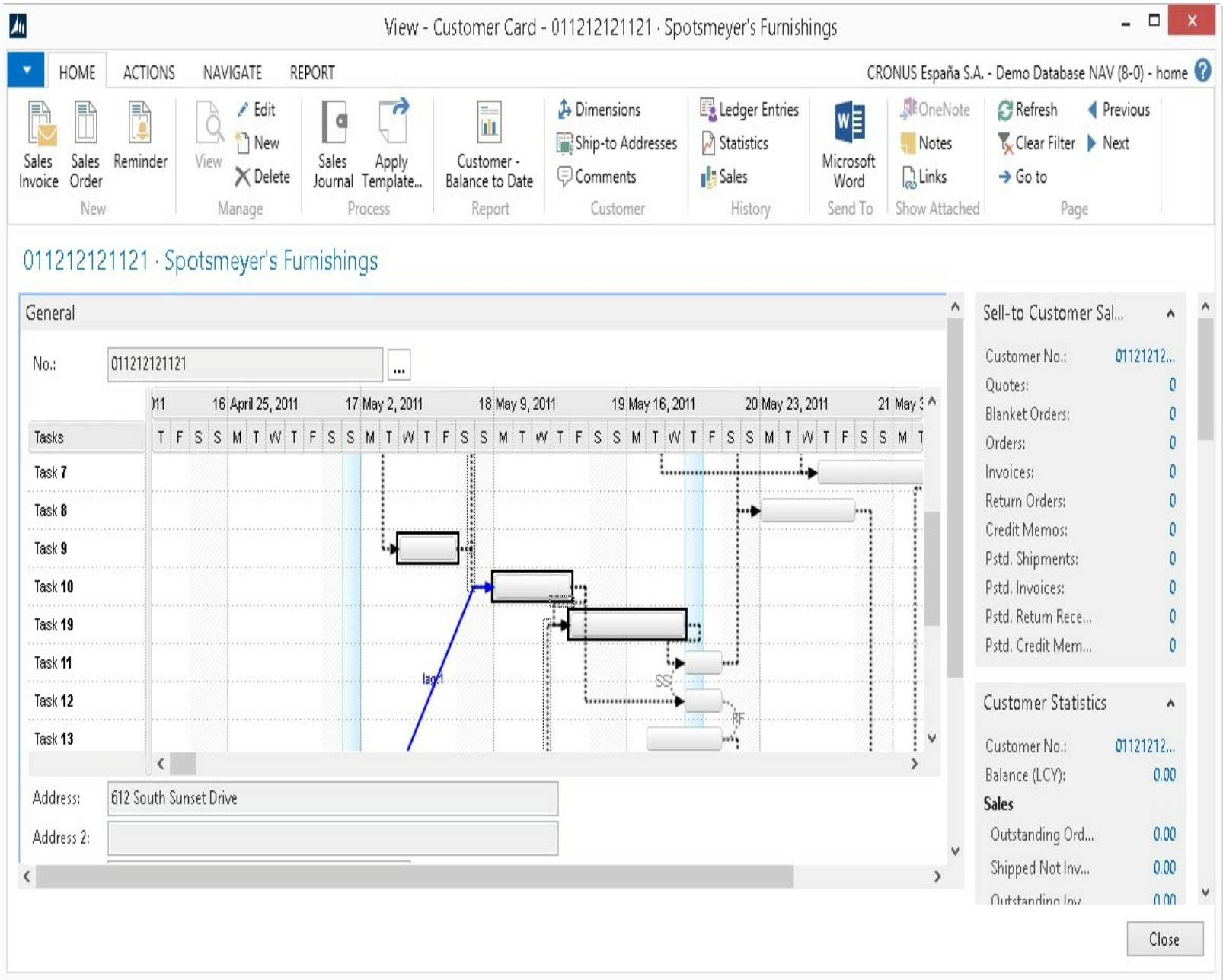
NAVHost Control-AddIn

The Exontrol's NAVHost/NET assembly is a Control-AddIn for Microsoft Dynamics NAV that allows you to use any UI element of the /NET Framework on any page. For instance, 'Height=32; AssemblyQualifiedName = "System.Windows.Forms.TrackBar, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"; Host.Template = "Value = 5"' adds a track-bar to your NAV form.

Features include:

- Control-AddIn for Microsoft Dynamics NAV
- X-Script / Template support, or ability to call properties or methods of the hosting control using strings
- Events support for hosting control through the OnControlAddIn trigger



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 NAVHostType

The NAVHostType type specifies the type of the control that the control hosts. The IsCreated / Create method returns a NAVHostType expression that can be one of the following:

Name	Value	Description
NAVHostType_exNAVHostNothing	0	The component hosts nothing.
NAVHostType_exNAVHostControl	1	The component hosts an object of System.Windows.Forms.Control type.
NAVHostType_exNAVHostObject	2	The component hosts a general object.

constants NAVHostVarEnum

The NAVHostVarEnum type specifies the VARIANT types. The VtType property specifies the VARIANT type of the object being hold by a NAVHostObject or NAVObjectTemplate object. The NAVHostVarEnum type supports the following values:

Name	Value	Description
NAVHostVarEnum_VT_EMPTY	0	Indicates that a value was not specified.
NAVHostVarEnum_VT_NULL	1	Indicates a null value, similar to a null value in SQL.
NAVHostVarEnum_VT_I2	2	Indicates a short integer.
NAVHostVarEnum_VT_I4	3	Indicates a long integer.
NAVHostVarEnum_VT_R4	4	Indicates a float value.
NAVHostVarEnum_VT_R8	5	Indicates a double value.
NAVHostVarEnum_VT_CY	6	Indicates a currency value.
NAVHostVarEnum_VT_DATE	7	Indicates a DATE value.
NAVHostVarEnum_VT_BSTR	8	Indicates a BSTR string.
NAVHostVarEnum_VT_DISPATCH	9	Indicates an IDispatch pointer.
NAVHostVarEnum_VT_ERROR	10	Indicates an SCODE.
NAVHostVarEnum_VT_BOOL	11	Indicates a Boolean value.
NAVHostVarEnum_VT_VARIANT	12	Indicates a VARIANT far pointer.
NAVHostVarEnum_VT_UNKNOWN	13	Indicates a IUnknown pointer.
NAVHostVarEnum_VT_DECIMAL	14	Indicates a decimal value.
NAVHostVarEnum_VT_I1	16	Indicates a char value.
NAVHostVarEnum_VT_UI1	17	Indicates a byte.
NAVHostVarEnum_VT_UI2	18	Indicates an unsignedshort.
NAVHostVarEnum_VT_UI4	19	Indicates an unsignedlong.
NAVHostVarEnum_VT_I8	20	Indicates a 64-bit integer.
NAVHostVarEnum_VT_UI8	21	Indicates an 64-bit unsigned integer.
NAVHostVarEnum_VT_INT	22	Indicates an integer value.
NAVHostVarEnum_VT_UINT	23	Indicates an unsigned integer value.
NAVHostVarEnum_VT_VOID	24	Indicates a C style void.
NAVHostVarEnum_VT_HRESULT	25	Indicates an HRESULT.
NAVHostVarEnum_VT_PTR	26	Indicates a pointer type.

NAVHostVarEnum_VT_SAFEARRAY	27	Indicates a SAFEARRAY. Not valid in a VARIANT.
NAVHostVarEnum_VT_CARRAY	28	Indicates a C style array.
NAVHostVarEnum_VT_USERDEFINED	29	Indicates a user defined type.
NAVHostVarEnum_VT_LPSTR	30	Indicates a null-terminated string.
NAVHostVarEnum_VT_LPWSTR	31	Indicates a wide string terminated by null.
NAVHostVarEnum_VT_RECORD	36	Indicates a user defined type.
NAVHostVarEnum_VT_FILETIME	64	Indicates a FILETIME value.
NAVHostVarEnum_VT_BLOB	65	Indicates length prefixed bytes.
NAVHostVarEnum_VT_STREAM	66	Indicates that the name of a stream follows.
NAVHostVarEnum_VT_STORAGE	67	Indicates that the name of a storage follows.
NAVHostVarEnum_VT_STREAMED_OBJECT	68	Indicates that a stream contains an object.
NAVHostVarEnum_VT_STORED_OBJECT	69	Indicates that a storage contains an object.
NAVHostVarEnum_VT_BLOB_OBJECT	70	Indicates that a blob contains an object.
NAVHostVarEnum_VT_CF	71	Indicates the clipboard format.
NAVHostVarEnum_VT_CLSID	72	Indicates a class ID.
NAVHostVarEnum_VT_VECTOR	4096	Indicates a simple, counted array.
NAVHostVarEnum_VT_ARRAY	8192	Indicates a SAFEARRAY pointer.
NAVHostVarEnum_VT_BYREF	16384	Indicates that a value is a reference.

NAVHostCtrl object

The Exontrol's NAVHost/NET assembly is a Control-AddIn for Microsoft Dynamics NAV that allows you to use any UI element of the /NET Framework on any page. For instance, 'Height=32; AssemblyQualifiedName = "System.Windows.Forms.TrackBar, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"; Host.Template = "Value = 5"' adds a track-bar to your NAV form.

Name	Description
AssemblyLocation	Specifies the fully qualified path of the assembly/file to load.
AssemblyName	Specifies the assembly name of the type, to be created (requires AssemblyLocation).
AssemblyQualifiedName	Specifies the assembly-qualified name of the type, to be created.
BackgroundColor	Specifies the hosting's background color.
Create	Creates/Loads the assembly giving fully qualified path of the assembly/file or/and the assembly/qualified name of the type to be created.
Destroy	Destroys the control and unloads the assembly.
Host	Gets the object being hosted.
HostEvents	Specifies the list of events to be handled through the control's Event event, else all events are handled (missing or not set).
hWnd	Indicates the handle to the window (HWND)that hosts the assembly.
IsCreated	Specifies if the assembly is loaded and the control created.
Template	Executes x-script code.
Version	Indicates the version of the NAVHost control.

property NAVHostCtrl.AssemblyLocation as String

Specifies the fully qualified path of the assembly/file to load.

Type	Description
String	A String expression that specifies the full path or UNC location of the loaded file that contains the manifest/component/assembly.

By default, the AssemblyLocation property is empty. The AssemblyLocation property specifies the full path or UNC location of the loaded file that contains the manifest/component/assembly. The assemblyLocation parameter of the Create method indicates the same value as AssemblyLocation property. In /NET framework, the AssemblyLocation property is similar with the Location property of System.Reflection.Assembly class. The Destroy method unloads the hosting control.

There are three ways of loading/creating a manifest/component/assembly as listed:

- AssemblyLocation property, loads the first public, browseable control found in the specified location. If there are more public, browseable controls in the assembly, you can use the AssemblyName or AssemblyQualifiedName property to specify the fully qualified name of the type to be hosted.
- AssemblyQualifiedName property, loads and creates the object based on the assembly-qualified name of the type, which includes the name of the assembly from which this type object was loaded. *Sample: "System.Windows.Forms.ListView, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"*
- Create(assemblyLocation, assemblyName) method loads the specified file, and creates the giving type.

