



Enterprise Architect

User Guide Series

Enterprise Architect Add-In Model

How do I create Add-Ins to extend the Enterprise Architect User Interface? Use the Add-In model to enhance the user interface by adding new menus, windows and controls to perform a variety of functions.

Author: Sparx Systems

Date: 2021-09-02

Version: 15.2

CREATED WITH  ENTERPRISE
ARCHITECT

Table of Contents

Enterprise Architect Add-In Model	8
The Add-In Manager	11
Add-In Tasks	13
Create Add-Ins	15
Define Menu Items	17
Deploy Add-Ins	21
Tricks and Traps	25
Add-In Search	30
EA_SampleSearch	32
XML Format (Search Data)	34
Add-In Events	37
EA_OnAddinPropertiesTabChanging	39
EA_Connect	41
EA_Disconnect	43
EA_GetMenuItems	44
EA_GetMenuState	47
EA_GetRibbonCategory	50
EA_MenuClick	52
EA_OnOutputItemClicked	55
EA_OnOutputItemDoubleClicked	58
EA_ShowHelp	61
Broadcast Events	63
Add-In License Management Events	66

EA_AddinLicenseValidate	67
EA_AddinLicenseGetDescription	69
EA_GetSharedAddinName	71
Custom Table Events	74
EA_OnCustomTableBeginEdit	76
EA_OnCustomTableEndEdit	78
EA_OnCustomTableSelectionChanged	80
EA_OnCustomTableCellUpdated	82
Schema Composer Events	84
EA_GenerateFromSchema	85
EA_GetProfileInfo	87
EA_IsSchemaExporter	89
Compartment Events	91
EA_QueryAvailableCompartments	92
EA_GetCompartmentData	95
Context Item Events	100
EA_OnContextItemChanged	101
EA_OnContextItemDoubleClicked	104
EA_OnNotifyContextItemModified	107
EA_FileClose	109
EA_FileNew	111
EA_FileOpen	113
EA_OnPostCloseDiagram	115
EA_OnPostInitialized	117
EA_OnPostOpenDiagram	119
EA_OnPostTransform	121
EA_OnPreExitInstance	123

EA_OnRetrieveModelTemplate	124
EA_OnTabChanged	127
EA_LoadWindowManager	129
Model Validation Events	132
EA_OnInitializeUserRules	134
EA_OnStartValidation	136
EA_OnEndValidation	138
EA_OnRunElementRule	140
EA_OnRunPackageRule	142
EA_OnRunDiagramRule	144
EA_OnRunConnectorRule	146
EA_OnRunAttributeRule	148
EA_OnRunMethodRule	150
EA_OnRunParameterRule	152
Model Validation Example	155
Post-New Events	166
EA_OnPostNewElement	168
EA_OnPostNewConnector	170
EA_OnPostNewDiagram	172
EA_OnPostNewDiagramObject	174
EA_OnPostNewAttribute	176
EA_OnPostNewMethod	178
EA_OnPostNewPackage	180
EA_OnPostNewGlossaryTerm	182
Pre-Deletion Events	184
EA_OnPreDeleteElement	186
EA_OnPreDeleteAttribute	188

EA_OnPreDeleteMethod	190
EA_OnPreDeleteConnector	192
EA_OnPreDeleteDiagram	194
EA_OnPreDeleteDiagramObject	196
EA_OnPreDeletePackage	198
EA_OnPreDeleteGlossaryTerm	200
Pre New-Object Events	202
EA_OnPreNewElement	204
EA_OnPreNewConnector	206
EA_OnPreNewDiagram	209
EA_OnPreNewDiagramObject	211
EA_OnPreDropFromTree	214
EA_OnPreNewAttribute	216
EA_OnPreNewMethod	218
EA_OnPreNewPackage	220
EA_OnPreNewGlossaryTerm	222
Tagged Value Events	224
EA_OnAttributeTagEdit	225
EA_OnConnectorTagEdit	227
EA_OnElementTagEdit	229
EA_OnMethodTagEdit	231
Technology Events	233
EA_OnInitializeTechnologies	234
EA_OnPreActivateTechnology	236
EA_OnPostActivateTechnology	238
EA_OnPreDeleteTechnology	240
EA_OnDeleteTechnology	243

EA_OnImportTechnology	246
Custom Views	249
Create a Custom View	250
Custom Docked Window	252
MDG Add-Ins	255
MDG Events	256
MDG_BuildProject	258
MDG_Connect	260
MDG_Disconnect	263
MDG_GetConnectedPackages	265
MDG_GetProperty	267
MDG_Merge	270
MDG_NewClass	276
MDG_PostGenerate	278
MDG_PostMerge	281
MDG_PreGenerate	283
MDG_PreMerge	285
MDG_PreReverse	287
MDG_RunExe	289
MDG_View	291
Workflow Add-In Events	294
EA_AllowPropertyUpdate	296
EA_AllowTagUpdate	298
EA_CanEditProperty	300
EA_CanEditTag	302

Enterprise Architect Add-In Model



The **Add-In** facility provides a means of extending Enterprise Architect, allowing the programmer to enhance the user interface by adding new menus, sub menus, windows and other controls to perform a variety of functions. An Add-In is an ActiveX COM object that is notified of events in the user interface, such as mouse clicks and element selections, and has access to the repository content through the **Object Model**. **Add-Ins** can also be integrated with the license management system.

