdb"
```

```
   .DataMember("Items") = "Employees"
   .DataField(exItemsDataSource) = "Items"
   .DataField(exItemsID) = "EmployeeID"
   .DataSource("Tasks") = .DataSource("Items")
   .DataMember("Tasks") = "EmployeeDetails"
   .DataField(exTasksDataSource) = "Tasks"
   .DataField(exTasksItemID) = "EmployeeID"
   .DataField(exTasksStart) = "DateStart"
   .DataField(exTasksEnd) = "DateEnd"
End With
```

When using the **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)**, the links can be loaded/saved from a different data-source, so the following properties are required (**chart/links** section):

- DataSource property, specifies the data-source for the host. It could be the ACCDB/MDB/DBF file, or an object of ADODB.Recordset or ADOR.Recordset type
- DataMember property, specifies the query or the table definition to be bounded with the chart/links section of the host, if the DataSource property is of string type, else it is ignored.
- DataField(exLinksDataSource) property, which specifies the name of the data-source / data-member which bounds the links of the host to the data-source
- DataField(exLinksStart) property, specifies the name of the field from the links data source, that specifies the start of the link.
- DataField(exLinksEnd) property, specifies the name of the field from the links data source, that specifies the end of the link.

in addition for chart/tasks section, the following is required too:

- DataField(exTasksID) property, Specifies the name of the field from the tasks data source, that specifies the unique key/identifier of bar inside the row/item.

The following sample bounds the items, tasks and links (different tables) **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)** way:

```
With G2Host1
   .DataSource("Items") = "C:\Program Files\Exontrol\ExG2Host\Sample\sample.accdb"
   .DataMember("Items") = "Employees"
   .DataField(exItemsDataSource) = "Items"
   .DataField(exItemsID) = "EmployeeID"
   .DataSource("Tasks") = .DataSource("Items")
```

```
    .DataMember("Tasks") = "EmployeeDetails"
    .DataField(exTasksDataSource) = "Tasks"
    .DataField(exTasksItemID) = "EmployeeID"
    .DataField(exTasksStart) = "DateStart"
    .DataField(exTasksEnd) = "DateEnd"
    .DataField(exTasksID) = "TaskID"
    .DataSource("Links") = .DataSource("Items")
    .DataMember("Links") = "EmployeeLinks"
    .DataField(exLinksDataSource) = "Links"
    .DataField(exLinksStart) = "Start"
    .DataField(exLinksEnd) = "End"
End With
```

property G2Host.DataMember(Name as String) as String

Specifies the host's data member.

Type	Description
Name as String	A String expression that defines the name of the data-source.
String	A String expression that could be a query, a table def, while the data-source is passed as a string/file.

When you are bound the control to a data-source you can automatically loads/saves the items, tasks and links of the host. The following properties are used together to provide data-source for the control:

- DataSource property, specifies the data-source for the host.
- DataMember property, specifies the query or the name of the table from the DataSource that's bounded with a specified part of the host. This property should be used if the DataSource property refers a file, else the property is ignored
- DataField property, associates a data field to a property of the object of the host.
- DataTechnology property specifies the host's data technology, like XML, ADO or/and DAO to be used.

Based on the DataTechnology and DataSource properties, the control can use any of the following technologies:

- **XML (Extensible Markup Language)**, if the DataTechnology property includes the "MSXML.DOMDocument" and the DataSource property points to a XML file, or the DataSource property refers to an object of MSXML.DOMDocument type.
- **ADO (ActiveX Data Objects)**, if the DataTechnology property includes the "ADODB.Recordset" or "ADOR.Recordset" and the DataSource property points to a ACCDB / MDB / DBF file, or the DataSource property refers to an object of ADODB.Recordset or ADOR.Recordset type.
- **DAO (Data Access Object)**, if the DataTechnology property includes the "DAO.DBEngine.120" or "DAO.DBEngine.36" and the DataSource property points to a ACCDB / MDB, or the DataSource property refers to an object of DAO.DBEngine.36 or DAO.DBEngine.120.

to bound the host to a data-source.

If using the **XML (Extensible Markup Language)** technology, only following properties are required:

- DataSource property, specifies the data-source for the host. It could be the XML file, or an object of MSXML.DOMDocument type

- DataField(exItemsDataSource) property, which specifies the name of the data-source to be bounded with the control.

The following sample bounds the control **XML (Extensible Markup Language)** way:

```
With G2Host1
  .DataSource("Items") = "C:\Program Files\Exontrol\ExG2Host\Sample\sample.xml"
  .DataField(exItemsDataSource) = "Items"
End With
```

When using the **XML (Extensible Markup Language)** technology the control uses the Host's LoadXML and SaveXML to perform savings.

When using the **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)** technology, the following properties are required (**items/columns** section):

- DataSource property, specifies the data-source for the host. It could be the ACCDB/MDB/DBF file, or an object of ADODB.Recordset or ADOR.Recordset type
- DataMember property, specifies the query or the table definition to be bounded with the items section of the host, if the DataSource property is of string type, else it is ignored.
- DataField(exItemsDataSource) property, which specifies the name of the data-source / data-member which bounds the items of the host to the data-source

The following sample bounds the items **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)** way:

```
With G2Host1
  .DataSource("Items") = "C:\Program Files\Exontrol\ExG2Host\Sample\sample.accdb"
  .DataMember("Items") = "Employees"
  .DataField(exItemsDataSource) = "Items"
End With
```

A) When using the **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)**, the tasks can be loaded/saved to the same data-source as items, and so only one task is possible per item, so in this case the following properties are required (**chart/tasks** section):

- DataField(exTasksDataSource) property must be the same as DataField(exItemsDataSource) property
- DataField(exTasksStart) property, specifies the name of the field from the tasks data source, that specifies the start date-time of the task.
- DataField(exTasksEnd) property, specifies the name of the field from the tasks data

source, that specifies the end date-time of the task.

The following sample bounds the items and tasks (same table) **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)** way:

```
With G2Host1
   .DataSource("Items") = "C:\Program Files\Exontrol\ExG2Host\Sample\sample.accdb"
   .DataMember("Items") = "Employees"
   .DataField(exItemsDataSource) = "Items"
   .DataField(exTasksDataSource) = .DataField(exItemsDataSource)
   .DataField(exTasksStart) = "BirthDate"
   .DataField(exTasksEnd) = "HireDate"
End With
```

B) When using the **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)**, the tasks can be loaded/saved from a different data-source, the following properties are required (**chart/tasks** section):

- DataSource property, specifies the data-source for the host. It could be the ACCDB/MDB/DBF file, or an object of ADODB.Recordset or ADOR.Recordset type
- DataMember property, specifies the query or the table definition to be bounded with the chart/tasks section of the host, if the DataSource property is of string type, else it is ignored.
- DataField(exTasksDataSource) property, which specifies the name of the data-source / data-member which bounds the chart/tasks of the host to the data-source
- DataField(exTasksItemID) property, specifies the name of the field from the tasks data source, that specifies the index/identifier of the row/item.
- DataField(exTasksStart) property, specifies the name of the field from the tasks data source, that specifies the start date-time of the task.
- DataField(exTasksEnd) property, specifies the name of the field from the tasks data source, that specifies the end date-time of the task.

in addition for items/columns section, the following is required too:

- DataField(exItemsID) property, specifies the name of the field from the items data source, that specifies the index/identifier of the row/item.

The following sample bounds the items and tasks (different tables) **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)** way:

```
With G2Host1
   .DataSource("Items") = "C:\Program Files\Exontrol\ExG2Host\Sample\sample.accdb"
```

```
   .DataMember("Items") = "Employees"
   .DataField(exItemsDataSource) = "Items"
   .DataField(exItemsID) = "EmployeeID"
   .DataSource("Tasks") = .DataSource("Items")
   .DataMember("Tasks") = "EmployeeDetails"
   .DataField(exTasksDataSource) = "Tasks"
   .DataField(exTasksItemID) = "EmployeeID"
   .DataField(exTasksStart) = "DateStart"
   .DataField(exTasksEnd) = "DateEnd"
End With
```

When using the **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)**, the links can be loaded/saved from a different data-source, so the following properties are required (**chart/links** section):

- DataSource property, specifies the data-source for the host. It could be the ACCDB/MDB/DBF file, or an object of ADODB.Recordset or ADOR.Recordset type
- DataMember property, specifies the query or the table definition to be bounded with the chart/links section of the host, if the DataSource property is of string type, else it is ignored.
- DataField(exLinksDataSource) property, which specifies the name of the data-source / data-member which bounds the links of the host to the data-source
- DataField(exLinksStart) property, specifies the name of the field from the links data source, that specifies the start of the link.
- DataField(exLinksEnd) property, specifies the name of the field from the links data source, that specifies the end of the link.

in addition for chart/tasks section, the following is required too:

- DataField(exTasksID) property, Specifies the name of the field from the tasks data source, that specifies the unique key/identifier of bar inside the row/item.

The following sample bounds the items, tasks and links (different tables) **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)** way:

```
With G2Host1
   .DataSource("Items") = "C:\Program Files\Exontrol\ExG2Host\Sample\sample.accdb"
   .DataMember("Items") = "Employees"
   .DataField(exItemsDataSource) = "Items"
   .DataField(exItemsID) = "EmployeeID"
   .DataSource("Tasks") = .DataSource("Items")
```

```
    .DataMember("Tasks") = "EmployeeDetails"
    .DataField(exTasksDataSource) = "Tasks"
    .DataField(exTasksItemID) = "EmployeeID"
    .DataField(exTasksStart) = "DateStart"
    .DataField(exTasksEnd) = "DateEnd"
    .DataField(exTasksID) = "TaskID"
    .DataSource("Links") = .DataSource("Items")
    .DataMember("Links") = "EmployeeLinks"
    .DataField(exLinksDataSource) = "Links"
    .DataField(exLinksStart) = "Start"
    .DataField(exLinksEnd) = "End"
End With
```

property G2Host.DataSource(Name as String) as Variant

Specifies the host's data source.

Type	Description
Name as String	A String expression that defines the name of the data-source.
Variant	A String expression that could refer a file such as XML, ACCDB, MDB or DBF, or an object of XML, ADO or DAO type

When you are bound the control to a data-source you can automatically loads/saves the items, tasks and links of the host. The following properties are used together to provide data-source for the control:

- DataSource property, specifies the data-source for the host.
- DataMember property, specifies the query or the name of the table from the DataSource that's bounded with a specified part of the host. This property should be used if the DataSource property refers a file, else the property is ignored
- DataField property, associates a data field to a property of the object of the host.
- DataTechnology property specifies the host's data technology, like XML, ADO or/and DAO to be used.

Based on the DataTechnology and DataSource properties, the control can use any of the following technologies:

- **XML (Extensible Markup Language)**, if the DataTechnology property includes the "MSXML.DOMDocument" and the DataSource property points to a XML file, or the DataSource property refers to an object of MSXML.DOMDocument type.
- **ADO (ActiveX Data Objects)**, if the DataTechnology property includes the "ADODB.Recordset" or "ADOR.Recordset" and the DataSource property points to a ACCDB / MDB / DBF file, or the DataSource property refers to an object of ADODB.Recordset or ADOR.Recordset type.
- **DAO (Data Access Object)**, if the DataTechnology property includes the "DAO.DBEngine.120" or "DAO.DBEngine.36" and the DataSource property points to a ACCDB / MDB, or the DataSource property refers to an object of DAO.DBEngine.36 or DAO.DBEngine.120.

to bound the host to a data-source.

If using the **XML (Extensible Markup Language)** technology, only following properties are required:

- DataSource property, specifies the data-source for the host. It could be the XML file,

or an object of MSXML.DOMDocument type
- DataField(exItemsDataSource) property, which specifies the name of the data-source to be bounded with the control.

The following sample bounds the control **XML (Extensible Markup Language)** way:

```
With G2Host1
  .DataSource("Items") = "C:\Program Files\Exontrol\ExG2Host\Sample\sample.xml"
  .DataField(exItemsDataSource) = "Items"
End With
```

When using the **XML (Extensible Markup Language)** technology the control uses the Host's LoadXML and SaveXML to perform savings.

When using the **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)** technology, the following properties are required (**items/columns** section):

- DataSource property, specifies the data-source for the host. It could be the ACCDB/MDB/DBF file, or an object of ADODB.Recordset or ADOR.Recordset type
- DataMember property, specifies the query or the table definition to be bounded with the items section of the host, if the DataSource property is of string type, else it is ignored.
- DataField(exItemsDataSource) property, which specifies the name of the data-source / data-member which bounds the items of the host to the data-source

The following sample bounds the items **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)** way:

```
With G2Host1
  .DataSource("Items") = "C:\Program Files\Exontrol\ExG2Host\Sample\sample.accdb"
  .DataMember("Items") = "Employees"
  .DataField(exItemsDataSource) = "Items"
End With
```

A) When using the **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)**, the tasks can be loaded/saved to the same data-source as items, and so only one task is possible per item, so in this case the following properties are required (**chart/tasks** section):

- DataField(exTasksDataSource) property must be the same as DataField(exItemsDataSource) property
- DataField(exTasksStart) property, specifies the name of the field from the tasks data source, that specifies the start date-time of the task.

- DataField(exTasksEnd) property, specifies the name of the field from the tasks data source, that specifies the end date-time of the task.

The following sample bounds the items and tasks (same table) **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)** way:

```
With G2Host1
   .DataSource("Items") = "C:\Program Files\Exontrol\ExG2Host\Sample\sample.accdb"
   .DataMember("Items") = "Employees"
   .DataField(exItemsDataSource) = "Items"
   .DataField(exTasksDataSource) = .DataField(exItemsDataSource)
   .DataField(exTasksStart) = "BirthDate"
   .DataField(exTasksEnd) = "HireDate"
End With
```

B) When using the **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)**, the tasks can be loaded/saved from a different data-source, the following properties are required (**chart/tasks** section):

- DataSource property, specifies the data-source for the host. It could be the ACCDB/MDB/DBF file, or an object of ADODB.Recordset or ADOR.Recordset type
- DataMember property, specifies the query or the table definition to be bounded with the chart/tasks section of the host, if the DataSource property is of string type, else it is ignored.
- DataField(exTasksDataSource) property, which specifies the name of the data-source / data-member which bounds the chart/tasks of the host to the data-source
- DataField(exTasksItemID) property, specifies the name of the field from the tasks data source, that specifies the index/identifier of the row/item.
- DataField(exTasksStart) property, specifies the name of the field from the tasks data source, that specifies the start date-time of the task.
- DataField(exTasksEnd) property, specifies the name of the field from the tasks data source, that specifies the end date-time of the task.

in addition for items/columns section, the following is required too:

- DataField(exItemsID) property, specifies the name of the field from the items data source, that specifies the index/identifier of the row/item.

The following sample bounds the items and tasks (different tables) **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)** way:

```
With G2Host1
```

```
  .DataSource("Items") = "C:\Program Files\Exontrol\ExG2Host\Sample\sample.accdb"
  .DataMember("Items") = "Employees"
  .DataField(exItemsDataSource) = "Items"
  .DataField(exItemsID) = "EmployeeID"
  .DataSource("Tasks") = .DataSource("Items")
  .DataMember("Tasks") = "EmployeeDetails"
  .DataField(exTasksDataSource) = "Tasks"
  .DataField(exTasksItemID) = "EmployeeID"
  .DataField(exTasksStart) = "DateStart"
  .DataField(exTasksEnd) = "DateEnd"
End With
```

When using the **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)**, the links can be loaded/saved from a different data-source, so the following properties are required (**chart/links** section):

- DataSource property, specifies the data-source for the host. It could be the ACCDB/MDB/DBF file, or an object of ADODB.Recordset or ADOR.Recordset type
- DataMember property, specifies the query or the table definition to be bounded with the chart/links section of the host, if the DataSource property is of string type, else it is ignored.
- DataField(exLinksDataSource) property, which specifies the name of the data-source / data-member which bounds the links of the host to the data-source
- DataField(exLinksStart) property, specifies the name of the field from the links data source, that specifies the start of the link.
- DataField(exLinksEnd) property, specifies the name of the field from the links data source, that specifies the end of the link.

in addition for chart/tasks section, the following is required too:

- DataField(exTasksID) property, Specifies the name of the field from the tasks data source, that specifies the unique key/identifier of bar inside the row/item.

The following sample bounds the items, tasks and links (different tables) **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)** way:

```
With G2Host1
  .DataSource("Items") = "C:\Program Files\Exontrol\ExG2Host\Sample\sample.accdb"
  .DataMember("Items") = "Employees"
  .DataField(exItemsDataSource) = "Items"
  .DataField(exItemsID) = "EmployeeID"
```

```
    .DataSource("Tasks") = .DataSource("Items")
    .DataMember("Tasks") = "EmployeeDetails"
    .DataField(exTasksDataSource) = "Tasks"
    .DataField(exTasksItemID) = "EmployeeID"
    .DataField(exTasksStart) = "DateStart"
    .DataField(exTasksEnd) = "DateEnd"
    .DataField(exTasksID) = "TaskID"
    .DataSource("Links") = .DataSource("Items")
    .DataMember("Links") = "EmployeeLinks"
    .DataField(exLinksDataSource) = "Links"
    .DataField(exLinksStart) = "Start"
    .DataField(exLinksEnd) = "End"
End With
```

property G2Host.DataTechnology(Name as String) as String

Specifies the host's data technology, like ADO or/and DAO.

Type	Description
Name as String	A String expression that defines the name of the data-source.
String	A String expression that defines the technologies available for the specified data-source separated by the ";" character.

By default, the DataTechnology property is "MSXML.DOMDocument;ADODB.Recordset;ADOR.Recordset;DAO.DBEngine.120;DAO.DI
For instance, if you want to force using the DAO technology you can set the DataTechnology property on "DAO.DBEngine.120;DAO.DBEngine.36" and so on other technology will be tried. When you are bound the control to a data-source you can automatically loads/saves the items, tasks and links of the host. The following properties are used together to provide data-source for the control:

- DataSource property, specifies the data-source for the host.
- DataMember property, specifies the query or the name of the table from the DataSource that's bounded with a specified part of the host. This property should be used if the DataSource property refers a file, else the property is ignored
- DataField property, associates a data field to a property of the object of the host.
- DataTechnology property specifies the host's data technology, like XML, ADO or/and DAO to be used.

Based on the DataTechnology and DataSource properties, the control can use any of the following technologies:

- **XML (Extensible Markup Language)**, if the DataTechnology property includes the "MSXML.DOMDocument" and the DataSource property points to a XML file, or the DataSource property refers to an object of MSXML.DOMDocument type.
- **ADO (ActiveX Data Objects)**, if the DataTechnology property includes the "ADODB.Recordset" or "ADOR.Recordset" and the DataSource property points to a ACCDB / MDB / DBF file, or the DataSource property refers to an object of ADODB.Recordset or ADOR.Recordset type.
- **DAO (Data Access Object)**, if the DataTechnology property includes the "DAO.DBEngine.120" or "DAO.DBEngine.36" and the DataSource property points to a ACCDB / MDB, or the DataSource property refers to an object of DAO.DBEngine.36 or DAO.DBEngine.120.

to bound the host to a data-source.

If using the **XML (Extensible Markup Language)** technology, only following properties are required:

- DataSource property, specifies the data-source for the host. It could be the XML file, or an object of MSXML.DOMDocument type
- DataField(exItemsDataSource) property, which specifies the name of the data-source to be bounded with the control.

The following sample bounds the control **XML (Extensible Markup Language)** way:

```
With G2Host1
   .DataSource("Items") = "C:\Program Files\Exontrol\ExG2Host\Sample\sample.xml"
   .DataField(exItemsDataSource) = "Items"
End With
```

When using the **XML (Extensible Markup Language)** technology the control uses the Host's LoadXML and SaveXML to perform savings.

When using the **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)** technology, the following properties are required (**items/columns** section):

- DataSource property, specifies the data-source for the host. It could be the ACCDB/MDB/DBF file, or an object of ADODB.Recordset or ADOR.Recordset type
- DataMember property, specifies the query or the table definition to be bounded with the items section of the host, if the DataSource property is of string type, else it is ignored.
- DataField(exItemsDataSource) property, which specifies the name of the data-source / data-member which bounds the items of the host to the data-source

The following sample bounds the items **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)** way:

```
With G2Host1
   .DataSource("Items") = "C:\Program Files\Exontrol\ExG2Host\Sample\sample.accdb"
   .DataMember("Items") = "Employees"
   .DataField(exItemsDataSource) = "Items"
End With
```

A) When using the **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)**, the tasks can be loaded/saved to the same data-source as items, and so only one task is possible per item, so in this case the following properties are required (**chart/tasks** section):

- DataField(exTasksDataSource) property must be the same as DataField(exItemsDataSource) property
- DataField(exTasksStart) property, specifies the name of the field from the tasks data source, that specifies the start date-time of the task.
- DataField(exTasksEnd) property, specifies the name of the field from the tasks data source, that specifies the end date-time of the task.

The following sample bounds the items and tasks (same table) **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)** way:

```
With G2Host1
   .DataSource("Items") = "C:\Program Files\Exontrol\ExG2Host\Sample\sample.accdb"
   .DataMember("Items") = "Employees"
   .DataField(exItemsDataSource) = "Items"
   .DataField(exTasksDataSource) = .DataField(exItemsDataSource)
   .DataField(exTasksStart) = "BirthDate"
   .DataField(exTasksEnd) = "HireDate"
End With
```

B) When using the **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)**, the tasks can be loaded/saved from a different data-source, the following properties are required (**chart/tasks** section):

- DataSource property, specifies the data-source for the host. It could be the ACCDB/MDB/DBF file, or an object of ADODB.Recordset or ADOR.Recordset type
- DataMember property, specifies the query or the table definition to be bounded with the chart/tasks section of the host, if the DataSource property is of string type, else it is ignored.
- DataField(exTasksDataSource) property, which specifies the name of the data-source / data-member which bounds the chart/tasks of the host to the data-source
- DataField(exTasksItemID) property, specifies the name of the field from the tasks data source, that specifies the index/identifier of the row/item.
- DataField(exTasksStart) property, specifies the name of the field from the tasks data source, that specifies the start date-time of the task.
- DataField(exTasksEnd) property, specifies the name of the field from the tasks data source, that specifies the end date-time of the task.

in addition for items/columns section, the following is required too:

- DataField(exItemsID) property, specifies the name of the field from the items data source, that specifies the index/identifier of the row/item.

The following sample bounds the items and tasks (different tables) **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)** way:

```
With G2Host1
   .DataSource("Items") = "C:\Program Files\Exontrol\ExG2Host\Sample\sample.accdb"
   .DataMember("Items") = "Employees"
   .DataField(exItemsDataSource) = "Items"
   .DataField(exItemsID) = "EmployeeID"
   .DataSource("Tasks") = .DataSource("Items")
   .DataMember("Tasks") = "EmployeeDetails"
   .DataField(exTasksDataSource) = "Tasks"
   .DataField(exTasksItemID) = "EmployeeID"
   .DataField(exTasksStart) = "DateStart"
   .DataField(exTasksEnd) = "DateEnd"
End With
```

When using the **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)**, the links can be loaded/saved from a different data-source, so the following properties are required (**chart/links** section):

- DataSource property, specifies the data-source for the host. It could be the ACCDB/MDB/DBF file, or an object of ADODB.Recordset or ADOR.Recordset type
- DataMember property, specifies the query or the table definition to be bounded with the chart/links section of the host, if the DataSource property is of string type, else it is ignored.
- DataField(exLinksDataSource) property, which specifies the name of the data-source / data-member which bounds the links of the host to the data-source
- DataField(exLinksStart) property, specifies the name of the field from the links data source, that specifies the start of the link.
- DataField(exLinksEnd) property, specifies the name of the field from the links data source, that specifies the end of the link.

in addition for chart/tasks section, the following is required too:

- DataField(exTasksID) property, Specifies the name of the field from the tasks data source, that specifies the unique key/identifier of bar inside the row/item.

The following sample bounds the items, tasks and links (different tables) **ADO (ActiveX Data Objects)** or **DAO (Data Access Object)** way:

```
With G2Host1
```

```
    .DataSource("Items") = "C:\Program Files\Exontrol\ExG2Host\Sample\sample.accdb"
    .DataMember("Items") = "Employees"
    .DataField(exItemsDataSource) = "Items"
    .DataField(exItemsID) = "EmployeeID"
    .DataSource("Tasks") = .DataSource("Items")
    .DataMember("Tasks") = "EmployeeDetails"
    .DataField(exTasksDataSource) = "Tasks"
    .DataField(exTasksItemID) = "EmployeeID"
    .DataField(exTasksStart) = "DateStart"
    .DataField(exTasksEnd) = "DateEnd"
    .DataField(exTasksID) = "TaskID"
    .DataSource("Links") = .DataSource("Items")
    .DataMember("Links") = "EmployeeLinks"
    .DataField(exLinksDataSource) = "Links"
    .DataField(exLinksStart) = "Start"
    .DataField(exLinksEnd) = "End"
End With
```

method G2Host.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 G2Host1.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.*

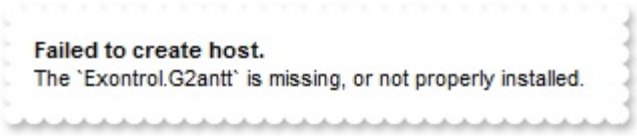
property G2Host.Host as Object

Specifies the reference to the host.

Type	Description
Object	An Object of ExG2antt type.

By default, the control creates an instance of ExG2antt type, and the Host property returns a reference to the newly created host. Once the Host is created, the HostInitTemplate property is applied, so columns, and colors are being initialized. You can use the Host property to access the inner host. The Host property returns Nothing, if the host has not been created. The HostReadOnly property specifies whether the host is read only. Once the Host property is invoked, the control's data is loaded based on the DataTechnology, DataSource, DataMember and DataField properties.

If the control is not able to create the Host, it may display:



Usually, this is happen, if no ExG2antt is installed on the machine.

If the control is not able to create the Host, it may display:

Usually, this is happen, if the version of the ExG2antt on the machine, is lower than 15.0. The eXG2Host tool requires version 15.0 of ExG2antt or upper.

If the host is created successfully, it may display:

In case you want to have some intelisense for VB6, all you need is to add reference to exg2antt control, Project\References\ExG2antt 1.0 Control Library and add a code such as:

```
Dim g As EXG2ANTTLib.G2antt
```

```
Set g = G2Host1.Host
With g
'     Now g is the inner control of EXG2Host, which is actually a ExG2antt control.
End With
```

For instance, the following sample adds Start and End columns:

```
With G2Host1
   .HostReadOnly = HostReadOnlyEnum.exHostReadWrite Or
HostReadOnlyEnum.exHostAllowAddEmptyItem
   Dim g As EXG2ANTTLib.G2antt
   Set g = G2Host1.Host
   With g
      .SingleSel = False
      .OnResizeControl = 1
      .ScrollBars = &H800 Or ScrollBarsEnum.exDisableNoVertical
      With .Columns.Add("Start")
         .AllowSizing = False
         .Def(18) = 1
         .Editor.EditType = 7
      End With
      With .Columns.Add("End")
         .AllowSizing = False
         .Def(18) = 2
         .Editor.EditType = 7
      End With
      .Items.AllowCellValueToItemBar = True
      With .Chart
         .AllowCreateBar = 1
         .PaneWidth(False) = 256
         .Bars.Item("Task").OverlaidType =
OverlaidBarsTypeEnum.exOverlaidBarsStackAutoArrange Or
OverlaidBarsTypeEnum.exOverlaidBarsStack
      End With
   End With
End With
```

property G2Host.HostContext as HostContext

Gets the host's context at the current cursor position.

Type	Description
HostContext	A HostContext object that holds information about item, task and link from the current cursor position

The HostContext property holds information about item, task and link from the current cursor position. The HitTest property determines the host' objects that can be accessed at current cursor location. For instance, if the HitTest property includes the exHTValidColumn flag, it indicates that the cursor hovers a column, and the Column property gives the index of the column from the cursor. The Context event notifies your application once the user right clicks the control. The HostContext property can be used on any event.

The following VB sample displays information about the current context, when user right clicks the control:

```
Private Sub G2Host1_Context()
   With G2Host1
      Debug.Print( .HostContext.ToString )
   End With
End Sub
```

property G2Host.HostDef(Property as DefHostPropertyEnum) as Variant

Defines the default value for specified property.

Type	Description
Property as DefHostPropertyEnum	A DefHostPropertyEnum property that specify the property to be accessed.
Variant	A Variant expression that defines the default value for giving property.

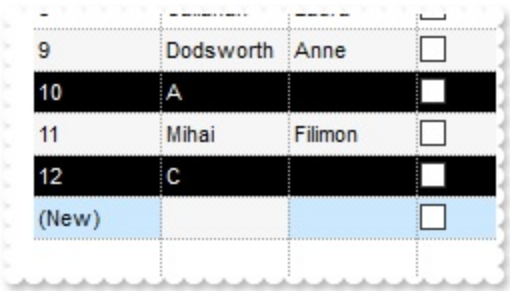
The HostDef property defines the default value for specified property. For instance, the HostDef(exNew) property defines the "(New)" caption to be displayed on the AddNew row.

The Refresh method refreshes the control including the inner host. The Refresh method may be required for changing some properties of the HostDef property.

The following VB sample changes the colors to highlight the errors:

```
With G2Host1
   .DataSource("Items") = "C:\Program Files\Exontrol\ExG2Host\Sample\sample.accdb"
   .DataMember("Items") = "Employees"
   .DataField(exItemsDataSource) = "Items"
   .DataField(exItemsID) = "EmployeeID"
   .HostDef(exErrorBackColor) = 0
   .HostDef(exErrorForeColor) = 16777215
End With
```

The following picture shows how errors shows up:



property G2Host.HostDirty as Boolean

You can use the HostDirty property to determine whether the current host has been modified since it was last saved.

Type	Description
Boolean	A Boolean expression that specifies whether the current host has been modified since it was last saved.

By default, the HostDirty property is False. Once a change occurs, the HostDirty property is set on True. Generally, any UI interaction of the control may produce a change, and if the change occurs the HostDirty property on True. For instance, the user resizes one or more task bars, a change occurs, and so the HostDirty property is set on True. The Save method saves the control's data, while the HostDirty property is True. If no change occurs since last save, nothing will be saved. The Save method may not be allowed based on the data-source provided. Before calling programmatically the Save method or performing changes in the control (like changing a property of a task bar) you should call the HostDirty property on True. The Reload method reloads the control's data for original sources. The Refresh method refreshes the control including the inner host.

Once the HostDirty property is set, the Save will be performed if:

- calling programmatically the Save method
- after elapsed time, if the AutoSave property is positive
- once the user closes the form/dialog

Once the Save method is performed, the HostDirty property is set on False.

The Error event notifies your application once any error occurs.

For instance, the following error:

0 The data-source provides no support for AddNew method[...]
0 The data-source provides no support for Delete method[...]
0 The data-source provides no support for Update method[...]

Indicates that no AddNew, Delete or Update is allowed, and so no Save will be performed.

property G2Host.HostEventParam(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 specifies the index of the parameter to be accessed. The HostEventParam(-1) property returns the count of parameters for current event.
Variant	A Variant value that indicates the value of the event's parameter.

The HostEventParam property retrieves or sets a value that indicates the current's event parameter. The HostEvent is fired each time the host fires a event. In other words, the HostEvent event forwards any event that the host may fire. Each event has different number of parameters, which is indicated by the HostEventParam(-1) property. Each parameter of the event can be accessed through the HostEventParam property. The HostEventParam(-2) gives a general information of the event.

The following VB sample displays information about the host's events:

```
Private Sub G2Host1_HostEvent(ByVal EventID As EXG2HOSTLibCtl.HostEventEnum)
   Debug.Print G2Host1.HostEventParam(-2)
End Sub
```

The following VB sample displays a message box, before pressing the Delete key, and cancel the operation if user selects No or Cancel:

```
Private Sub G2Host1_HostEvent(ByVal EventID As EXG2HOSTLibCtl.HostEventEnum)
   If (EventID = exHostKeyDown) Then
      If (vbKeyDelete = CInt(G2Host1.HostEventParam(0))) Then
         If Not (MsgBox("Do you want to delete?", vbYesNoCancel) = vbYes) Then
            G2Host1.HostEventParam(0) = 0
         End If
      End If
   End If
End Sub
```

You can disable completely deletion by removing the exHostAllowDelete flag from the HostReadOnly property.

property G2Host.HostInitTemplate as String

Specifies the template to init the host.

Type	Description
String	A String expression that defines the initial template to be applied when the control creates the host.

By default, the HostInitTemplate property defines the template to be applied on the Host, when it is created. In other words, it's the initialization code of the host, which include adding columns, colors, and so on. 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).

At creation time, the following code is applied to the host:

```
With .Host
   .BeginUpdate
   With .Columns
      .Clear
      .Add("Tasks").Editor.EditType = EditType
      With .Add("")
         .AllowSort = False
         .AllowSizing = False
         .FormatColumn = "1 pos ``"
         .Position = 0
         .Width = 32
         .Alignment = RightAlignment
         .Def(exCellPaddingLeft) = 2
         .Def(exCellPaddingRight) = 2
      End With
   End With
   .AntiAliasing = True
   .ColumnAutoResize = True
   .ScrollBySingleLine = True
   .HeaderHeight = 24
   .DefaultItemHeight = 20
   .ContinueColumnScroll = False
   .Appearance = None2
```