If succeeded, (the assembly is loaded and the object is created), the

- IsCreated property returns the type of the object being created such as: NAVHostType_exNAVHostControl, if an object of System.Windows.Forms.Control type was created or NAVHostType_exNAVHostObject a generic object was created.
- AssemblyLocation property returns the location of the loaded file that contains the manifest/component/assembly. *Sample: "C:\Windows\assembly\GAC_MSIL\System.Windows.Forms\2.0.0.0__b77a5c561934e*
- AssemblyName property returns the fully qualified name of the type, including its namespace but not its assembly. *Sample: "System.Windows.Forms.ScrollableControl"*
- AssemblyQualifiedName property gets the assembly-qualified name of the type, which includes the name of the assembly from which this type object was loaded. *Sample: "System.Windows.Forms.ScrollableControl, System.Windows.Forms,*

Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"

The Host property returns the object being hosted by the NAVHost control.

If fails, the

- IsCreated property returns NAVHostType_exNAVHostNothing, which indicates no object has been created
- AssemblyLocation, AssemblyName and AssemblyQualifiedName return empty string
- Host property returns nothing.

property NAVHostCtrl.AssemblyName as String

Specifies the assembly name of the type, to be created (requires AssemblyLocation).

Type	Description
String	A string expression that specifies the fully qualified name of the type, including its namespace but not its assembly.

By default, the AssemblyName property is empty. The AssemblyName property specifies the fully qualified name of the type, including its namespace but not its assembly. The assemblyName parameter of the Create method indicates the same value as AssemblyName property. In /NET framework, the AssemblyName property is similar with the FullName property of System.Type class. The Destroy method unloads the hosting control.

There are three ways of loading/creating a manifest/component/assembly as listed:

- AssemblyLocation property, loads the first public, browseable control found in the specified location. If there are more public, browseable controls in the assembly, you can use the AssemblyName or AssemblyQualifiedName property to specify the fully qualified name of the type to be hosted.
- AssemblyQualifiedName property, loads and creates the object based on the assembly-qualified name of the type, which includes the name of the assembly from which this type object was loaded. *Sample: "System.Windows.Forms.ListView, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"*
- Create(assemblyLocation, assemblyName) method loads the specified file, and creates the giving type.

If succeeded, (the assembly is loaded and the object is created), the

- IsCreated property returns the type of the object being created such as: NAVHostType_exNAVHostControl, if an object of System.Windows.Forms.Control type was created or NAVHostType_exNAVHostObject a generic object was created.
- AssemblyLocation property returns the location of the loaded file that contains the manifest/component/assembly. *Sample: "C:\Windows\assembly\GAC_MSIL\System.Windows.Forms\2.0.0.0__b77a5c561934e*
- AssemblyName property returns the fully qualified name of the type, including its namespace but not its assembly. *Sample: "System.Windows.Forms.ScrollableControl"*
- AssemblyQualifiedName property gets the assembly-qualified name of the type, which includes the name of the assembly from which this type object was loaded. *Sample: "System.Windows.Forms.ScrollableControl, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"*

The Host property returns the object being hosted by the NAVHost control.

If fails, the

- IsCreated property returns NAVHostType_exNAVHostNothing, which indicates no object has been created
- AssemblyLocation, AssemblyName and AssemblyQualifiedName return empty string
- Host property returns nothing.

property NAVHostCtrl.AssemblyQualifiedName as String

Specifies the assembly-qualified name of the type, to be created.

Type	Description
String	A String expression that specifies the assembly-qualified name of the type, which includes the name of the assembly from which this type object should be created.

By default, the AssemblyQualifiedName property is empty. The AssemblyQualifiedName property specifies the assembly-qualified name of the type, which includes the name of the assembly from which this type object should be created. In /NET framework, the AssemblyQualifiedName property is similar with the AssemblyQualifiedName property of System.Type class. The Destroy method unloads the hosting control.

There are three ways of loading/creating a manifest/component/assembly as listed:

- AssemblyLocation property, loads the first public, browseable control found in the specified location. If there are more public, browseable controls in the assembly, you can use the AssemblyName or AssemblyQualifiedName property to specify the fully qualified name of the type to be hosted.
- AssemblyQualifiedName property, loads and creates the object based on the assembly-qualified name of the type, which includes the name of the assembly from which this type object was loaded. *Sample: "System.Windows.Forms.ListView, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"*
- Create(assemblyLocation, assemblyName) method loads the specified file, and creates the giving type.

If succeeded, (the assembly is loaded and the object is created), the

- IsCreated property returns the type of the object being created such as: NAVHostType_exNAVHostControl, if an object of System.Windows.Forms.Control type was created or NAVHostType_exNAVHostObject a generic object was created.
- AssemblyLocation property returns the location of the loaded file that contains the manifest/component/assembly. *Sample: "C:\Windows\assembly\GAC_MSIL\System.Windows.Forms\2.0.0.0__b77a5c561934e*
- AssemblyName property returns the fully qualified name of the type, including its namespace but not its assembly. *Sample: "System.Windows.Forms.ScrollableControl"*
- AssemblyQualifiedName property gets the assembly-qualified name of the type, which includes the name of the assembly from which this type object was loaded. *Sample: "System.Windows.Forms.ScrollableControl, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"*

The Host property returns the object being hosted by the NAVHost control.

If fails, the

- IsCreated property returns NAVHostType_exNAVHostNothing, which indicates no object has been created
- AssemblyLocation, AssemblyName and AssemblyQualifiedName return empty string
- Host property returns nothing.

property NAVHostCtrl.BackgroundColor as Long

Specifies the hosting's background color.

Type	Description
Long	A Long expression that specifies the color (RGB color) to be applied to the host's background.

By default, the BackgroundColor property indicates the NAVHost container's background color. Use the BackgroundColor property to apply a different background color to the NAVHost control. The BackgroundColor property does not change the background color of the hosting control. *To change the background color of the hosting control, you need to consult the hosting control's documentation, and use it in a Template or Item property like in the following samples. Most of the controls provide a BackColor property that change the control's background color, and so that's the property it must be used to change the hosting control's background color.* The BackgroundColor property changes the background color behind the hosting control.

method NAVHostCtrl.Create (AssemblyLocation as String, AssemblyName as String)

Creates/Loads the assembly giving fully qualified path of the assembly/file or/and the assembly/qualified name of the type to be created.

Type	Description
AssemblyLocation as String	A String expression that specifies the full path or UNC location of the loaded file that contains the manifest/component/assembly. *Sample: "C:\Windows\assembly\GAC_MSIL\System.Windows.Form*
AssemblyName as String	A String expression that specifies the fully qualified name of the type, including its namespace but not its assembly. *Sample: "System.Windows.Forms.TreeView"*

Return	Description
NAVHostType	A NAVHostType expression that specifies the type of the object being created, the same as value being returned by the IsCreated property. The return value can be one of the following: • **0**/ NAVHostType_exNAVHostNothing, The NAVHost control hosts nothing. • **1**/ NAVHostType_exNAVHostControl, The NAVHost component hosts an object of System.Windows.Forms.Control type. • **2**/ NAVHostType_exNAVHostObject, The NAVHost component hosts a general object.

The Create method loads the specified file (AssemblyLocation), and creates the giving type (AssemblyName). The assemblyLocation parameter of the Create method indicates the same value as AssemblyLocation property. The assemblyName parameter of the Create method indicates the same value as AssemblyName property. The Create method returns the type of the object being created, the same as IsCreated property returns the type of the object being created such as: NAVHostType_exNAVHostControl, if an object of System.Windows.Forms.Control type was created or NAVHostType_exNAVHostObject a generic object was created. The Host property returns the object being created and hosted by the NAVHost control. The Destroy method unloads the hosting control.

There are three ways of loading/creating a manifest/component/assembly as listed:

- AssemblyLocation property, loads the first public, browseable control found in the

specified location. If there are more public, browseable controls in the assembly, you can use the AssemblyName or AssemblyQualifiedName property to specify the fully qualified name of the type to be hosted.
- AssemblyQualifiedName property, loads and creates the object based on the assembly-qualified name of the type, which includes the name of the assembly from which this type object was loaded. *Sample: "System.Windows.Forms.ListView, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"*
- Create(assemblyLocation, assemblyName) method loads the specified file, and creates the giving type.

If succeeded, (the assembly is loaded and the object is created), the

- IsCreated property returns the type of the object being created such as: NAVHostType_exNAVHostControl, if an object of System.Windows.Forms.Control type was created or NAVHostType_exNAVHostObject a generic object was created.
- AssemblyLocation property returns the location of the loaded file that contains the manifest/component/assembly. *Sample: "C:\Windows\assembly\GAC_MSIL\System.Windows.Forms\2.0.0.0__b77a5c561934e*
- AssemblyName property returns the fully qualified name of the type, including its namespace but not its assembly. *Sample: "System.Windows.Forms.ScrollableControl"*
- AssemblyQualifiedName property gets the assembly-qualified name of the type, which includes the name of the assembly from which this type object was loaded. *Sample: "System.Windows.Forms.ScrollableControl, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"*

If fails, the

- IsCreated property returns NAVHostType_exNAVHostNothing, which indicates no object has been created
- AssemblyLocation, AssemblyName and AssemblyQualifiedName return empty string
- Host property returns nothing.

method NAVHostCtrl.Destroy ()

Destroys the control and unloads the assembly.

Type	Description

The Destroy method unloads the hosting control. The Create(assemblyLocation, assemblyName) method loads the specified file, and creates the giving type.

If Destroy method is called, the

- IsCreated property returns NAVHostType_exNAVHostNothing, which indicates no object has been created
- AssemblyLocation, AssemblyName and AssemblyQualifiedName return empty string
- Host property returns nothing.

There are three ways of loading/creating a manifest/component/assembly as listed:

- AssemblyLocation property, loads the first public, browseable control found in the specified location. If there are more public, browseable controls in the assembly, you can use the AssemblyName or AssemblyQualifiedName property to specify the fully qualified name of the type to be hosted.
- AssemblyQualifiedName property, loads and creates the object based on the assembly-qualified name of the type, which includes the name of the assembly from which this type object was loaded. *Sample: "System.Windows.Forms.ListView, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"*
- Create(assemblyLocation, assemblyName) method loads the specified file, and creates the giving type.

property NAVHostCtrl.Host as NAVHostObject

Gets the object being hosted.

Type	Description
NAVHostObject	A NAVHostObject object that holds a reference to the hosting control. The Value property of the NAVHostObject object specifies the original object being hosted.

By default, the Host property holds nothing. The Host property returns the object being hosted by the NAVHost control. Use the AssemblyLocation, AssemblyQualifiedName or Create method to create and host a specified type. **The hosting control's properties or methods must be called using the Item, SetTemplateDef or Template property.** The HostEvent event notifies your application once the hosting control fires an event. The HostEvents property of the NAVHost control specifies the list of events that the control should handle.

There are three ways of loading/creating a manifest/component/assembly as listed:

- AssemblyLocation property, loads the first public, browseable control found in the specified location. If there are more public, browseable controls in the assembly, you can use the AssemblyName or AssemblyQualifiedName property to specify the fully qualified name of the type to be hosted.
- AssemblyQualifiedName property, loads and creates the object based on the assembly-qualified name of the type, which includes the name of the assembly from which this type object was loaded. *Sample: "System.Windows.Forms.ListView, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"*
- Create(assemblyLocation, assemblyName) method loads the specified file, and creates the giving type.