Using this powerful facility, you can extend Enterprise Architect to create new features not available in the core product, and these can be compiled and easily distributed to a community of users within an organization, or more broadly to an entire industry. Using the Add-In facility it is even possible to create support for modeling languages and frameworks not supported in the core product.

Add-Ins have several advantages over stand-alone automation clients:

- Add-Ins can (and should) be written as in-process (DLL) components; this provides lower call overhead and better integration into the Enterprise Architect environment
- Because a current version of Enterprise Architect is

already running there is no requirement to start a second copy of Enterprise Architect via the automation interface

- Because the Add-In receives object handles associated with the currently running copy of Enterprise Architect, more information is available about the current user's activity; for example, which diagram objects are selected
- You are not required to do anything other than to install the Add-In to make it usable; that is, you do not have to configure Add-Ins to run on your systems
- Because Enterprise Architect is constantly evolving in response to customer requests, the Add-In interface is flexible
- The Add-In interface does not have its own version, rather it is identified by the version of Enterprise Architect it first appeared in; for example, the current version of the Enterprise Architect Add-In interface is version 2.1
- When creating your Add-In, you do not have to subscribe to a type-library (Add-Ins created before 2004 are no longer supported - if an Add-In subscribes to the `Addn_Tmpl.tlb` interface (2003 style), it fails on load; in this event, contact the vendor or author of the Add-In and request an upgrade)
- Add-Ins do not have to implement methods that they never use
- Add-Ins prompt users via context menus in the tree view and the diagram
- Menu check and disable states can be controlled by the Add-In

Add-Ins enhance the existing functionality of Enterprise Architect through a variety of mechanisms, such as Scripts, UML Profiles and the **Automation Interface**. Once an Add-In is registered, it can be managed using the Add-In Manager.

The Add-In Manager

If you want to check what **Add-Ins** are available on your system, and enable or disable them for use, you can review the '**Add-In Manager**' dialog. This dialog lists the Add-Ins that have been registered on your system, and their current status (Enabled or Disabled).

Access

Ribbon	Specialize > Add-Ins > Manage-Addin
--------	--

Enable/Disable Add-Ins

Action	Detail
Enable an Add-In	<p>To enable an Add-In so that it is available for use, select the 'Load on Startup' checkbox corresponding to the name.</p> <p>Click on the OK button.</p> <ul style="list-style-type: none">Any Add-In specific features, facilities and Help are made available through

	<p>the 'Specialize <add-in name>' context menu option</p> <ul style="list-style-type: none">Any defined Add-In windows are populated with information; select the 'Specialize > Add-Ins > Windows' menu option
Disable an Add-In	<p>To disable an Add-In so that it is not available for use, clear the 'Load on Startup' checkbox corresponding to the name.</p> <p>Click on the OK button.</p> <p>All menu options, features and facilities specific to the Add-In are hidden and made inactive.</p>

Notes

- When you enable or disable an **Add-In**, you must re-start Enterprise Architect to action the change

Add-In Tasks

This topic directs you to information on creating, testing, deploying and managing **Add-Ins**.

Create an Add-In

Task	Information
Create the Add-In.	See the <i>Create Add-Ins</i> topic.
Define Menu Items.	See the <i>Define Menu Items</i> topic.
Respond to Menu Events.	See the <i>EA_MenuClick</i> topic.
Handle Add-In Events.	See the <i>Add-In Events</i> topic.

Deploy your Add-In

Consideration	Information
Tricks and Traps	See the <i>Tricks and Traps</i> topic.

Manage Add-Ins

Task	Information
Register an Add-In (developed in-house or brought-in).	Brought-in applications are referred to as Commercial Off The Shelf (COTS) software. See the <i>Register Add-In</i> topic.
The Add-In Manager.	See <i>The Add-In Manager</i> topic.

Create Add-Ins

Before you start you must have an application development tool that is capable of creating ActiveX COM objects supporting the IDispatch interface, such as:

- Embarcadero Delphi, or Borland Delphi
- Microsoft Visual Basic
- Microsoft Visual Studio .NET

You should consider how to define menu items. To help with this, you could review some examples of Automation Interfaces - examples of code used to create **Add-Ins** for Enterprise Architect - on the Sparx Systems web page.

Create an Enterprise Architect Add-In

Step	Action
1	Use a development tool to create an ActiveX COM DLL project. Visual Basic users, for example, choose File>Create New Project-ActiveX DLL.
2	Connect to the interface using the syntax appropriate to the language.

3	Create a COM Class and implement each of the general Add-In Events applicable to your Add-In. You only have to define methods for events to respond to.
4	Add a registry key that identifies your Add-In to Enterprise Architect, as described in the Deploy Add-Ins topic.