```
    .HeaderAppearance = Etched
    .SelForeColor = RGB(0,0,0)
    .SelBackColor = RGB(204,232,255)
    .BackColor = RGB(255,255,255)
    .ForeColor = RGB(0,0,0)
    .BackColorAlternate = RGB(246,246,246)
    .BackColorHeader = .BackColor
    .BackColorLevelHeader = .BackColor
    .Background(exCursorHoverColumn) = -1
    .DrawGridLines = exAllLines
    With .Chart
        .SelBackColor = G2antt1.SelBackColor
        .BackColor = RGB(255,255,255)
        .NonworkingDaysPattern = exPatternBDiagonal
        .LevelCount = 2
        .OverviewVisible = exOverviewShowAll
        .AllowResizeChart = ResizeChartEnum.exAllowChangeUnitScale Or
ResizeChartEnum.exAllowResizeChartMiddle Or
ResizeChartEnum.exAllowResizeChartHeader
        .DrawGridLines = exAllLines
        .FirstWeekDay = .LocFirstWeekDay
        .MonthNames = .LocMonthNames
        .WeekDays = .LocWeekDays
        .AMPM = .LocAMPM
        .UnitScale = exDay
    End With
    .EndUpdate
End With
```

property G2Host.HostReadOnly as HostReadOnlyEnum

Retrieves or sets a value that indicates whether the host is readonly.

Type	Description
HostReadOnlyEnum	A HostReadOnlyEnum expression that defines the operation being allowed

By default, the HostReadOnly property is exHostReadWrite, which indicates that all operations are allowed. Even so, if the control's data-source does not allow changes, certain operations may be not allowed.

The HostReadOnly property changes the following properties of the Host:

- ReadOnly property, Retrieves or sets a value that indicates whether the control is read only
- AutoDrag property, Gets or sets a value that indicates the way the component supports the AutoDrag feature
- Chart.AllowCreateBar property, Allows creating new bars using the mouse.
- Chart.BarsAllowSizing property, Specifies whether bars can be resized at run-time.
- Chart.AllowLinkBars property, Specifies whether the user can link the bars using the mouse.

The Error event notifies your application once any error occurs.

For instance, the following error:

0 The data-source provides no support for AddNew method[...]
0 The data-source provides no support for Delete method[...]
0 The data-source provides no support for Update method[...]

Indicates that no AddNew, Delete or Update is allowed, and so no Save will be performed.

For instance, the the HostReadOnly on exHostReadOnly, makes the control read-only, so no changes are allowed. Even so, you can still change the Host's properties like ReadOnly, AllowCreateBar, to make the control works the way as you desire.

property G2Host.hWnd as Long

Retrieves the control's window handle.

Type	Description
Long	A long expression that indicates the control's window handle.

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. The hWnd property of the Host object defines the hWnd of the inside host.

method G2Host.Refresh ()

Refreshes the host.

Type	Description

The Refresh method refreshes the control including the inner host. The Refresh method may be required for changing some properties of the HostDef property. The Reload method reloads the control's data for original sources.

method G2Host.Reload ()

Reloads the entire host.

Type	Description

method G2Host.Save ()

Saves the host, if it was modified.

Type	Description

The Save method saves the control's data, while the HostDirty property is True. If no change occurs since last save, nothing will be saved. The Save method may not be allowed based on the data-source provided. Before calling programmatically the Save method or performing changes in the control (like changing a property of a task bar) you should call the HostDirty property on True. Generally, any UI interaction of the control may produce a change, and if the change occurs the HostDirty property on True. For instance, the user resizes one or more task bars, a change occurs, and so the HostDirty property is set on True. The Reload method reloads the control's data for original sources. The Refresh method refreshes the control including the inner host.

Once the HostDirty property is set, the Save will be performed if:

- calling programmatically the Save method
- after elapsed time, if the AutoSave property is positive
- once the user closes the form/dialog

Once the Save method is performed, the HostDirty property is set on False.

The Error event notifies your application once any error occurs.

For instance, the following error:

0 The data-source provides no support for AddNew method[...]
0 The data-source provides no support for Delete method[...]
0 The data-source provides no support for Update method[...]

Indicates that no AddNew, Delete or Update is allowed, and so no Save will be performed.

property G2Host.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 G2Host.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 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, 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 G2Host.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 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 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, 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 G2Host.Version as String

Retrieves the control's version.

Type	Description
String	A String expression that determines the version of the running control, including the version of the host. For instance, "10.0.0.1.100000.DEBUG/15.0.0.1.150000.DEMO"

The Version property specifies the control's version. The Version property of the Host object defines the version of the inside host.

HostContext object

The HostContext object holds information about item, column, task, links, ... from the current cursor position. The HostContext property gets the host's context at the current cursor position. The HostContext property supports the following properties and methods:

Name	Description
Bar	Returns the key of the bar from the current cursor location.
Column	Returns the index of the column from the current cursor location.
Date	Specifies the date-time from the current cursor location.
EmptySpace	Returns the number of rows, between current cursor position and the the last fully-visible item of the control.
HitTest	Determines the host' objects that can be accessed at current cursor location
Item	Returns the handle of the item from the current cursor location.
Link	Returns the key of the link from the current cursor location.
ToString	Gets a string representation of the object.

property HostContext.Bar as Variant

Returns the key of the bar from the current cursor location.

Type	Description
Variant	A Variant expression that specifies the key/identifier of the task from the current cursor location.

The Bar property returns a valid value, while the HitTest property includes the exHTValidBar. As each task is assigned to an item, if the exHTValidItem is automatically set on the HitTest property, if the exHTValidBar is present. The ItemBar property of the Items object, specifies a property of the task, like colors, start and end margins, and so on. The Item property returns the handle of the item from the current cursor location. The exHTValidBar may occur while the cursor hovers the chart/tasks section of the control.

The following sample changes the color of the task's frame, when the user right clicks the control:

```
Private Sub G2Host1_Context()
   With G2Host1
      If (.HostContext.HitTest And exHTValidBar) Then
         .Host.Items.ItemBar(.HostContext.Item, .HostContext.Bar,
EXG2ANTTLibCtl.ItemBarPropertyEnum.exBarFrameColor) = RGB(255, 0, 0)
      End If
   End With
End Sub
```

property HostContext.Column as Long

Returns the index of the column from the current cursor location.

Type	Description
Long	A Long expression that specifies the index of the column from the current cursor location.

The Column property returns a valid value, while the HitTest property includes the exHTValidColumn. The Column property returns the index of the column from the current cursor location. For instance, the CellBold property of the Items object bolds or un-bolds the cell giving its handle and the index of the column. The Column value can be used on any property of the Items object that has a ColIndex parameter.

The following sample toggles the cell's bold attribute when the user right clicks the control:

```
Private Sub G2Host1_Context()
  With G2Host1
    If (.HostContext.HitTest And exHTValidItem) Then
      If (.HostContext.HitTest And exHTValidColumn) Then
        .Host.Items.CellBold(.HostContext.Item, .HostContext.Column) = Not
.Host.Items.CellBold(.HostContext.Item, .HostContext.Column)
      End If
    End If
  End With
End Sub
```

property HostContext.Date as Date

Specifies the date-time from the current cursor location.

Type	Description
Date	A Date expression that determines the date-time from the current cursor location.

The Date property returns a valid value, while the HitTest property includes the exHTValidDate. The exHTValidDate may occur while the cursor hovers the chart/tasks section of the control.

The following sample displays a message box with the date being clicked, once the user right-clicks the control:

```
Private Sub G2Host1_Context()
   With G2Host1
      If (.HostContext.HitTest And exHTValidDate) Then
         MsgBox .HostContext.Date
      End If
   End With
End Sub
```

The message box occurs only if clicking the chart section of the control.

property HostContext.EmptySpace as Long

Returns the number of rows, between current cursor position and the the last fully-visible item of the control.

Type	Description
Long	A Long expression that defines the number of rows, between current cursor position and the the last fully-visible item of the control.

The EmptySpace property returns a valid value, while the HitTest property includes the exHTEmptySpace. The EmptySpace property returns the number of rows, between current cursor position and the the last fully-visible item of the control..

property HostContext.HitTest as ContextHitTestInfoEnum

Determines the host' objects that can be accessed at current cursor location

Type	Description
ContextHitTestInfoEnum	A ContextHitTestInfoEnum expression that specifies the valid properties that can be accessed at current cursor location.

The HitTest property determines the host' objects that can be accessed at current cursor location. For instance, if the HitTest property includes the exHTValidColumn flag, it indicates that the cursor hovers a column, and the Column property gives the index of the column from the cursor. The Context event notifies your application once the user right clicks the control. The HostContext property can be used on any event.

The following sample changes the color of the task's frame, when the user right clicks the control:

```
Private Sub G2Host1_Context()
  With G2Host1
    If (.HostContext.HitTest And exHTValidBar) Then
      .Host.Items.ItemBar(.HostContext.Item, .HostContext.Bar,
EXG2ANTTLibCtl.ItemBarPropertyEnum.exBarFrameColor) = RGB(255, 0, 0)
    End If
  End With
End Sub
```

property HostContext.Item as Long

Returns the handle of the item from the current cursor location.

Type	Description
Long	A Long expression that specifies the handle of the item from the current cursor location.

The Item property returns a valid value, while the HitTest property includes the exHTValidItem. The Item property returns the handle of the item from the current cursor location. For instance, the ItemBold property of the Items object bolds or un-bolds the item giving its handle. The Item value can be used on any property of the Items object that has a Item parameter.

The following sample toggles the item's bold attribute when the user right clicks the control:

```
Private Sub G2Host1_Context()
   With G2Host1
      If (.HostContext.HitTest And exHTValidItem) Then
         .Host.Items.ItemBold(.HostContext.Item) = Not
.Host.Items.ItemBold(.HostContext.Item)
      End If
   End With
End Sub
```

property HostContext.Link as Variant

Returns the key of the link from the current cursor location.

Type	Description
Variant	A Variant expression that specifies the key / identifier of the link from current cursor location.

The Link property returns a valid value, while the HitTest property includes the exHTValidLink. The exHTValidLink may occur while the cursor hovers the chart/tasks section of the control. The Link property of the Items object defines properties of the link based on its identifier / key.

The following sample displays a message box with the key of the link being clicked, once the user right-clicks the control:

```
Private Sub G2Host1_Context()
  With G2Host1
    If (.HostContext.HitTest And exHTValidLink) Then
      MsgBox .HostContext.Link
    End If
  End With
End Sub
```

property HostContext.ToString as String

Gets a string representation of the object.

Type	Description
String	Gets a string representation of the object such as: exHTItemsArea,exHTEmptySpace,exHTValidColumn

The ToString property Gets a string representation of the object. The HitTest property determines the host' objects that can be accessed at current cursor location. For instance, if the HitTest property includes the exHTValidColumn flag, it indicates that the cursor hovers a column, and the Column property gives the index of the column from the cursor. The Context event notifies your application once the user right clicks the control.

The following sample displays brief information about the object from the current cursor position:

```
' HostEvent event - Notifies the application once the host fires an event.
Private Sub G2Host1_HostEvent(ByVal EventID As EXG2HOSTLibCtl.HostEventEnum)
    With G2Host1
        Debug.Print( .HostContext.ToString )
    End With
End Sub

With G2Host1
    .DataSource("Items") = "C:\Program Files\Exontrol\ExG2Host\Sample\sample.accdb"
    .DataMember("Items") = "Employees"
    .DataField(exItemsDataSource) = "Items"
    .HostReadOnly = HostReadOnlyEnum.exHostAllowUpdate Or
HostReadOnlyEnum.exHostAllowAddNew
End With
```

ExG2Host events

The ExG2Host component supports the following events:

Name	Description
Context	Indicates the the user right-clicks the host on an object.
Error	Fired when an internal error occurs.
HostEvent	Notifies the application once the host fires an event.

event Context ()

Indicates the the user right-clicks the host on an object.

Type	Description

The Context event notifies your application once the user right clicks the control. The HostContext property returns information about the object from the current cursor position. The Context event occurs right after HostEvent(exHostRClick) event.

The following VB sample displays information about the current context, when user right clicks the control:

```
Private Sub G2Host1_Context()
   With G2Host1
      Debug.Print( .HostContext.ToString )
   End With
End Sub
```

Syntax for Context event, **/NET** version, on:

C#

```
private void Context(object sender)
{
}
```

VB

```
Private Sub Context(ByVal sender As System.Object) Handles Context
End Sub
```

Syntax for Context event, **/COM** version, on:

C#

```
private void Context(object sender, EventArgs e)
{
}
```

C++

```
void OnContext()
{
}
```

C++ Builder

```
void __fastcall Context(TObject *Sender)
{
}
```

Delphi
```
procedure Context(ASender: TObject; );
begin
end;
```

Delphi 8 (.NET only)
```
procedure Context(sender: System.Object; e: System.EventArgs);
begin
end;
```

Powe…
```
begin event Context()

end event Context
```

VB.NET
```
Private Sub Context(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Context
End Sub
```

VB6
```
Private Sub Context()
End Sub
```

VBA
```
Private Sub Context()
End Sub
```

VFP
```
LPARAMETERS nop
```

Xbas…
```
PROCEDURE OnContext(oG2Host)

RETURN
```

Syntax for Context event, ***/COM*** version (others), on:

Java…
```
<SCRIPT EVENT="Context()" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function Context()
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComContext
    Forward Send OnComContext
End_Procedure
```

Visual Objects

```
METHOD OCX_Context() CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_Context()
{
}
```

XBasic