If succeeded, (the assembly is loaded and the object is created), the

- IsCreated property returns the type of the object being created such as: NAVHostType_exNAVHostControl, if an object of System.Windows.Forms.Control type was created or NAVHostType_exNAVHostObject a generic object was created.
- AssemblyLocation property returns the location of the loaded file that contains the manifest/component/assembly. *Sample: "C:\Windows\assembly\GAC_MSIL\System.Windows.Forms\2.0.0.0__b77a5c561934e*
- AssemblyName property returns the fully qualified name of the type, including its namespace but not its assembly. *Sample: "System.Windows.Forms.ScrollableControl"*
- AssemblyQualifiedName property gets the assembly-qualified name of the type, which includes the name of the assembly from which this type object was loaded. *Sample:*

"System.Windows.Forms.ScrollableControl, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"

The following samples show how you can call hosting control's properties or methods:

- The *Host.Item("GetType().Name").Value* gets the type of the hosting control.
- The *Host.Item("GetType().BaseType().Name").Value* property gets the type of the base of the hosting control.
- The *Host.Item("GetType().IsCOMObject()").Value*, property indicates if the hosting control is /COM object or a/NET assembly.
- The *Host.Item("GetType().GUID()").Value.AsString*, returns the GUID of the hosting control.

property NAVHostCtrl.HostEvents as String

Specifies the list of events to be handled through the control's Event event, else all events are handled (missing or not set).

Type	Description
String	A String expression that specifies the list of events to be handled through the control's Event event, else all events are handled (missing or not set). The list of events is separated by any of the following characters: ' ', ',', '.', ':', '\t'

By default, the HostEvents property is empty, which indicates that all events of the hosting event are fired through the NAVHost's HostEvent event. The HostEvent event notifies your application once the hosting control (Host) fires an event. The HostEvents property of the NAVHost control specifies the list of events that the control should handle. The AsString property of the NAVHostEvent object gives a brief description of the event that occurred including the event's name, identifier and its list of arguments. Each control that the NAVHost host provides its own events, so for what events the hosting control supports consult its documentation.

Use the following properties to identify/filter the event:

- Name, Indicates the name of the event.
- ID, Indicates the identifier of the event. The ID property may give different values for different versions of hosting control, so you must check for compatibility, so it is not guaranteed that the ID will be unique for any version of the hosting control.
- HostEvents property specifies the list of events to be handled through the control's Event event, else all events are handled (missing or not set).

property NAVHostCtrl.hWnd as Long

Indicates the handle to the window (HWND) that hosts the assembly.

Type	Description
Long	A Long expression that indicates the handle to the window (HWND) that hosts the assembly.

The hWnd property returns the handle of the window that displays the NAVHost control. The hWnd property does not return the handle of the hosting control. The Microsoft Windows operating environment identifies each form and control in an application by assigning it a handle, or hWnd. The hWnd property is used with Windows API calls. Many Windows operating environment functions require the hWnd of the active window as an argument. The Host property returns the object being created and hosted by the NAVHost control. **The hosting control's properties or methods must be called using the Item, SetTemplateDef or Template property.**

property NAVHostCtrl.IsCreated as NAVHostType

Specifies if the assembly is loaded and the control created.

Type	Description
NAVHostType	A NAVHostType expression that specifies the type of object being hosted which can be one of the following: • **0**/NAVHostType_exNAVHostNothing, The NAVHost control hosts nothing. • **1**/NAVHostType_exNAVHostControl, The NAVHost component hosts an object of System.Windows.Forms.Control type. • **2**/NAVHostType_exNAVHostObject, The NAVHost component hosts a general object.

By default, the IsCreated property is NAVHostType_exNAVHostNothing, which indicates that the NAVHost control hosts nothing. The IsCreated property specifies the type of the object the NAVHost control hosts. The Create(assemblyLocation, assemblyName) method loads the specified file, and creates the giving type. The Create method returns the same value as IsCreated property. The Destroy method unloads the hosting control.

- The *Host.Item("GetType().Name").Value* gets the type of the hosting control.
- The *Host.Item("GetType().BaseType().Name").Value* property gets the type of the base of the hosting control.
- The *Host.Item("GetType().IsCOMObject()").Value*, property indicates if the hosting control is /COM object or a/NET assembly.
- The *Host.Item("GetType().GUID()").Value.AsString*, returns the GUID of the hosting control.

There are three ways of loading/creating a manifest/component/assembly as listed:

- AssemblyLocation property, loads the first public, browseable control found in the specified location. If there are more public, browseable controls in the assembly, you can use the AssemblyName or AssemblyQualifiedName property to specify the fully qualified name of the type to be hosted.
- AssemblyQualifiedName property, loads and creates the object based on the assembly-qualified name of the type, which includes the name of the assembly from which this type object was loaded. *Sample: "System.Windows.Forms.ListView, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"*
- Create(assemblyLocation, assemblyName) method loads the specified file, and creates the giving type.

If succeeded, (the assembly is loaded and the object is created), the

- IsCreated property returns the type of the object being created such as: NAVHostType_exNAVHostControl, if an object of System.Windows.Forms.Control type was created or NAVHostType_exNAVHostObject a generic object was created.
- AssemblyLocation property returns the location of the loaded file that contains the manifest/component/assembly. *Sample: "C:\Windows\assembly\GAC_MSIL\System.Windows.Forms\2.0.0.0__b77a5c561934e*
- AssemblyName property returns the fully qualified name of the type, including its namespace but not its assembly. *Sample: "System.Windows.Forms.ScrollableControl"*
- AssemblyQualifiedName property gets the assembly-qualified name of the type, which includes the name of the assembly from which this type object was loaded. *Sample: "System.Windows.Forms.ScrollableControl, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"*

The Host property returns the object being hosted by the NAVHost control.

If fails, the

- IsCreated property returns NAVHostType_exNAVHostNothing, which indicates no object has been created
- AssemblyLocation, AssemblyName and AssemblyQualifiedName return empty string
- Host property returns nothing.

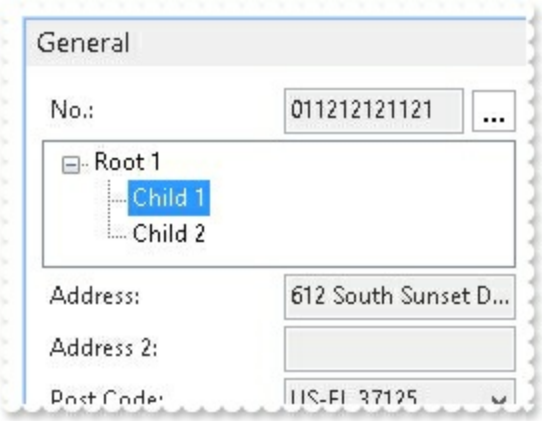
property NAVHostCtrl.Template as String

Executes x-script code.

Type	Description
String	A String expression that specifies the x-script code to be executed.

The Caption property of any field (whose ControlAddIn property value is 'Exontrol.NAVHost;PublicKeyToken=aaab53cf43a9de9d'), supports x-script or template script. The Template/ x-script code is a simple way of calling control/object's properties, methods/ events using strings. Exontrol owns the x-script implementation in its easiest way and it does not require any VB engine to get executed. Our simple rule is using the component alone without any other dependency than the Windows system.

For instance Caption property on: 'Height=64;AssemblyQualifiedName = "System.Windows.Forms.TreeView, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089";Host.Item("Nodes.Add(`Root 1`)").Template = "Nodes{ Add(`Child 1`); Add(`Child 2`) }; Expand() }"' generates a result such as:



or in other words inserts a TreeView control with a root and two-child elements.

The **Template/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>
<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>
```

where:

<identifier> indicates an identifier of the variable, property or method, and should start with a letter.
<type> indicates the type the CreateObject function creates, as the assembly-qualified name of the type to create.
<text> any string of characters

The Template / x-script is composed by lines of instructions. Instructions are separated by "\r\n" (new line characters) or ";" character. The TemplateThrowError property specifies whether the control fires an exception/error when the Template call fails. The TemplateError / TemplateException gets the error if the Template calls fails. The TemplateResult property returns the result of the last instruction into a Template call, as a NAVObjectTemplate object.

An x-script instruction/line can be one of the following:

- **Dim variable**[, variable, ...] *declares the variables in the context. Multiple variables are separated by commas. The SetTemplateDef method can declare new variables to*

be available for the main context. (Sample: Dim h, h1, h2)

- **variable =** [object.][property/method(arguments).]**property/method(arguments)** *assigns the result of the **property/method** call to the **variable**. (Sample: h = Nodes.Add(`Node`))*
- [object.][property/method(arguments).]**property(arguments) = value** *assigns the **value** to the **property.** (Sample: Nodes.Add(`Node`).BackColor = RGB(255,0,0))*
- [object.][property/method(arguments).]**property/method(arguments)** *invokes the **property/method**. (Sample: Nodes.Add(`Node`))*
- **{**context **}** *delimits the object's context. The properties/fields or methods called between { and } are related to the last object returned by the property/method prior to { declaration. (Sample: Nodes{Add(`Child 1`);Add(`Child 2`)})*
- ***.** delimits the object than its property or method. (Sample: Nodes.Add(`Element`), or Nodes.Add(`Element`) and Nodes{Add(`Element`)} are equivalents)*

where

- ***variable** is the name of a variable declared with Dim command or previously defined using the SetTemplateDef method.*
- ***property** is the name of a property/field of the current object in the current context.*
- ***method** is the name of a method of the current object in the current context.*
- ***arguments** include constants and/or variables and/or property/method calls separated by comma character.*
- ***object** can be a variable of an Object type, **Me** or **CreateObject** call.*

The x-script uses constant expressions as follows:

- *boolean* expression with possible values as ***True*** or ***False**. The True value is equivalent with -1, while False with 0. (Sample: Visible = 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. (Sample: BackColor = 0xFF0000)*
- *date* expression is delimited by **#** character in the format **#mm/dd/yyyy hh:mm:ss#**. *For instance, #31/12/1971# indicates the December 31, 1971 (Sample: Chart.FirstVisibleDate = #1/1/2001#)*
- *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"** or **`text`** indicates the string text, while the ' text , specifies the comment text. (Sample: Text = "caption")*

Also , the template or x-script code supports general functions as follows:

- **Me** *property indicates the original object, and it is defined as a predefined variable. (*

Sample: Me.Nodes.Add(`Root 1`))

- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the Red Green Blue bytes for the color being specified. (Sample: Nodes.Add(`Root 1`).BackColor = RGB(255,0,0))*
- **LoadPicture(**file**)** *property loads a picture from a file and returns a Picture object required by the picture properties. (Sample: BackgroundImage = LoadPicture(`C:\exontrol\images\auction.gif`)*
- **CreateObject(***assemblyQualifiedName***)** *property creates an instance of the specified type using that type's default constructor. The assemblyQualifiedName indicates the assembly-qualified name of the type to get. See AssemblyQualifiedName. If the type is in the currently executing assembly or in Mscorlib.dll, it is sufficient to supply the type name qualified by its namespace. (Sample: "CreateObject(`System.Windows.Forms.TabPage, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089`){Text = `Page`;UseVisualStyleBackColor = True}")*

property NAVHostCtrl.Version as String

Indicates the version of the NAVHost control.

Type	Description
String	A string expression that indicates the control's version.

The Version property specifies the NAVHost control's version. If the Version property includes the DEMO, it indicates that you are running a trial version of the NAVHost control. The Version property does not get the hosting control's Version, for that you have to use the AssemblyVersion property of the Host object.

NAVHostCtrlEvents object

The NAVHostEvent object holds information about the event that the Host control fires. The HostEvent event notifies your application once the hosting control (Host) fires an event. The HostEvents property of the NAVHost control specifies the list of events that the control should handle.

Name	Description
HostEvent	The hosting control fires an event.

method NAVHostCtrlEvents.HostEvent (Ev as NAVHostEvent)

The hosting control fires an event.

Type	Description
Ev as NAVHostEvent	A NAVHostEvent object that holds information about the firing event.

The OnControlAddIn(Index : Integer;Data : Text) trigger notifies your application once the hosting control fires any event. The events being fired depends on the hosting control which usually is specified by the AssemblyQualifiedName call in the Caption property.

In order to handle the OnControlAddin trigger of the Control-AddIn in the NAV's page you have to:

- Open the NAV's page that hosts the Control-AddIn, in design mode
- Go to the field that hosts the Control-AddIn
- Right-Click the field, and choose "C/AL Code"
- Locate the "OnControlAddIn(Index : Integer;Data : Text)" for the field
- Add the Message(Data) as in the following sample:

```
Name - OnControlAddIn(Index : Integer;Data : Text)
  MESSAGE(Data);
```

- Save and Close the page
- Run the Page, and click the Control-AddIn, and so the message box "Click" is displayed.

The HostEvent event notifies your application once the hosting control (Host) fires an event. The HostEvents property of the NAVHost control specifies the list of events that the control should handle. The AsString property of the NAVHostEvent object gives a brief description of the event that occurred including the event's name, identifier and its list of arguments. Each control that the NAVHost host provides its own events, so for what events the hosting control supports consult its documentation. The Version property specifies the NAVHost control's version, which includes the DEMO if you are running the trial version of the control.

Use the following properties to identify/filter the event:

- Name, Indicates the name of the event.
- ID, Indicates the identifier of the event. The ID property may give different values for different versions of hosting control, so you must check for compatibility, so it is not guaranteed that the ID will be unique for any version of the hosting control.

- HostEvents property specifies the list of events to be handled through the control's Event event, else all events are handled (missing or not set).

Use the following properties to access the arguments of the event:

- AsString, Gives a brief description of the event including its arguments.
- Arguments, gives a NAVObjectTemplate object, whose Item or Template properties can be used to access the event's argument using the x-script language.

NAVHostEvent object

The NAVHostEvent object holds information about the event that the Host control fires. The HostEvent event notifies your application once the hosting control (Host) fires an event. The HostEvents property of the NAVHost control specifies the list of events that the control should handle. The NAVHostEvent object supports the following properties and methods:

Name	Description
Arguments	Gets the arguments of the event.
AsString	Gives a brief description of the event including its arguments.
ID	Indicates the identifier of the event.
Name	Indicates the name of the event.

property NAVHostEvent.Arguments as NAVObjectTemplate

Gets the arguments of the event.

Type	Description
NAVObjectTemplate	A NAVObjectTemplate object that holds arguments of the hosting event.

The Arguments property, gives a NAVObjectTemplate object, whose Item or Template properties can be used to access the event's argument using the x-script language. The AsString property gives a brief description including name, identifier and arguments of the event.

property NAVHostEvent.AsString as String

Gives a brief description of the event including its arguments.

Type	Description
String	A String expression that describes the event being fired including the event's name [event's identifier] { event's arguments }

The AsString property gives a general idea of what data the event contains. The Name property indicates the name of the event. The ID property indicates the identifier of the event. The Arguments property, gives a NAVObjectTemplate object, whose Item or Template properties can be used to access the event's argument using the x-script language. The AsString property may returns: "The trial/evaluation version of the control limits firing this event. In other words, using the trial/evaluation version won't fire the event every time it should.", only for not-registered version. The Version property specifies the NAVHost control's version, which includes the DEMO if you are running the trial version of the control.

property NAVHostEvent.ID as Long

Indicates the identifier of the event.

Type	Description
Long	A Long expression

The ID property indicates the identifier of the event. The ID property may give different values for different versions of hosting control, so you must check for compatibility, so it is not guaranteed that the ID will be unique for any version of the hosting control. The Name property indicates the name of the event. You can use the Name or ID property to identify a specified event you need to handle in the hosting control. The AsString property gives a brief description including name, identifier and arguments of the event.

The Arguments property, gives a NAVObjectTemplate object, whose Item or Template properties can be used to access the event's argument using the x-script language. The AsString property may returns: "The trial/evaluation version of the control limits firing this event. In other words, using the trial/evaluation version won't fire the event every time it should.", only for not-registered version.

property NAVHostEvent.Name as String

Indicates the name of the event.

Type	Description
String	A String expression that specifies the name of the event being fired.

The Name property indicates the name of the event. The ID property indicates the identifier of the event. You can use the Name or ID property to identify a specified event you need to handle in the hosting control. The AsString property gives a brief description including name, identifier and arguments of the event.

The Arguments property, gives a NAVObjectTemplate object, whose Item or Template properties can be used to access the event's argument using the x-script language. The AsString property may returns: "The trial/evaluation version of the control limits firing this event. In other words, using the trial/evaluation version won't fire the event every time it should.", only for not-registered version.

NAVHostObject object

The NAVHostObject object holds a .NET Framework object. The NAVHostObject type supports the following properties and method:

Name	Description
AsBoolean	Returns a boolean value that represents the current object's value.
AsDate	Returns a date value that represents the current object's value.
AsDouble	Returns a numeric value that represents the current object's value.
AsInt	Returns an integer value that represents the current object's value.
AssemblyVersion	Indicates the version of the assembly being loaded.
AsString	Returns a string that represents the current object's value.
Item	Executes the template and returns the result.
SetTemplateDef	Defines inside variables for the next Template/ExecuteTemplate call.
SetValue	Specifies the value of the object.
Template	Executes the x-script code.
TemplateError	Indicates the error code of the last Template call.
TemplateException	Indicates the detailed information about the exception that occurs.
TemplateResult	Indicates the result of the last Template call.
TemplateThrowError	Specifies whether the execution of the template stops once an error occurs.
Type	Indicates the type of the object's value.
Value	Specifies the value of the object.
VtType	Indicates the type/vartype of the object's value.

property NAVHostObject.AsBoolean as Boolean

Returns a boolean value that represents the current object's value.

Type	Description
Boolean	A Boolean expression that specifies the Value converted as boolean.

The AsBoolean property converts the Value to a Boolean expression. If the conversion is not possible, the AsBoolean property returns False. The Value property holds the original object. The VtType property indicates the VARIANT type of the object that the current NAVHostObject object holds. The Type property returns a string that specifies the fully assembly-qualified name of the type, which includes the name of the assembly from which this Type object is loaded. If the NAVHostObject holds a class or an object/IDispatch/IUnknown that supports properties, fields, members, any of these can be called through the NAVHostObject properties like: Item, SetTemplateDef or Template property.

You can use the following properties to convert the current Value to indicated standard types:

- AsBoolean, converts the value to a boolean expression.
- AsDate, converts the value to a DATE-TIME/double expression.
- AsDouble, converts the value to a double expression.
- AsInt, converts the value to an integer-32 expression.
- AsString, gets the value converted to a string expression.

property NAVHostObject.AsDate as Date

Returns a date value that represents the current object's value.

Type	Description
Date	A Date/Double expression that specifies the Value converted as date-time.

The AsDate property converts the Value to a DATE-TIME expression. If the conversion is not possible, the AsDate property returns 0. The Value property holds the original object. The VtType property indicates the VARIANT type of the object that the current NAVHostObject object holds. The Type property returns a string that specifies the fully assembly-qualified name of the type, which includes the name of the assembly from which this Type object is loaded. If the NAVHostObject holds a class or an object/IDispatch/IUnknown that supports properties, fields, members, any of these can be called through the NAVHostObject properties like: Item, SetTemplateDef or Template property.

You can use the following properties to convert the current Value to indicated standard types:

- AsBoolean, converts the value to a boolean expression.
- AsDate, converts the value to a DATE-TIME/double expression.
- AsDouble, converts the value to a double expression.
- AsInt, converts the value to an integer-32 expression.
- AsString, gets the value converted to a string expression.

property NAVHostObject.AsDouble as Double

Returns a numeric value that represents the current object's value.

Type	Description
Double	A Double expression that specifies the Value converted as double.

The AsDouble property converts the Value to a double expression. If the conversion is not possible, the AsDouble property returns 0. The Value property holds the original object. The VtType property indicates the VARIANT type of the object that the current NAVHostObject object holds. The Type property returns a string that specifies the fully assembly-qualified name of the type, which includes the name of the assembly from which this Type object is loaded. If the NAVHostObject holds a class or an object/IDispatch/IUnknown that supports properties, fields, members, any of these can be called through the NAVHostObject properties like: Item, SetTemplateDef or Template property.

You can use the following properties to convert the current Value to indicated standard types:

- AsBoolean, converts the value to a boolean expression.
- AsDate, converts the value to a DATE-TIME/double expression.
- AsDouble, converts the value to a double expression.
- AsInt, converts the value to an integer-32 expression.
- AsString, gets the value converted to a string expression.

property NAVHostObject.AsInt as Long

Returns an integer value that represents the current object's value.

Type	Description
Long	A Long expression that specifies the Value converted as long (32-bit integer).

The AsInt property converts the Value to a long expression. If the conversion is not possible, the AsInt property returns 0. The Value property holds the original object. The VtType property indicates the VARIANT type of the object that the current NAVHostObject object holds. The Type property returns a string that specifies the fully assembly-qualified name of the type, which includes the name of the assembly from which this Type object is loaded. If the NAVHostObject holds a class or an object/IDispatch/IUnknown that supports properties, fields, members, any of these can be called through the NAVHostObject properties like: Item, SetTemplateDef or Template property.

You can use the following properties to convert the current Value to indicated standard types:

- AsBoolean, converts the value to a boolean expression.
- AsDate, converts the value to a DATE-TIME/double expression.
- AsDouble, converts the value to a double expression.
- AsInt, converts the value to an integer-32 expression.
- AsString, gets the value converted to a string expression.

property NAVHostObject.AssemblyVersion as String

Indicates the version of the assembly being loaded.

Type	Description
String	A string expression that indicates the hosting control's version.

The AssemblyVersion property indicates the version of the assembly being loaded. The Version property specifies the NAVHost control's version.

property NAVHostObject.AsString as String

Returns a string that represents the current object's value.

Type	Description
String	A String expression that specifies the Value converted as string.

The AsString property converts the Value to a string expression. If the conversion is not possible, the AsString property returns "" (empty). The Value property holds the original object. The VtType property indicates the VARIANT type of the object that the current NAVHostObject object holds. The Type property returns a string that specifies the fully assembly-qualified name of the type, which includes the name of the assembly from which this Type object is loaded. If the NAVHostObject holds a class or an object/IDispatch/IUnknown that supports properties, fields, members, any of these can be called through the NAVHostObject properties like: Item, SetTemplateDef or Template property.

You can use the following properties to convert the current Value to indicated standard types:

- AsBoolean, converts the value to a boolean expression.
- AsDate, converts the value to a DATE-TIME/double expression.
- AsDouble, converts the value to a double expression.
- AsInt, converts the value to an integer-32 expression.
- AsString, gets the value converted to a string expression.

property NAVHostObject.Item (Template as String) as NAVObjectTemplate

Executes the template and returns the result.

Type	Description
Template as String	A String expression that specifies the x-script/template code to be executed.
NAVObjectTemplate	A NAVObjectTemplate property that holds the result of the last instruction within the Template.

Use the Template/Item property to get/set properties / fields / parameters, invoke methods of the hosting /NET framework Value, using the x-script code. The Item property does exactly the same thing as Template call, excepts that it returns the TemplateResult property. For instance, using the Template/Item property you can change the hosting control's background color, add nodes, and so on. Prior to Template/Item call, you can invoke the SetTemplateDef to define values from your code to Template's code (TemplateDef variables).

The Template / x-script is composed by lines of instructions. Instructions are separated by "\r\n" (new line characters) or ";" character. The TemplateThrowError property specifies whether the control fires an exception/error when the Template call fails. The TemplateError / TemplateException gets the error if the Template calls fails. The TemplateResult property returns the result of the last instruction into a Template call, as a NAVObjectTemplate object.

An x-script instruction/line can be one of the following:

- **Dim variable**[, variable, ...] *declares the variables in the context. Multiple variables are separated by commas. The SetTemplateDef method can declare new variables to be available for the main context. (Sample: Dim h, h1, h2)*
- **variable =** [object.][property/method(arguments).]**property/method(arguments)** *assigns the result of the **property/method** call to the **variable**. (Sample: h = Nodes.Add(`Node`))*
- [object.][property/method(arguments).]**property(arguments) = value** *assigns the **value** to the **property.** (Sample: Nodes.Add(`Node`).BackColor = RGB(255,0,0))*
- [object.][property/method(arguments).]**property/method(arguments)** *invokes the **property/method**. (Sample: Nodes.Add(`Node`))*
- **{**context **}** *delimits the object's context. The properties/fields or methods called between { and } are related to the last object returned by the property/method prior to { declaration. (Sample: Nodes{Add(`Child 1`);Add(`Child 2`)})*
- **.** *delimits the object than its property or method. (Sample: Nodes.Add(`Element`), or Nodes.Add(`Element`) and Nodes{Add(`Element`)} are equivalents)*

where

- ***variable*** *is the name of a variable declared with Dim command or previously defined using the SetTemplateDef method.*
- ***property*** *is the name of a property/field of the current object in the current context.*
- ***method*** *is the name of a method of the current object in the current context.*
- ***arguments*** *include constants and/or variables and/or property/method calls separated by comma character.*
- ***object*** *can be a variable of an Object type,* ***Me*** *or* ***CreateObject*** *call.*

The x-script uses constant expressions as follows:

- *boolean* expression with possible values as ***True*** or ***False****. The True value is equivalent with -1, while False with 0. (Sample: Visible = 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. (Sample: BackColor = 0xFF0000)*
- *date* expression is delimited by **#** character in the format **#mm/dd/yyyy hh:mm:ss#**. *Sample: #31/12/1971# indicates the December 31, 1971 (Sample: FirstVisibleDate = #1/1/2001#)*
- *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"*** *or* ***`text`*** *indicates the string text, while the ' text , specifies the comment text. (Sample: Text = "caption")*

Also , the template or x-script code supports general functions as follows:

- **Me** *property indicates the original object, and it is defined as a predefined variable. (Sample: Me.Nodes.Add(`Root 1`))*
- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the Red Green Blue bytes for the color being specified. (Sample: Nodes.Add(`Root 1`).BackColor = RGB(255,0,0))*
- **LoadPicture(**file**)** *property loads a picture from a file and returns a Picture object required by the picture properties. (Sample: BackgroundImage = LoadPicture(`C:\exontrol\images\auction.gif`)*
- **CreateObject(***assemblyQualifiedName***)** *property creates an instance of the specified type using that type's default constructor. The assemblyQualifiedName indicates the assembly-qualified name of the type to get. See AssemblyQualifiedName. If the type is in the currently executing assembly or in Mscorlib.dll, it is sufficient to supply the type name qualified by its namespace. (Sample: "CreateObject(`System.Windows.Forms.TabPage, System.Windows.Forms,*

Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089`){Text = `Page`;UseVisualStyleBackColor = True}")

The **Template/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>
<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>
```