Define Menu Items

Tasks

Task	Detail
Define Menu Items	<p>Menu items are defined by responding to the GetMenuItems event.</p> <p>The first time this event is called, MenuName is an empty string, representing the top-level menu. For a simple Add-In with just a single menu option you can return a string.</p> <pre>Function EA_GetMenuItems(Repository as EA.Repository, MenuLocation As String, MenuName As String) As Variant EA_GetMenuItems = "&Joe's Add-In" End Function</pre>
Define Sub-Menus	<p>To define sub-menus, prefix a parent menu with a dash. Parent and sub-items are defined in this way:</p> <pre>Function EA_GetMenuItems(Repository as EA.Repository, MenuLocation As String, MenuName As String) As Variant</pre>

	<pre> Select Case MenuName Case "" 'Parent Menu Item EA_GetMenuItems = "-&Joe's Add-In" Case "-&Joe's Add-In" 'Define Sub-Menu Items using the Array notation. 'In this example, "Diagram" and "Treeview" compose the "Joe's Add-In" sub-menu. EA_GetMenuItems = Array("&Diagram", "&Treeview") Case Else MsgBox "Invalid Menu", vbCritical End Select End Function </pre>
Define Further Sub-Menus	<p>Similarly, you can define further sub-items:</p> <pre> Function EA_GetMenuItems(Repository as EA.Repository, MenuLocation As String, MenuName As String) As Variant Select Case MenuName Case "" EA_GetMenuItems = "-Joe's </pre>

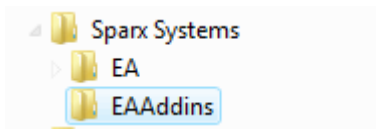
	<pre> Add-In" Case "-Joe's Add-In" EA_GetMenuItems = Array("-&Diagram", "&TreeView") Case "-&Diagram" EA_GetMenuItems = "&Properties" Case Else MsgBox "Invalid Menu", vbCritical End Select End Function </pre>
Enable/Disable menu options	<p>To enable or disable menu options by default, you can use this method to show particular items to the user:</p> <pre> Sub EA_GetMenuState(Repository As EA.Repository, Location As String, MenuName As String, ItemName As String, IsEnabled As Boolean, IsChecked As Boolean) Select Case Location Case "TreeView" 'Always enable Case "Diagram" 'Always enable Case "MainMenu" </pre>

	<pre>Select Case ItemName Case "&Translate", "Save &Project" If GetIsProjectSelected() Then IsEnabled = False End If End Select End Select IsChecked = GetIsCurrentSelection() End Sub</pre>
--	---

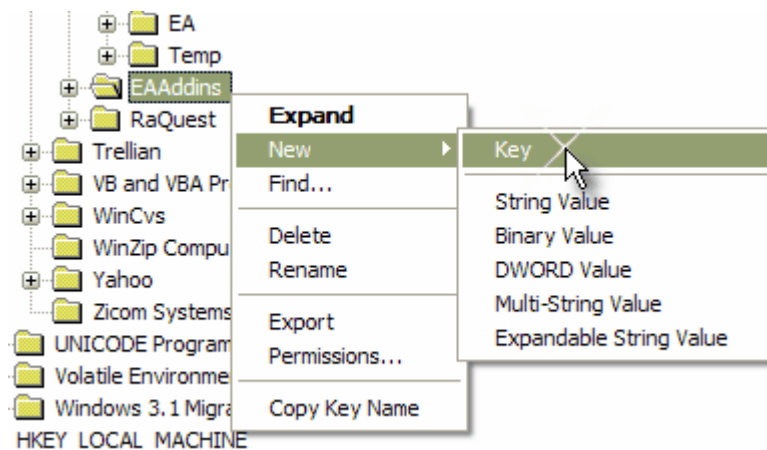
Deploy Add-Ins

Deploy Add-Ins to users' sites

Step	Action
1	<p>Add the Add-In DLL file to an appropriate directory on the user's computer; that is:</p> <p>C:\Program Files\ (new dir)</p>
2	<p>Register the DLL as appropriate to your platform:</p> <ul style="list-style-type: none">• If compiled as a native Win32 DDL, such as VB or C++, register the DDL using the regsvr32 command from the command prompt regsvr32 "C:\Program Files\MyCompany\EAAddin\EAAddin.dll"• If compiled as a .NET DLL, such as C# or VB.NET, register the DLL using the RegAsm command from the command prompt C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\RegAsm.exe "C:\Program

	Files\MyCompany\EAAaddin\EAAaddin.dll" /codebase
3	Place a new entry into the registry using the registry editor (run regedit) so that Enterprise Architect recognizes the presence of your Add-In .
4	<p>Add a new key 'EAAAddIns' under one of these locations:</p> <ul style="list-style-type: none">• For the current user only [HKEY_CURRENT_USER\Software\Sparx Systems]• For multiple users on a machine<ul style="list-style-type: none">- Under 32-bit versions of Windows [HKEY_LOCAL_MACHINE\Software\Sparx Systems]- Under 64-bit versions of Windows [HKEY_LOCAL_MACHINE\Software\Wow6432Node\Sparx Systems] 
5	Add a new key under this key with the

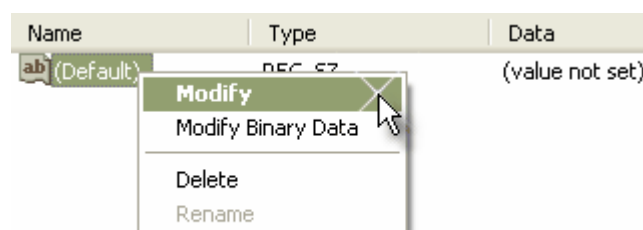
project name.