```
function Context as v ()
end function
```

dBASE

```
function nativeObject_Context()
return
```

event Error (Error as Long, Description as String)

Fired when an internal error occurs.

Type	Description
Error as Long	A Long expression that specifies the code of the error. If 0, the error indicates a warning or an information.
Description as String	A String expression that defines the error.

The Error event notifies your application once an error occurs. For instance, if the user lets a required field un-filed, the control fires the Event "You must enter a value in the '...' field. [...]". For instance, If the DataField(exItemsID) property is set, the control highlights in red, the items with errors. The HostDef(exErrorBackColor) and HostDef(exErrorForeColor) specify the colors to highlight the item with errors. If an error occurs during saving, the change won't be updated to the data-source, until the error is corrected. The HostDef(exErrorClearOnChange) specifies whether the items with errors are highlighted or un-highlighted once a change occurs.

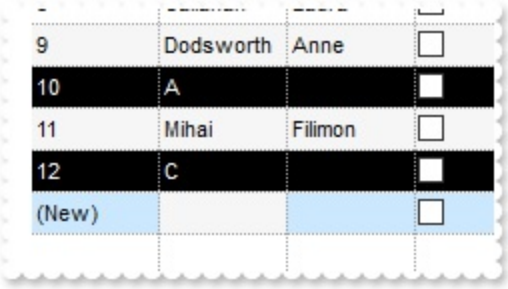
The following indicates warnings/information:

0 Use(ADODB.Recordset)[...]
0 The DateField(exTasksDataSource) is empty or refers to a missing data source

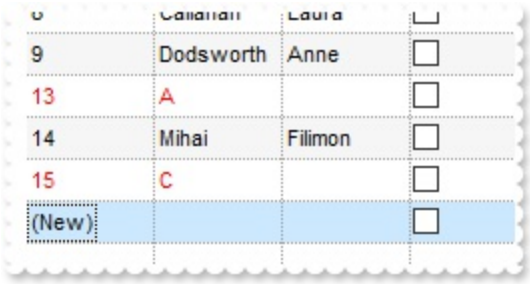
The following indicates errors:

-2147217887 You must enter a value in the '...' field.[...]
-2147217887 Field '...' cannot be a zero-length string.[...]

The following screen shot shows items with errors (white on black) :



The following screen shot shows items with errors (red) :



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, AxEXG2HOSTLib._IG2HostEvents_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:
AxEXG2HOSTLib._IG2HostEvents_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
AxEXG2HOSTLib._IG2HostEvents_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(oG2Host,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 HostEvent (EventID as HostEventEnum)

Notifies the application once the host fires an event.

Type	Description
EventID as HostEventEnum	A Long expression that defines the identifier of the event that occurs. The EventID parameter can be: • positive, to indicate that the control did not handle yet the event (before) • negative, to indicate that the control handled the event (after)

The HostEvent(**EventID**) event occurs once the host-control (exg2antt) fires an event and before the control itself to handle the event. The HostEvent(**-EventID**) event (negative event) occurs once the host-control (exg2antt) fires an event and after the control itself handles the event (starting from version 19.0). Each event has different number of parameters, which is indicated by the HostEventParam(-1) property. Each parameter of the event can be accessed through the HostEventParam property. The HostEventParam(-2) gives a general information of the event.

The HostEvent event occurs once the host-control (exg2antt) fires an event. The HostEvent(**EventID**) event (positive-event) is fired before the control itself to handle the event, while HostEvent(**-EventID**) event (negative-event) is fired after the control itself handles the event. For instance, HostEvent(**exHostCreateBar**) event notifies your application that a new bar is about to be created by drag and drop (the bar is not yet created), while HostEvent(**-exHostCreateBar**) event occurs once the control created the newly bar. During the HostEvent(-exHostCreateBar) event you can access the newly created bar using the HostEventParam(0) that returns the handle of the item that hosts the item-bar, and HostDef(exNewTaskID) gets the key of the newly created bar.

The following VB sample displays information about the host's events:

```
Private Sub G2Host1_HostEvent(ByVal EventID As EXG2HOSTLibCtl.HostEventEnum)
    Debug.Print G2Host1.HostEventParam(-2)
End Sub
```

The following VB sample changes the color for the newly created bar:

```
Private Sub G2Host1_HostEvent(ByVal EventID As EXG2HOSTLibCtl.HostEventEnum)
```

```
   If (EventID = -exHostCreateBar) Then
      G2Host1.Host.Items.ItemBar(G2Host1.HostEventParam(0),
G2Host1.HostDef(exNewTaskID), 33) = RGB(255, 0, 0)
   End If
End Sub
```

The following VB sample displays a message box, before pressing the Delete key, and cancel the operation if user selects No or Cancel:

```
Private Sub G2Host1_HostEvent(ByVal EventID As EXG2HOSTLibCtl.HostEventEnum)
   If (EventID = exHostKeyDown) Then
      If (vbKeyDelete = CInt(G2Host1.HostEventParam(0))) Then
         If Not (MsgBox("Do you want to delete?", vbYesNoCancel) = vbYes) Then
            G2Host1.HostEventParam(0) = 0
         End If
      End If
   End If
End Sub
```

You can disable completely deletion by removing the exHostAllowDelete flag from the HostReadOnly property.

Syntax for HostEvent event, **/NET** version, on:

C#
```
private void HostEvent(object sender,exontrol.EXG2HOSTLib.HostEventEnum
EventID)
{
}
```

VB
```
Private Sub HostEvent(ByVal sender As System.Object,ByVal EventID As
exontrol.EXG2HOSTLib.HostEventEnum) Handles HostEvent
End Sub
```

Syntax for HostEvent event, **/COM** version, on:

C#
```
private void HostEvent(object sender,
AxEXG2HOSTLib._IG2HostEvents_HostEventEvent e)
{
}
```

C++

```
void OnHostEvent(long   EventID)
{
}
```

C++ Builder

```
void __fastcall HostEvent(TObject *Sender,Exg2hostlib_tlb::HostEventEnum
EventID)
{
}
```

Delphi

```
procedure HostEvent(ASender: TObject; EventID : HostEventEnum);
begin
end;
```

Delphi 8 (.NET only)

```
procedure HostEvent(sender: System.Object; e:
AxEXG2HOSTLib._IG2HostEvents_HostEventEvent);
begin
end;
```

Powe…

```
begin event HostEvent(long  EventID)

end event HostEvent
```

VB.NET

```
Private Sub HostEvent(ByVal sender As System.Object, ByVal e As
AxEXG2HOSTLib._IG2HostEvents_HostEventEvent) Handles HostEvent
End Sub
```

VB6

```
Private Sub HostEvent(ByVal EventID As EXG2HOSTLibCtl.HostEventEnum)
End Sub
```

VBA

```
Private Sub HostEvent(ByVal EventID As Long)
End Sub
```

VFP

```
LPARAMETERS EventID

```

Xbas…

```
PROCEDURE OnHostEvent(oG2Host,EventID)

RETURN
```

Syntax for HostEvent event, **/COM** version (others), on:

Java…

```
<SCRIPT EVENT="HostEvent(EventID)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function HostEvent(EventID)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComHostEvent OLEHostEventEnum   llEventID
    Forward Send OnComHostEvent llEventID
End_Procedure
```

Visual Objects

```
METHOD OCX_HostEvent(EventID) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_HostEvent(int   _EventID)
{
}
```

XBasic

```
function HostEvent as v (EventID  as  OLE::Exontrol.G2Host.1::HostEventEnum)
end function
```

dBASE

```
function nativeObject_HostEvent(EventID)
return
```