where:

<identifier> indicates an identifier of the variable, property or method, and should start with a letter.
<type> indicates the type the CreateObject function creates, as the assembly-qualified

name of the type to create.
<text> any string of characters

method NAVHostObject.SetTemplateDef (Value as Variant)

Defines inside variables for the next Template/ExecuteTemplate call.

Type	Description
Value as Variant	If calling the first time, A String expression that indicates the DIM command to define the variables that follows, or a VARIANT expression that defines the value of the variable in the order as they were defined.

The SetTemplateDef method was provided to let you use values/objects inside the next Template/Item call. For instance, let's say you have a date field in your form, and once the user fills it, you want a /NET Frameworks MonthCalendar object to select it. In order to do that you have to call a code like:

```
With NAVHost1
   .BackgroundColor = 16777215
   .Create
"C:\Windows\assembly\GAC_MSIL\System.Windows.Forms\2.0.0.0__b77a5c561934e089\Sy

   With .Host
      .SetTemplateDef "Dim x"
      .SetTemplateDef #1/1/2001#
      .Template = "MaxSelectionCount = 1;SelectionStart = x"
   End With
End With
```

This sample defines the variable x to be 1/1/2001 for the Template call, so the SelectionStart will be set on 1/1/2001.

The call of SetTemplateDef method consists in:

- **First** call of SetTemplateDef method should be on a form of SetTemplateDef("Dim variable[,variable,...]"). This defines the name of the variable that follow to be defined.
- **Next** calls, must be exactly the same as with the number of variables you defined, which will define the variable one by one. For instance, if your first call was SetTemplateDef("Dim h1,h2,h3"), it means that the next-three calls of SetTemplateDef defines the variable h1, h2 and h3

Once you defined the variables, they will be available for the next calls of Template/Item properties.

method NAVHostObject.SetValue (Value as Variant)

Specifies the value of the object.

Type	Description
Value as Variant	A VARIANT expression that specifies the new value to be assigned to the NAVHostObject object.

Use the SetValue property to change the object being hosted by the current NAVHostObject object. The Value property holds the original object. The VtType property indicates the VARIANT type of the object that the current NAVHostObject object holds. The Type property returns a string that specifies the fully assembly-qualified name of the type, which includes the name of the assembly from which this Type object is loaded. If the NAVHostObject holds a class or an object/IDispatch/IUnknown that supports properties, fields, members, any of these can be called through the NAVHostObject properties like: Item, SetTemplateDef or Template property.

You can use the following properties to convert the current Value to indicated standard types:

- AsBoolean, converts the value to a boolean expression.
- AsDate, converts the value to a DATE-TIME/double expression.
- AsDouble, converts the value to a double expression.
- AsInt, converts the value to an integer-32 expression.
- AsString, gets the value converted to a string expression.

property NAVHostObject.Template as String

Executes the x-script code.

Type	Description
String	A String expression that specifies the x-script/template code to be executed.

Use the Template/Item property to get/set properties / fields / parameters, invoke methods of the hosting /NET framework Value, using the x-script code. The Item property does exactly the same thing as Template call, excepts that it returns the TemplateResult property. For instance, using the Template/Item property you can change the hosting control's background color, add nodes, and so on. Prior to Template/Item call, you can invoke the SetTemplateDef to define values from your code to Template's code (TemplateDef variables).