(ProjectName) is not necessarily the name of your DLL, but the name of the Project; in Visual Basic, this is the value for the property Name corresponding to the project file.

6

Specify the default value by modifying the default value of the key.



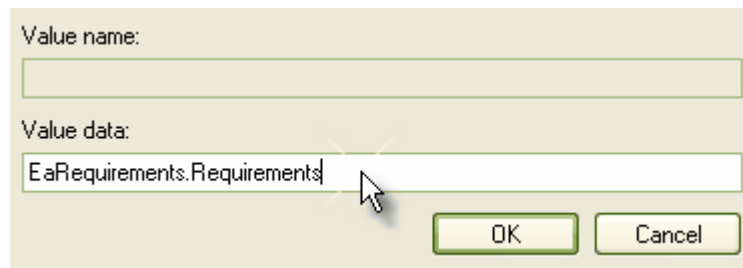
7

Enter the value of the key by typing in the (project name).(class name), such as:

EaRequirements.Requirements

where *EaRequirements* is the project

name, as shown in this example:



The screenshot shows a dialog box with a light beige background. It contains two text input fields. The first field is labeled "Value name:" and is empty. The second field is labeled "Value data:" and contains the text "EaRequirements.Requirements". A mouse cursor is pointing at the end of the text in the second field. At the bottom right of the dialog box are two buttons: "OK" and "Cancel".

Tricks and Traps

Considerations

Item	Detail
Visual Basic 5/6 Users Note	<p>Visual Basic users should note that the version number of the Enterprise Architect interface is stored in the VBP project file in a form similar to this:</p> <p>Reference=*\G{64FB2BF4-9EFA-11D2-8307-C45586000000}#2.2#0#..\..\..\Program Files\Sparx Systems\EA\EA.TLB#Enterprise Architect Object Model 2.02</p> <p>If you experience problems moving from one version of Enterprise Architect to another, open the VBP file in a text editor and remove this line. Then open the project in Visual Basic and use Project-References to create a new reference to the Enterprise Architect Object model.</p>
Holding State Information	<p>It is possible for an Add-In to hold state information, meaning that data can be stored in member variables in response to</p>

one event and retrieved in another. There are some dangers in doing this:

- Enterprise Architect Automation Objects do not update themselves in response to user activity, to activity on other workstations, or even to the actions of other objects in the same automation client; retaining handles to such objects between calls can result in the second event querying objects that have no relationship with the current state of Enterprise Architect
- When you close Enterprise Architect, all **Add-Ins** are asked to shut down; if there are any external automation clients Enterprise Architect must stay active, in which case all the Add-Ins are reloaded, losing all the data
- Enterprise Architect acting as an automation client does not close if an Add-In still holds a reference to it (releasing all references in the Disconnect() event avoids this problem)

It is recommended that unless there is a specific reason for doing so, the Add-In should use the repository parameter and its method and properties to provide the necessary data.

Enterprise Architect Not Closing	<p>.NET Specific Issues</p> <p>Automation checks the use of objects and will not allow any of them to be destroyed until they are no longer being used.</p> <p>As noted in the <i>Automation Interface</i> topic, if your automation controller was written using the .NET framework, Enterprise Architect does not close even after you release all your references to it. To force the release of the COM pointers, call the memory management functions as shown:</p> <pre>GC.Collect(); GC.WaitForPendingFinalizers();</pre> <p>Additionally, because automation clients hook into Enterprise Architect, which creates Add-Ins that in turn hook back into Enterprise Architect, it is possible to get into a deadlock situation where Enterprise Architect and the Add-Ins will not let go of one another and keep each other active. An Add-In might retain hooks into Enterprise Architect because:</p> <ul style="list-style-type: none">• It keeps a private reference to an Enterprise Architect object (see the earlier <i>Holding State Information</i>), or• It has been created by .NET and the
----------------------------------	--

GC mechanism has not yet released it
There are two actions required to avoid deadlock situations:

- Automation controllers must call `Repository.CloseAddins()` at some point (perhaps at the end of processing)
- Add-Ins must release all references to Enterprise Architect in the `Disconnect()` event; see the *Add-In Events* topic for details

It is possible that your Automation client controls a running instance of Enterprise Architect where the Add-Ins have not complied with the rules. In this case you could call `Repository.Exit()` to terminate Enterprise Architect.

Miscellaneous

In developing Add-Ins using the .NET framework you must select COM Interoperability in the project's properties in order for it to be recognized as an Add-In.

Some development environments do not automatically register COM DLLs on creation. You might have to do that manually before Enterprise Architect recognizes the Add-In.

	<p>You can use your private Add-In key (as required for Add-In deployment) to store configuration information pertinent to your Add-In.</p>
--	---

Add-In Search

Enterprise Architect enables Extensions to integrate with the **Model Search**. Searches can be defined that execute a method within your **Add-In** and display your results in an integrated way.

Details

Item
The method that runs the search must be structured in this way.
Defines the XML structure expected by Enterprise Architect to specify search results.
<p>In addition to the displayed results, two additional hidden fields can be passed into the XML that provide special functionality.</p> <ul style="list-style-type: none">• CLASSTYPE - Returning a field of CLASSTYPE, containing the Object_Type value from the t_object table, displays the appropriate icon in the column in which you place the field• CLASSGUID - Returning a field of CLASSGUID, containing an ea_guid value, enables the Model Search to track the object in the Browser window and open

the **Properties window** for the element by
double-clicking in the Model Search

EA_SampleSearch

This defines the signature required for the function Enterprise Architect calls when executing an **Add-In** search. The name can be changed to any valid function name in your target programming language.

Syntax

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the Enterprise Architect model about to be closed. Poll its members to retrieve model data and user interface status information.
SearchText	String Direction: IN Description: Provides the value (if any) entered by the user in the search term field in the model search window.
XMLResults	String

	Direction: OUT Description: Provides the value (if any) entered by the user in the search term field in the model search window.
--	---

Return Value

The method must return any non-empty value for the results to be displayed.

XML Format (Search Data)

This example XML provides the format for the sSearchData parameter of the RunModelSearch method.

```
<ReportViewData UID=\"MySearchID\">
```

```
<!--
```

//The UID attribute enables XML type searches to persist column information. That is, if you run the search, group by column or adjust

//column widths, then close the window and run the search again, the format/organization changes are retained. To avoid persisting column

//arrangements, leave the attribute value blank or remove it altogether. Use this section to declare all possible fields - columns that appear

//in Enterprise Architect's **Search window** - that are used below in <Rows/>. The order of the columns of information to be appended here must

//match the order that the search run in Enterprise Architect would normally display. Furthermore, if you append results onto a custom SQL

//Search, then the order used in your Custom SQL must match the order used here.

```
-->
```

```
<Fields>
```

```
<Field name=\"\"/>
```

```
<Field name=""/>
<Field name=""/>
<Field name=""/>
</Fields>
<Rows>
  <Row>
    <Field name="" value=""/>
    <Field name="" value=""/>
    <Field name="" value=""/>
    <Field name="" value=""/>
  </Row>
  <Row>
    <Field name="" value=""/>
    <Field name="" value=""/>
    <Field name="" value=""/>
    <Field name="" value=""/>
  </Row>
  <Row>
    <Field name="" value=""/>
    <Field name="" value=""/>
    <Field name="" value=""/>
    <Field name="" value=""/>
  </Row>
</Rows>
</ReportViewData>
```


Add-In Events

All Enterprise Architect **Add-Ins** can choose to respond to general **Add-In** events.

Events

Event
<i>EA_Connect</i> - Add-Ins can use this to identify their type and to respond to Enterprise Architect start up.
<i>EA_Disconnect</i> - Add-Ins can use this to respond to user requests to disconnect the model branch from an external project.
<i>EA_GetMenuItems</i> - Add-Ins can use this to provide the Enterprise Architect user interface with additional Add-In menu options in various context menus.
<i>EA_GetMenuState</i> - Add-Ins can use this to set a particular menu option to either enabled or disabled.
<i>EA_GetRibbonCategory</i> - Add-Ins can use this to identify the Ribbon panel in which to house their calling icon.

EA_MenuClick - received by an **Add-In** in response to user selection of a menu option.

EA_OnOutputItemClicked - informs **Add-Ins** that the user has clicked on a list entry in the system tab or one of the user defined output tabs.

EA_OnOutputItemDoubleClicked - informs **Add-Ins** that the user has used the mouse to double-click on a list entry in one of the user-defined output tabs.

EA_ShowHelp - **Add-Ins** can use this to show a Help topic for a particular menu option.

EA_OnAddinPropertiesTabChanging

Indicates that a value in a properties list added via Repository.AddPropertiesTab has been changed by the user.

Syntax

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects describing the field changed: <ul style="list-style-type: none">• TabName: The name of the Add-Ins window tab changing• PropID: Unique ID assign to Property item within the xml definition.

	<ul style="list-style-type: none">• ChangeValue: The value the Property is changing to.• OriginalValue: The original value assigned to the Property
--	--

Return Value

- Return **False** to indicate that this change was rejected
- Return **True** to indicate that the change is accepted

EA_Connect

Add-Ins can use EA_Connect events to identify their type and to respond to Enterprise Architect start up.