The **Template/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>
<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>
```

where:

<identifier> indicates an identifier of the variable, property or method, and should start with a letter.
<type> indicates the type the CreateObject function creates, as the assembly-qualified name of the type to create.
<text> any string of characters

The Template / x-script is composed by lines of instructions. Instructions are separated by "\r\n" (new line characters) or ";" character. The TemplateThrowError property specifies whether the control fires an exception/error when the Template call fails. The TemplateError / TemplateException gets the error if the Template calls fails. The TemplateResult property returns the result of the last instruction into a Template call, as a NAVObjectTemplate object.

An x-script instruction/line can be one of the following:

- **Dim variable**[, variable, ...] *declares the variables in the context. Multiple variables are separated by commas. The SetTemplateDef method can declare new variables to be available for the main context. (Sample: Dim h, h1, h2)*
- **variable =** [object.][property/method(arguments).]**property/method(arguments)** *assigns the result of the **property/method** call to the **variable**. (Sample: h = Nodes.Add(`Node`))*
- [object.][property/method(arguments).]**property(arguments) = value** *assigns the **value** to the **property.** (Sample: Nodes.Add(`Node`).BackColor = RGB(255,0,0))*
- [object.][property/method(arguments).]**property/method(arguments)** *invokes the **property/method**. (Sample: Nodes.Add(`Node`))*
- **{**context **}** *delimits the object's context. The properties/fields or methods called between { and } are related to the last object returned by the property/method prior to { declaration. (Sample: Nodes{Add(`Child 1`);Add(`Child 2`)})*
- **.** *delimits the object than its property or method. (Sample: Nodes.Add(`Element`), or Nodes.Add(`Element`) and Nodes{Add(`Element`)} are equivalents)*

where

- ***variable*** *is the name of a variable declared with Dim command or previously defined using the SetTemplateDef method.*
- ***property*** *is the name of a property/field of the current object in the current context.*
- ***method*** *is the name of a method of the current object in the current context.*
- ***arguments*** *include constants and/or variables and/or property/method calls separated by comma character.*
- ***object*** *can be a variable of an Object type,* ***Me*** *or* ***CreateObject*** *call.*

The x-script uses constant expressions as follows:

- *boolean* expression with possible values as ***True*** or ***False****. The True value is equivalent with -1, while False with 0. (Sample: Visible = 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. (Sample: BackColor = 0xFF0000)*
- *date* expression is delimited by **#** character in the format **#mm/dd/yyyy hh:mm:ss#**. *Sample: #31/12/1971# indicates the December 31, 1971 (Sample: FirstVisibleDate = #1/1/2001#)*
- *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"*** *or* ***`text`*** *indicates the string text, while the ' text , specifies the comment text. (Sample: Text = "caption")*

Also , the template or x-script code supports general functions as follows:

- **Me** *property indicates the original object, and it is defined as a predefined variable. (Sample: Me.Nodes.Add(`Root 1`))*
- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the Red Green Blue bytes for the color being specified. (Sample: Nodes.Add(`Root 1`).BackColor = RGB(255,0,0))*
- **LoadPicture(**file**)** *property loads a picture from a file and returns a Picture object required by the picture properties. (Sample: BackgroundImage = LoadPicture(`C:\exontrol\images\auction.gif`)*
- **CreateObject(***assemblyQualifiedName***)** *property creates an instance of the specified type using that type's default constructor. The assemblyQualifiedName indicates the assembly-qualified name of the type to get. See AssemblyQualifiedName. If the type is in the currently executing assembly or in Mscorlib.dll, it is sufficient to supply the type name qualified by its namespace. (Sample: "CreateObject(`System.Windows.Forms.TabPage, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089`){Text = `Page`;UseVisualStyleBackColor = True}")*

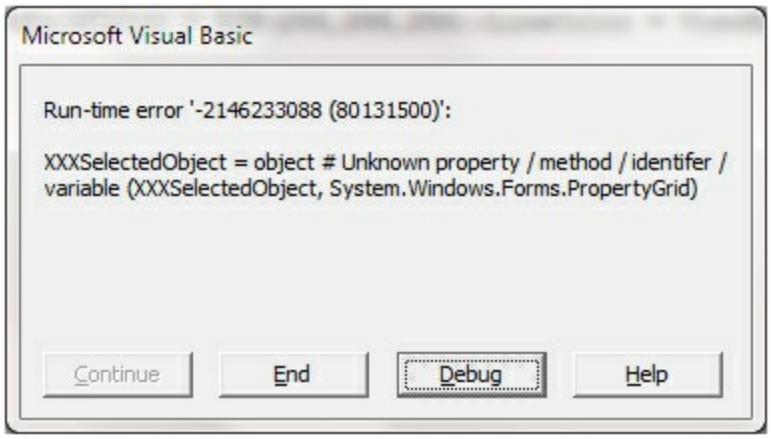
property NAVHostObject.TemplateError as Long

Indicates the error code of the last Template call.

Type	Description
Long	A long expression that describes the error/exception in the Item/Template call.

By default, the TemplateError property returns 0. The TemplateError / TemplateException property indicates the error/exception that occurred in the Item/Template call. The TemplateThrowError property specifies whether the control fires an exception/error when the Template call fails. The TemplateError / TemplateException gets the error if the Template calls fails.

By default, the control fires an exception/error when the Item/Template call fails like shown in the following screen shot:



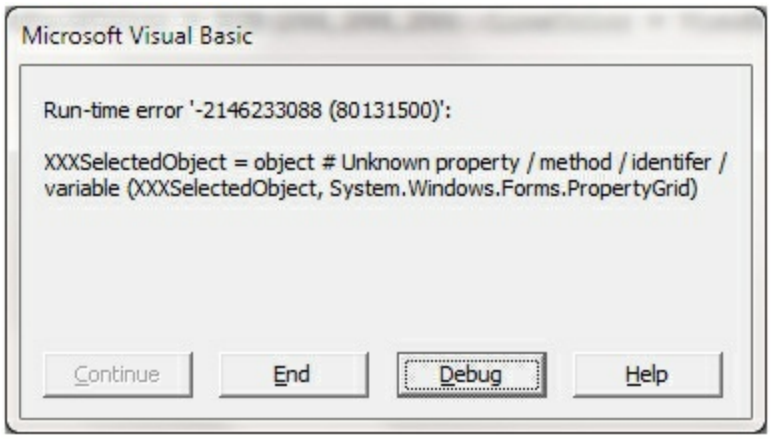
property NAVHostObject.TemplateException as String

Indicates the detailed information about the exception that occurs.

Type	Description
String	A String expression that describes the error/exception in the Item/Template call.

By default, the TemplateException property is empty. The TemplateError / TemplateException property indicates the error/exception that occurred in the Item/Template call. The TemplateThrowError property specifies whether the control fires an exception/error when the Template call fails. The TemplateError / TemplateException gets the error if the Template calls fails.

By default, the control fires an exception/error when the Item/Template call fails like shown in the following screen shot:



property NAVHostObject.TemplateResult as NAVObjectTemplate

Indicates the result of the last Template call.

Type	Description
NAVObjectTemplate	A NAVObjectTemplate object that holds the result of the last Template call.

The TemplateResult property returns the result of the last instruction into a Template call, as a NAVObjectTemplate object. Use the Template/Item property to get/set properties / fields / parameters, invoke methods of the hosting /NET framework Value, using the x-script code. The Item property does exactly the same thing as Template call, excepts that it returns the TemplateResult property. For instance, using the Template/Item property you can change the hosting control's background color, add nodes, and so on. The TemplateThrowError property specifies whether the control fires an exception/error when the Template call fails. The TemplateError / TemplateException gets the error if the Template calls fails.

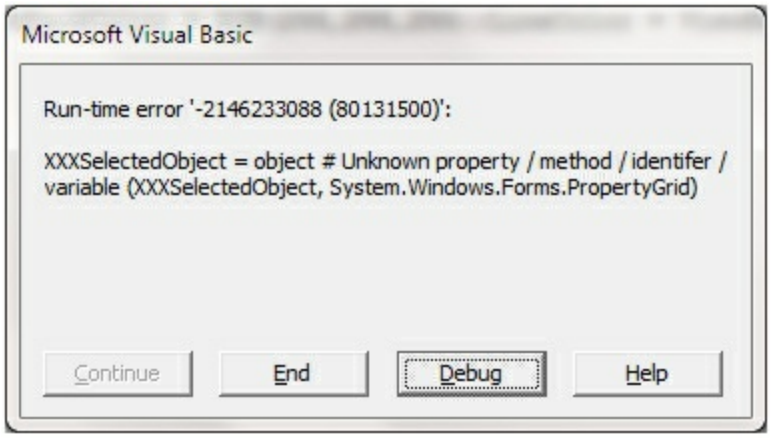
property NAVHostObject.TemplateThrowError as Boolean

Specifies whether the execution of the template stops once an error occurs.

Type	Description
Boolean	A Boolean expression that specifies whether the NAVHost control fires an error/exception when an error occurs in the Item/Template call.

By default, the TemplateThrowError property is False. The TemplateThrowError property specifies whether the control fires an exception/error when the Template call fails. The TemplateError / TemplateException property indicates the error/exception that occurred in the Item/Template call. The TemplateError / TemplateException gets the error if the Template calls fails.

By default, the control fires an exception/error when the Item/Template call fails like shown in the following screen shot:



property NAVHostObject.Type as String

Indicates the type of the object's value.

Type	Description
String	A string that specifies the fully assembly-qualified name of the type, which includes the name of the assembly from which this Type object is loaded. For instance: "System.Windows.Forms.TreeView, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"

The Type property returns a string that specifies the fully assembly-qualified name of the type, which includes the name of the assembly from which this Type object is loaded. The VtType property indicates the VARIANT type of the object that the current NAVHostObject object holds. If the NAVHostObject holds a class or an object/IDispatch/IUnknown that supports properties, fields, members, any of these can be called through the NAVHostObject properties like: Item, SetTemplateDef or Template property. The Value property holds the original object.

You can use the following properties to convert the current Value to indicated standard types:

- AsBoolean, converts the value to a boolean expression.
- AsDate, converts the value to a DATE-TIME/double expression.
- AsDouble, converts the value to a double expression.
- AsInt, converts the value to an integer-32 expression.
- AsString, gets the value converted to a string expression.

property NAVHostObject.Value as Variant

Specifies the value of the object.

Type	Description
Variant	A VARIANT expression that specifies the original object (.NET Framework object) hold by the current NAVHostObject object.

The Value property holds the original object. Use the SetValue property to change the object being hosted by the current NAVHostObject object. The VtType property indicates the VARIANT type of the object that the current NAVHostObject object holds. The Type property returns a string that specifies the fully assembly-qualified name of the type, which includes the name of the assembly from which this Type object is loaded. If the NAVHostObject holds a class or an object/IDispatch/IUnknown that supports properties, fields, members, any of these can be called through the NAVHostObject properties like: Item, SetTemplateDef or Template property.

You can use the following properties to convert the current Value to indicated standard types:

- AsBoolean, converts the value to a boolean expression.
- AsDate, converts the value to a DATE-TIME/double expression.
- AsDouble, converts the value to a double expression.
- AsInt, converts the value to an integer-32 expression.
- AsString, gets the value converted to a string expression.

property NAVHostObject.VtType as NAVHostVarEnum

Indicates the type/vartype of the object's value.

Type	Description
NAVHostVarEnum	A NAVHostVarEnum expression that specifies the VARIANT-type of the Value.

The VtType property indicates the VARIANT type of the object that the current NAVHostObject object holds. The Type property returns a string that specifies the fully assembly-qualified name of the type, which includes the name of the assembly from which this Type object is loaded. If the NAVHostObject holds a class or an object/IDispatch/IUnknown that supports properties, fields, members, any of these can be called through the NAVHostObject properties like: Item, SetTemplateDef or Template property. The Value property holds the original object.

You can use the following properties to convert the current Value to indicated standard types:

- AsBoolean, converts the value to a boolean expression.
- AsDate, converts the value to a DATE-TIME/double expression.
- AsDouble, converts the value to a double expression.
- AsInt, converts the value to an integer-32 expression.
- AsString, gets the value converted to a string expression.

NAVObjectTemplate object

The NAVObjectTemplate object holds a .NET Framework object. The NAVObjectTemplate type supports the following properties and method:

Name	Description
AsBoolean	Returns a boolean value that represents the current object's value.
AsDate	Returns a date value that represents the current object's value.
AsDouble	Returns a numeric value that represents the current object's value.
AsInt	Returns an integer value that represents the current object's value.
AsString	Returns a string that represents the current object's value.
Item	Executes the template and returns the result.
SetTemplateDef	Defines inside variables for the next Template/ExecuteTemplate call.
SetValue	Specifies the value of the object.
Template	Executes the x-script code.
TemplateError	Indicates the error code of the last Template call.
TemplateException	Indicates the detailed information about the exception that occurs.
TemplateResult	Indicates the result of the last Template call.
TemplateThrowError	Specifies whether the execution of the template stops once an error occurs.
Type	Indicates the type of the object's value.
Value	Specifies the value of the object.
VtType	Indicates the type/vartype of the object's value.

property NAVObjectTemplate.AsBoolean as Boolean

Returns a boolean value that represents the current object's value.

Type	Description
Boolean	A Boolean expression that specifies the Value converted as boolean.

The AsBoolean property converts the Value to a Boolean expression. If the conversion is not possible, the AsBoolean property returns False. The Value property holds the original object. The VtType property indicates the VARIANT type of the object that the current NAVObjectTemplate object holds. The Type property returns a string that specifies the fully assembly-qualified name of the type, which includes the name of the assembly from which this Type object is loaded. If the NAVObjectTemplate holds a class or an object/IDispatch/IUnknown that supports properties, fields, members, any of these can be called through the NAVObjectTemplate properties like: Item, SetTemplateDef or Template property.

You can use the following properties to convert the current Value to indicated standard types:

- AsBoolean, converts the value to a boolean expression.
- AsDate, converts the value to a DATE-TIME/double expression.
- AsDouble, converts the value to a double expression.
- AsInt, converts the value to an integer-32 expression.
- AsString, gets the value converted to a string expression.

property NAVObjectTemplate.AsDate as Date

Returns a date value that represents the current object's value.

Type	Description
Date	A Date/Double expression that specifies the Value converted as date-time.

The AsDate property converts the Value to a DATE-TIME expression. If the conversion is not possible, the AsDate property returns 0. The Value property holds the original object. The VtType property indicates the VARIANT type of the object that the current NAVObjectTemplate object holds. The Type property returns a string that specifies the fully assembly-qualified name of the type, which includes the name of the assembly from which this Type object is loaded. If the NAVObjectTemplate holds a class or an object/IDispatch/IUnknown that supports properties, fields, members, any of these can be called through the NAVObjectTemplate properties like: Item, SetTemplateDef or Template property.

You can use the following properties to convert the current Value to indicated standard types:

- AsBoolean, converts the value to a boolean expression.
- AsDate, converts the value to a DATE-TIME/double expression.
- AsDouble, converts the value to a double expression.
- AsInt, converts the value to an integer-32 expression.
- AsString, gets the value converted to a string expression.

property NAVObjectTemplate.AsDouble as Double

Returns a numeric value that represents the current object's value.

Type	Description
Double	A Double expression that specifies the Value converted as double.

The AsDouble property converts the Value to a double expression. If the conversion is not possible, the AsDouble property returns 0. The Value property holds the original object. The VtType property indicates the VARIANT type of the object that the current NAVObjectTemplate object holds. The Type property returns a string that specifies the fully assembly-qualified name of the type, which includes the name of the assembly from which this Type object is loaded. If the NAVObjectTemplate holds a class or an object/IDispatch/IUnknown that supports properties, fields, members, any of these can be called through the NAVObjectTemplate properties like: Item, SetTemplateDef or Template property.

You can use the following properties to convert the current Value to indicated standard types:

- AsBoolean, converts the value to a boolean expression.
- AsDate, converts the value to a DATE-TIME/double expression.
- AsDouble, converts the value to a double expression.
- AsInt, converts the value to an integer-32 expression.
- AsString, gets the value converted to a string expression.

property NAVObjectTemplate.AsInt as Long

Returns an integer value that represents the current object's value.

Type	Description
Long	A Long expression that specifies the Value converted as long (32-bit integer).

The AsInt property converts the Value to a long expression. If the conversion is not possible, the AsInt property returns 0. The Value property holds the original object. The VtType property indicates the VARIANT type of the object that the current NAVObjectTemplate object holds. The Type property returns a string that specifies the fully assembly-qualified name of the type, which includes the name of the assembly from which this Type object is loaded. If the NAVObjectTemplate holds a class or an object/IDispatch/IUnknown that supports properties, fields, members, any of these can be called through the NAVObjectTemplate properties like: Item, SetTemplateDef or Template property.

You can use the following properties to convert the current Value to indicated standard types:

- AsBoolean, converts the value to a boolean expression.
- AsDate, converts the value to a DATE-TIME/double expression.
- AsDouble, converts the value to a double expression.
- AsInt, converts the value to an integer-32 expression.
- AsString, gets the value converted to a string expression.

property NAVObjectTemplate.AsString as String

Returns a string that represents the current object's value.

Type	Description
String	A String expression that specifies the Value converted as string.

The AsString property converts the Value to a string expression. If the conversion is not possible, the AsString property returns "" (empty). The Value property holds the original object. The VtType property indicates the VARIANT type of the object that the current NAVObjectTemplate object holds. The Type property returns a string that specifies the fully assembly-qualified name of the type, which includes the name of the assembly from which this Type object is loaded. If the NAVObjectTemplate holds a class or an object/IDispatch/IUnknown that supports properties, fields, members, any of these can be called through the NAVObjectTemplate properties like: Item, SetTemplateDef or Template property.

You can use the following properties to convert the current Value to indicated standard types:

- AsBoolean, converts the value to a boolean expression.
- AsDate, converts the value to a DATE-TIME/double expression.
- AsDouble, converts the value to a double expression.
- AsInt, converts the value to an integer-32 expression.
- AsString, gets the value converted to a string expression.

property NAVObjectTemplate.Item (Template as String) as NAVObjectTemplate

Executes the template and returns the result.

Type	Description
Template as String	A String expression that specifies the x-script/template code to be executed.
NAVObjectTemplate	A NAVObjectTemplate property that holds the result of the last instruction within the Template.

Use the Template/Item property to get/set properties / fields / parameters, invoke methods of the hosting /NET framework Value, using the x-script code. The Item property does exactly the same thing as Template call, excepts that it returns the TemplateResult property. For instance, using the Template/Item property you can change the hosting control's background color, add nodes, and so on. Prior to Template/Item call, you can invoke the SetTemplateDef to define values from your code to Template's code (TemplateDef variables).

The Template/ x-script code is a simple way of calling hosting control/object's properties, methods/ events using strings. Exontrol owns the x-script implementation in its easiest way and it does not require any VB engine to get executed. Our simple rule is using the component alone without any other dependency than the Windows system.

The **Template/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>
<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>
```