This event occurs when Enterprise Architect first loads your **Add-In**. Enterprise Architect itself is loading at this time so that while a Repository object is supplied, there is limited information that you can extract from it.

The chief uses for EA_Connect are in initializing global Add-In data and for identifying the Add-In as an MDG Add-In.

Syntax

Function EA_Connect (Repository As EA.Repository) As String

The EA_Connect function syntax has this parameter:

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Return Value

A string identifying a specialized type of **Add-In**:

Type	Details
"MDG"	MDG Add-Ins receive MDG Events and extra menu options.
"Workflow"	Workflow add-ins receive additional events to control user ability to change specific fields.
""	A non-specialized Add-In .

EA_Disconnect

Add-Ins can use the EA_Disconnect event to respond to user requests to disconnect the model branch from an external project.

This function is called when Enterprise Architect closes. If you have stored references to Enterprise Architect objects (not recommended anyway), you must release them here.

In addition, .NET users must call memory management functions as shown:

```
GC.Collect();
```

```
GC.WaitForPendingFinalizers();
```

Syntax

```
Sub EA_Disconnect()
```

Return Value

None.

EA_GetMenuItems

The EA_GetMenuItems event enables the **Add-In** to provide the Enterprise Architect user interface with additional Add-In menu options in various context menus. When a user selects an Add-In menu option, an event is raised and passed back to the Add-In that originally defined that menu option.

This event is raised just before Enterprise Architect has to show particular menu options to the user, and its use is described in the *Define Menu Items* topic.

Syntax

Function EA_GetMenuItems (Repository As EA.Repository, MenuLocation As String, MenuName As String) As Variant

The EA_GetMenuItems function syntax has these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface

	status information.
MenuLocation	<p>String</p> <p>Direction: IN</p> <p>Description: A string representing the part of the user interface that brought up the context menu. This can be TreeView, MainMenu, Diagram or Other. You can add further values for MenuLocation at any time.</p> <p>A MenuLocation of 'TreeView' would indicate that the menu was displayed in the Browser window; 'MainMenu' would indicate that the menu was displayed from a ribbon option, and 'Diagram' that the menu was displayed within a diagram. 'Other' would indicate an unspecified location, which might be one of these:</p> <ul style="list-style-type: none">• Calendar• Dialog• Element List• Gantt• Model View• Project View• Relationship Matrix• Reviews

	<ul style="list-style-type: none">• Search• Specification Manager
MenuName	String Direction: IN Description: The name of the parent menu for which sub-items are to be defined. In the case of the top-level menu this is an empty string.

Return Value

One of these types:

- A string indicating the label for a single menu option
- An array of strings indicating multiple menu options
- Empty (Visual Basic/VB.NET) or null (C#) to indicate that no menu should be displayed

In the case of the top-level menu it should be a single string or an array containing only one item, or empty/null.

EA_GetMenuState

Add-Ins can use the EA_GetMenuState event to set a particular menu option to either enabled or disabled. This is useful when dealing with locked Packages and other situations where it is convenient to show a menu option, but not enable it for use.

This event is raised just before Enterprise Architect has to show particular menu options to the user. Its use is further described in the *Define Menu Items* topic.

Syntax

Sub EA_GetMenuState (Repository as EA.Repository, MenuLocation As String, MenuName as String, ItemName as String, IsEnabled as Boolean, IsChecked as Boolean)

The EA_GetMenuState function syntax has these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

MenuLocation	<p>String</p> <p>Direction: IN</p> <p>Description: A string representing the part of the user interface that brought up the menu. This can be TreeView, MainMenu or Diagram.</p>
MenuName	<p>String</p> <p>Direction: IN</p> <p>Description: The name of the parent menu for which sub-items must be defined. In the case of the top-level menu it is an empty string.</p>
ItemName	<p>String</p> <p>Direction: IN</p> <p>Description: The name of the option actually clicked; for example, 'Create a New Invoice'.</p>
IsEnabled	<p>Boolean</p> <p>Direction: OUT</p> <p>Description: Set to False to disable this particular menu option.</p>
IsChecked	<p>Boolean</p> <p>Direction: OUT</p>

	Description: Set to True to check this particular menu option.
--	---

Return Value

None.

EA_GetRibbonCategory

Add-Ins can use EA_GetRibbonCategory events to identify the Ribbon in which the **Add-In** should place its menu icon.

This event occurs when Enterprise Architect first loads your Add-In. Enterprise Architect itself is loading at this time so that while a Repository object is supplied, there is limited information that you can extract from it.

The chief use for EA_GetRibbonCategory is in initializing the Add-In access point.

Syntax

Function EA_GetRibbonCategory (Repository As EA.Repository) As String

The EA_GetRibbonCategory function syntax has this parameter:

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Return Value

A string matching the name of the selected ribbon (in English if you are using a translated version). The possible names are:

- Start
- Design
- Layout
- Publish
- Specialize
- Construct
- Code
- Simulate
- Execute
- Manage

It is not possible to include **Add-Ins** in the 'Specification - Specify' ribbon or 'Documentation - Edit' ribbon.

If the function isn't implemented (or if an invalid name is returned) the '**Add-In**' menu will be available from the 'Specialize' ribbon, 'Add-Ins' panel.

EA_MenuClick

EA_MenuClick events are received by an **Add-In** in response to user selection of a menu option.

The event is raised when the user clicks on a particular menu option. When a user clicks on one of your non-parent menu options, your Add-In receives a MenuClick event, defined as:

```
Sub EA_MenuClick(Repository As EA.Repository,  
ByVal MenuLocation As String, ByVal MenuName As  
String, ByVal ItemName As String)
```

This code is an example of use:

```
    If MenuName = "-&Diagram" And ItemName =  
"&Properties" then  
        MsgBox Repository.GetCurrentDiagram.Name,  
vbInformation  
    Else  
        MsgBox "Not Implemented", vbCritical  
    End If
```

Notice that your code can directly access Enterprise Architect data and UI elements using Repository methods.

Syntax

```
Sub EA_MenuClick (Repository As EA.Repository,  
MenuLocation As String, MenuName As String, ItemName
```

As String)

The EA_GetMenuClick function syntax has these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
MenuLocation	String Direction: IN Description: A string representing the part of the user interface that brought up the menu. This can be TreeView, MainMenu or Diagram.
MenuName	String Direction: IN Description: The name of the parent menu for which sub-items are to be defined. In the case of the top-level menu this is an empty string.
ItemName	String

	Direction: IN Description: The name of the option actually clicked; for example, 'Create a New Invoice'.
--	---

Return Value

None.

EA_OnOutputItemClicked

EA_OnOutputItemClicked events inform **Add-Ins** that the user has clicked on a list entry in the system tab or one of the user defined output tabs.

Usually an **Add-In** responds to this event in order to capture activity on an output tab they had previously created through a call to Repository.AddTab().

Note that every loaded Add-In receives this event for every click on an output tab in Enterprise Architect, irrespective of whether the Add-In created that tab. Add-Ins should therefore check the TabName parameter supplied by this event to ensure that they are not responding to other Add-Ins' events.

Syntax

EA_OnOutputItemClicked (Repository As EA.Repository, TabName As String, LineText As String, ID As Long)

The EA_OnOutputItemClicked function syntax has these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise

	Architect model. Poll its members to retrieve model data and user interface status information.
TabName	String Direction: IN Description: The name of the tab that the click occurred in. Usually this would have been created through 'Repository.AddTab()'.
LineText	String Direction: IN Description: The text that had been supplied as the String parameter in the original call to 'Repository.WriteOutput()'.
ID	Long Direction: IN Description: The ID value specified in the original call to Repository.WriteOutput().

Return Value

None.

EA_OnOutputItemDoubleClicked

EA_OnOutputItemDoubleClicked events inform **Add-Ins** that the user has used the mouse to double-click on a list entry in one of the user-defined output tabs.

Usually an **Add-In** responds to this event in order to capture activity on an output tab they had previously created through a call to Repository.AddTab().

Note that every loaded Add-In receives this event for every double-click on an output tab in Enterprise Architect, irrespective of whether the Add-In created that tab; Add-Ins should therefore check the TabName parameter supplied by this event to ensure that they are not responding to other Add-Ins' events.

Syntax

EA_OnOutputItemDoubleClicked (Repository As EA.Repository, TabName As String, LineText As String, ID As Long)

The EA_OnOutputItemClicked function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object

	representing the currently open Enterprise Architect model; poll its members to retrieve model data and user interface status information.
TabName	String Direction: IN Description: The name of the tab that the click occurred in; usually this would have been created through 'Repository.AddTab()'.
LineText	String Direction: IN Description: The text that had been supplied as the String parameter in the original call to 'Repository.WriteOutput()'.
ID	Long Direction: IN Description: The ID value specified in the original call to Repository.WriteOutput().

Return Value

None.

EA_ShowHelp

Add-Ins can use the EA_ShowHelp event to show a Help topic for a particular menu option. When the user has an **Add-In** menu option selected, pressing **F1** can be related to the required Help topic by the Add-In and a suitable Help message shown.

This event is raised when the user presses F1 on a menu option that is not a parent menu.

Syntax

Sub EA_ShowHelp (Repository as EA.Repository,
MenuLocation As String, MenuName as String, ItemName
as String)

The EA_ShowHelp function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

MenuLocation	<p>String</p> <p>Direction:</p> <p>Description: A string representing the part of the user interface that brought up the menu. This can be Treeview, MainMenu or Diagram.</p>
MenuName	<p>String</p> <p>Direction:</p> <p>Description: The name of the parent menu for which sub-items are to be defined. In the case of the top-level menu this is an empty string.</p>
ItemName	<p>String</p> <p>Direction:</p> <p>Description: The name of the option actually clicked; for example, 'Create a New Invoice'.</p>

Return Value

None.