where:

<identifier> indicates an identifier of the variable, property or method, and should start with a letter.
<type> indicates the type the CreateObject function creates, as the assembly-qualified name of the type to create.
<text> any string of characters

The Template / x-script is composed by lines of instructions. Instructions are separated by "\r\n" (new line characters) or ";" character. The TemplateThrowError property specifies whether the control fires an exception/error when the Template call fails. The TemplateError / TemplateException gets the error if the Template calls fails. The TemplateResult property returns the result of the last instruction into a Template call, as a NAVObjectTemplate object.

An x-script instruction/line can be one of the following:

- **Dim variable**[, variable, ...] *declares the variables in the context. Multiple variables are separated by commas. The SetTemplateDef method can declare new variables to be available for the main context. (Sample: Dim h, h1, h2)*
- **variable =** [object.][property/method(arguments).]**property/method(arguments)** *assigns the result of the **property/method** call to the **variable**. (Sample: h = Nodes.Add(`Node`))*
- [object.][property/method(arguments).]**property(arguments) = value** *assigns the **value** to the **property.** (Sample: Nodes.Add(`Node`).BackColor = RGB(255,0,0))*
- [object.][property/method(arguments).]**property/method(arguments)** *invokes the **property/method**. (Sample: Nodes.Add(`Node`))*

- **{**context **}** *delimits the object's context. The properties/fields or methods called between { and } are related to the last object returned by the property/method prior to { declaration. (Sample: Nodes{Add(`Child 1`);Add(`Child 2`)})*
- ***.** delimits the object than its property or method. (Sample: Nodes.Add(`Element`), or Nodes.Add(`Element`) and Nodes{Add(`Element`)} are equivalents)*

where

- ***variable** is the name of a variable declared with Dim command or previously defined using the SetTemplateDef method.*
- ***property** is the name of a property/field of the current object in the current context.*
- ***method** is the name of a method of the current object in the current context.*
- ***arguments** include constants and/or variables and/or property/method calls separated by comma character.*
- ***object** can be a variable of an Object type, **Me** or **CreateObject** call.*

The x-script uses constant expressions as follows:

- *boolean* expression with possible values as ***True*** or ***False**. The True value is equivalent with -1, while False with 0. (Sample: Visible = 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. (Sample: BackColor = 0xFF0000)*
- *date* expression is delimited by **#** character in the format **#mm/dd/yyyy hh:mm:ss#**. *Sample: #31/12/1971# indicates the December 31, 1971 (Sample: FirstVisibleDate = #1/1/2001#)*
- *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"** or **`text`** indicates the string text, while the ' text , specifies the comment text. (Sample: Text = "caption")*

Also , the template or x-script code supports general functions as follows:

- **Me** *property indicates the original object, and it is defined as a predefined variable. (Sample: Me.Nodes.Add(`Root 1`))*
- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the Red Green Blue bytes for the color being specified. (Sample: Nodes.Add(`Root 1`).BackColor = RGB(255,0,0))*
- **LoadPicture(**file**)** *property loads a picture from a file and returns a Picture object required by the picture properties. (Sample: BackgroundImage = LoadPicture(`C:\exontrol\images\auction.gif`)*
- **CreateObject(***assemblyQualifiedName***)** *property creates an instance of the specified*

type using that type's default constructor. The assemblyQualifiedName indicates the assembly-qualified name of the type to get. See AssemblyQualifiedName. If the type is in the currently executing assembly or in Mscorlib.dll, it is sufficient to supply the type name qualified by its namespace. (Sample: "CreateObject(`System.Windows.Forms.TabPage, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089`){Text = `Page`;UseVisualStyleBackColor = True}")

method NAVObjectTemplate.SetTemplateDef (Value as Variant)

Defines inside variables for the next Template/ExecuteTemplate call.

Type	Description
Value as Variant	If calling the first time, A String expression that indicates the DIM command to define the variables that follows, or a VARIANT expression that defines the value of the variable in the order as they were defined.

The SetTemplateDef method was provided to let you use values/objects inside the next Template/Item call. For instance, let's say you have a date field in your form, and once the user fills it, you want a /NET Frameworks MonthCalendar object to select it. In order to do that you have to call a code like:

```
With NAVHost1
   .BackgroundColor = 16777215
   .Create
"C:\Windows\assembly\GAC_MSIL\System.Windows.Forms\2.0.0.0__b77a5c561934e089\Sy

   With .Host
      .SetTemplateDef "Dim x"
      .SetTemplateDef #1/1/2001#
      .Template = "MaxSelectionCount = 1;SelectionStart = x"
   End With
End With
```

This sample defines the variable x to be 1/1/2001 for the Template call, so the SelectionStart will be set on 1/1/2001.

The call of SetTemplateDef method consists in:

- **First** call of SetTemplateDef method should be on a form of SetTemplateDef("Dim variable[,variable,...]"). This defines the name of the variable that follow to be defined.
- **Next** calls, must be exactly the same as with the number of variables you defined, which will define the variable one by one. For instance, if your first call was SetTemplateDef("Dim h1,h2,h3"), it means that the next-three calls of SetTemplateDef defines the variable h1, h2 and h3

Once you defined the variables, they will be available for the next calls of Template/Item properties.

method NAVObjectTemplate.SetValue (Value as Variant)

Specifies the value of the object.

Type	Description
Value as Variant	A VARIANT expression that specifies the new value to be assigned to the NAVObjectTemplate object.

Use the SetValue property to change the object being hosted by the current NAVObjectTemplate object. The Value property holds the original object. The VtType property indicates the VARIANT type of the object that the current NAVObjectTemplate object holds. The Type property returns a string that specifies the fully assembly-qualified name of the type, which includes the name of the assembly from which this Type object is loaded. If the NAVObjectTemplate holds a class or an object/IDispatch/IUnknown that supports properties, fields, members, any of these can be called through the NAVObjectTemplate properties like: Item, SetTemplateDef or Template property.

You can use the following properties to convert the current Value to indicated standard types:

- AsBoolean, converts the value to a boolean expression.
- AsDate, converts the value to a DATE-TIME/double expression.
- AsDouble, converts the value to a double expression.
- AsInt, converts the value to an integer-32 expression.
- AsString, gets the value converted to a string expression.

property NAVObjectTemplate.Template as String

Executes the x-script code.

Type	Description
String	A String expression that specifies the x-script/template code to be executed.

Use the Template/Item property to get/set properties / fields / parameters, invoke methods of the hosting /NET framework Value, using the x-script code. The Item property does exactly the same thing as Template call, excepts that it returns the TemplateResult property. For instance, using the Template/Item property you can change the hosting control's background color, add nodes, and so on. Prior to Template/Item call, you can invoke the SetTemplateDef to define values from your code to Template's code (TemplateDef variables).

The Template/ x-script code is a simple way of calling hosting control/object's properties, methods/ events using strings. Exontrol owns the x-script implementation in its easiest way and it does not require any VB engine to get executed. Our simple rule is using the component alone without any other dependency than the Windows system.

The **Template/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>
<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>
```

where:

<identifier> indicates an identifier of the variable, property or method, and should start with a letter.
<type> indicates the type the CreateObject function creates, as the assembly-qualified name of the type to create.
<text> any string of characters

The Template / x-script is composed by lines of instructions. Instructions are separated by "\r\n" (new line characters) or ";" character. The TemplateThrowError property specifies whether the control fires an exception/error when the Template call fails. The TemplateError / TemplateException gets the error if the Template calls fails. The TemplateResult property returns the result of the last instruction into a Template call, as a NAVObjectTemplate object.

An x-script instruction/line can be one of the following:

- **Dim variable**[, variable, ...] *declares the variables in the context. Multiple variables are separated by commas. The SetTemplateDef method can declare new variables to be available for the main context. (Sample: Dim h, h1, h2)*
- **variable =** [object.][property/method(arguments).]**property/method(arguments)** *assigns the result of the **property/method** call to the **variable**. (Sample: h = Nodes.Add(`Node`))*
- [object.][property/method(arguments).]**property(arguments) = value** *assigns the **value** to the **property.** (Sample: Nodes.Add(`Node`).BackColor = RGB(255,0,0))*
- [object.][property/method(arguments).]**property/method(arguments)** *invokes the **property/method**. (Sample: Nodes.Add(`Node`))*
- **{**context **}** *delimits the object's context. The properties/fields or methods called between { and } are related to the last object returned by the property/method prior to { declaration. (Sample: Nodes{Add(`Child 1`);Add(`Child 2`)})*

- *. delimits the object than its property or method. (Sample: Nodes.Add(`Element`), or Nodes.Add(`Element`) and Nodes{Add(`Element`)} are equivalents)*

where

- ***variable** is the name of a variable declared with Dim command or previously defined using the SetTemplateDef method.*
- ***property** is the name of a property/field of the current object in the current context.*
- ***method** is the name of a method of the current object in the current context.*
- ***arguments** include constants and/or variables and/or property/method calls separated by comma character.*
- ***object** can be a variable of an Object type, **Me** or **CreateObject** call.*

The x-script uses constant expressions as follows:

- *boolean* expression with possible values as ***True*** or ***False***. *The True value is equivalent with -1, while False with 0. (Sample: Visible = 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. (Sample: BackColor = 0xFF0000)*
- *date* expression is delimited by **#** character in the format **#mm/dd/yyyy hh:mm:ss#**. *For instance, #31/12/1971# indicates the December 31, 1971 (Sample: Chart.FirstVisibleDate = #1/1/2001#)*
- *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"** or `**text**` indicates the string text, while the ' text , specifies the comment text. (Sample: Text = "caption")*

Also , the template or x-script code supports general functions as follows:

- **Me** *property indicates the original object, and it is defined as a predefined variable. (Sample: Me.Nodes.Add(`Root 1`))*
- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the Red Green Blue bytes for the color being specified. (Sample: Nodes.Add(`Root 1`).BackColor = RGB(255,0,0))*
- **LoadPicture(**file**)** *property loads a picture from a file and returns a Picture object required by the picture properties. (Sample: BackgroundImage = LoadPicture(`C:\exontrol\images\auction.gif`)*
- **CreateObject(***assemblyQualifiedName***)** *property creates an instance of the specified type using that type's default constructor. The assemblyQualifiedName indicates the assembly-qualified name of the type to get. See AssemblyQualifiedName. If the type is in the currently executing assembly or in Mscorlib.dll, it is sufficient to supply the*

type name qualified by its namespace. (Sample: "CreateObject(`System.Windows.Forms.TabPage, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089`){Text = `Page`;UseVisualStyleBackColor = True}")

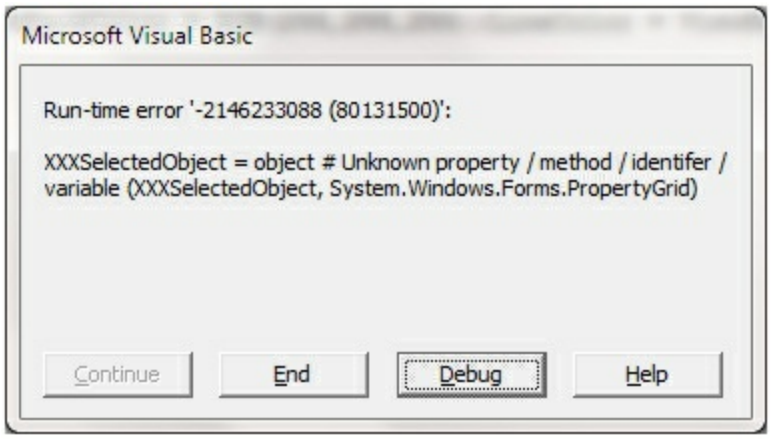
property NAVObjectTemplate.TemplateError as Long

Indicates the error code of the last Template call.

Type	Description
Long	A long expression that describes the error/exception in the Item/Template call.

By default, the TemplateError property returns 0. The TemplateError / TemplateException property indicates the error/exception that occurred in the Item/Template call. The TemplateThrowError property specifies whether the control fires an exception/error when the Template call fails. The TemplateError / TemplateException gets the error if the Template calls fails.

By default, the control fires an exception/error when the Item/Template call fails like shown in the following screen shot:



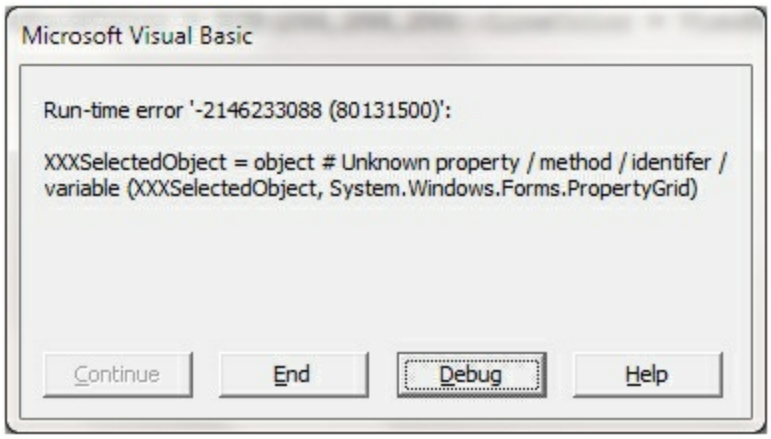
property NAVObjectTemplate.TemplateException as String

Indicates the detailed information about the exception that occurs.

Type	Description
String	A String expression that describes the error/exception in the Item/Template call.

By default, the TemplateException property is empty. The TemplateError / TemplateException property indicates the error/exception that occurred in the Item/Template call. The TemplateThrowError property specifies whether the control fires an exception/error when the Template call fails. The TemplateError / TemplateException gets the error if the Template calls fails.

By default, the control fires an exception/error when the Item/Template call fails like shown in the following screen shot:



property NAVObjectTemplate.TemplateResult as NAVObjectTemplate

Indicates the result of the last Template call.

Type	Description
NAVObjectTemplate	A NAVObjectTemplate object that holds the result of the last Template call.

The TemplateResult property returns the result of the last instruction into a Template call, as a NAVObjectTemplate object. Use the Template/Item property to get/set properties / fields / parameters, invoke methods of the hosting /NET framework Value, using the x-script code. The Item property does exactly the same thing as Template call, excepts that it returns the TemplateResult property. For instance, using the Template/Item property you can change the hosting control's background color, add nodes, and so on. The TemplateThrowError property specifies whether the control fires an exception/error when the Template call fails. The TemplateError / TemplateException gets the error if the Template calls fails.

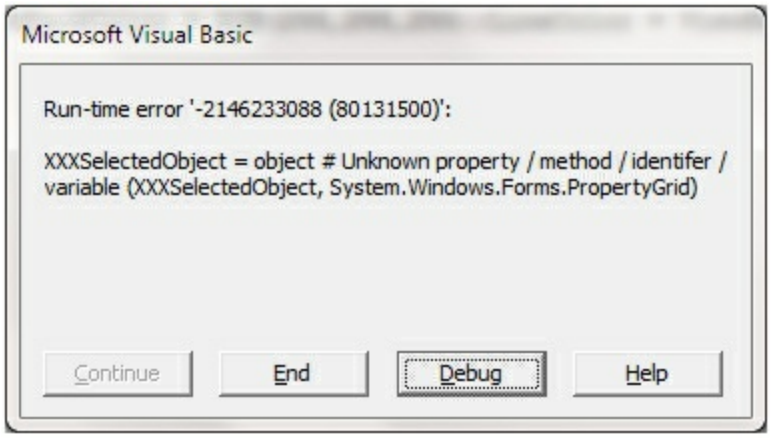
property NAVObjectTemplate.TemplateThrowError as Boolean

Specifies whether the execution of the template stops once an error occurs.

Type	Description
Boolean	A Boolean expression that specifies whether the NAVHost control fires an error/exception when an error occurs in the Item/Template call.

By default, the TemplateThrowError property is True. The TemplateThrowError property specifies whether the control fires an exception/error when the Template call fails. The TemplateError / TemplateException property indicates the error/exception that occurred in the Item/Template call. The TemplateError / TemplateException gets the error if the Template calls fails.

By default, the control fires an exception/error when the Item/Template call fails like shown in the following screen shot:



property NAVObjectTemplate.Type as String

Indicates the type of the object's value.

Type	Description
String	A string that specifies the fully assembly-qualified name of the type, which includes the name of the assembly from which this Type object is loaded. For instance: "System.Windows.Forms.TreeView, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"

property indicates the VARIANT type of the object that the current NAVObjectTemplate object holds. If the NAVObjectTemplate holds a class or an object/IDispatch/IUnknown that supports properties, fields, members, any of these can be called through the NAVObjectTemplate properties like: Item, SetTemplateDef or Template property. The Value property holds the original object.

You can use the following properties to convert the current Value to indicated standard types:

- AsBoolean, converts the value to a boolean expression.
- AsDate, converts the value to a DATE-TIME/double expression.
- AsDouble, converts the value to a double expression.
- AsInt, converts the value to an integer-32 expression.
- AsString, gets the value converted to a string expression.

property NAVObjectTemplate.Value as Variant

Specifies the value of the object.

Type	Description
Variant	A VARIANT expression that specifies the original object (.NET Framework object) hold by the current NAVObjectTemplate object.

The Value property holds the original object. Use the SetValue property to change the object being hosted by the current NAVHostObject object. The VtType property indicates the VARIANT type of the object that the current NAVObjectTemplate object holds. The Type property returns a string that specifies the fully assembly-qualified name of the type, which includes the name of the assembly from which this Type object is loaded. If the NAVObjectTemplate holds a class or an object/IDispatch/IUnknown that supports properties, fields, members, any of these can be called through the NAVObjectTemplate properties like: Item, SetTemplateDef or Template property.

You can use the following properties to convert the current Value to indicated standard types:

- AsBoolean, converts the value to a boolean expression.
- AsDate, converts the value to a DATE-TIME/double expression.
- AsDouble, converts the value to a double expression.
- AsInt, converts the value to an integer-32 expression.
- AsString, gets the value converted to a string expression.

property NAVObjectTemplate.VtType as NAVHostVarEnum

Indicates the type/vartype of the object's value.

Type	Description
NAVHostVarEnum	A NAVHostVarEnum expression that specifies the VARIANT-type of the Value.

The VtType property indicates the VARIANT type of the object that the current NAVObjectTemplate object holds. The Type property returns a string that specifies the fully assembly-qualified name of the type, which includes the name of the assembly from which this Type object is loaded. If the NAVObjectTemplate holds a class or an object/IDispatch/IUnknown that supports properties, fields, members, any of these can be called through the NAVObjectTemplate properties like: Item, SetTemplateDef or Template property. The Value property holds the original object.

You can use the following properties to convert the current Value to indicated standard types:

- AsBoolean, converts the value to a boolean expression.
- AsDate, converts the value to a DATE-TIME/double expression.
- AsDouble, converts the value to a double expression.
- AsInt, converts the value to an integer-32 expression.
- AsString, gets the value converted to a string expression.