Broadcast Events

Overview

Broadcast events are sent to all loaded **Add-Ins**. For an **Add-In** to receive the event, they must first implement the required automation event interface. If Enterprise Architect detects that the Add-In has the required interface, the event is dispatched to the Add-In.

MDG Events add a number of additional events, but the Add-In must first have registered as an MDG-style Add-In, rather than as a generic Add-In.

Event Type
Add-In License Management Events
Custom Table Events
Compartment Events
Context Item Events
File Close Event
File New Event
File Open Event

Model Validation Events
On Tab Changed Event
Post Close Diagram Event
Post Initialization Event
Post New Events
Post Open Diagram Event
Pre-Deletion Events
Pre-Exit Instance (not currently used)
On the creation of new objects
Retrieve Model Template Event
Schema Composer Events
Tagged Value Events
Technology Events
Transformation Event

Add-In License Management Events

Enterprise Architect **Add-Ins** can respond to events associated with **Add-In License Management**.

License Management Events

Event
EA_AddinLicenseValidate
EA_AddinLicenseGetDescription
EA_GetSharedAddinName

EA_AddinLicenseValidate

When a user directly enters into the 'License Management' dialog a license key that doesn't match a Sparx Systems key, EA_AddinLicenseValidate is broadcast to all Enterprise Architect **Add-Ins**, providing them with a chance to use the **Add-In** key to determine the level of functionality to provide. When a key is retrieved from the Sparx Systems **Keystore** only the target Add-In will be called with the key.

For the Add-In to validate itself against this key, the Add-In's EA_AddinLicenseValidate handler should return confirmation that the license has been validated. As the EA_AddinLicenseValidate event is broadcast to all Add-Ins, one license can validate many Add-Ins.

If an Add-In elects to handle a license key by returning a confirmation to EA_AddinLicenseValidate, it is called upon to provide a description of the license key through the EA_AddinLicenseGetDescription event. If more than one Add-In elects to handle a license key, the first Add-In that returns a confirmation to EA_AddinLicenseValidate is queried for the license key description.

Syntax

Function EA_AddinLicenseValidate (Repository As EA.Repository, AddinKey As String) As Boolean

Parameter	Type
-----------	------

Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
AddinKey	String Direction: IN Description: The Add-In license key that has been entered in the 'License Management' dialog.

Return Value

Returns **True** if the license key is validated for the current **Add-In**. Returns **False** otherwise.

EA_AddinLicenseGetDescription

Before the Enterprise Architect 'License Management' dialog is displayed, EA_AddinLicenseGetDescription is sent once for each **Add-In** key to the first Add-In that elected to handle that key.

The value returned by EA_AddinLicenseGetDescription is used as the key's plain text description.

Syntax

Function EA_AddinLicenseGetDescription (Repository as EA.Repository, AddinKey as String) As String

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open model. Poll its members to retrieve model data and user interface status information.
AddinKey	String Direction: IN Description: The Add-In license key that Enterprise Architect requires a

	description for.
--	------------------

Return Value

A String containing a plain text description of the provided AddinKey.

EA_GetSharedAddinName

As an **Add-In** writer you can distribute keys to your Add-In via the Enterprise Architect **Keystore**, provided that your keys are added using a prefix that allows the system to identify the Add-In to which they belong.

EA_GetSharedAddinName is called to determine what prefix the Add-In is using. If a matching key is found in the keystore the 'License Management' dialog will display the name returned by EA_AddinLicenseGetDescription to your users. Finally, when the user selects a key, that key will be passed to your Add-In to validate by calling EA_AddinLicenseValidate.

Syntax

Function EA_GetSharedAddinName (Repository as EA.Repository) As String

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open model. Poll its members to retrieve model data and user interface status information.

Return Value

A String containing a product name code for the provided **Add-In**, such as MYADDIN. This will be shown in plain text in any keys added to the keystore.

Notes

Shared **Add-In** keys have the format:

EASK-YOURCODE-REALKEY

- EASK - Constant string that identifies a shared key for an Enterprise Architect Add-In
- YOURCODE - The code you select and verify with us:
 - Displayed to the administrator of the keystore
 - Recommended length of 6-10 characters
 - Contains ASCII characters 33-126, except for '-' (45)
- REALKEY - Encoding of the actual key or checksums
 - Recommended length of 8-32 characters
 - Contains ASCII characters 33-126

We recommend that you contact Sparx Systems directly with proposed values to ensure that you don't clash with any other **Add-Ins**.

For example, these keys would all be interpreted as belonging to an Add-In returning MYADDIN from this

function:

- EASK-MYADDIN-Test
- EASK-MYADDIN-{7AC4D426-9083-4fa2-93B7-25E2B7FB8DC5}
- EASK-MYADDIN-7AC4D426-9083-4fa2-93B7
- EASK-MYADDIN-25E2B7FB8DC5
- EASK-MYADDIN-2hDfHKA5jf0GAjn92UvqAnxwC13dxQGJtH7zLHJ9Ym8=

Custom Table Events

The Custom Table element has an Operation called 'script', reserved for script execution, that can be used in two different, mutually exclusive ways, either:

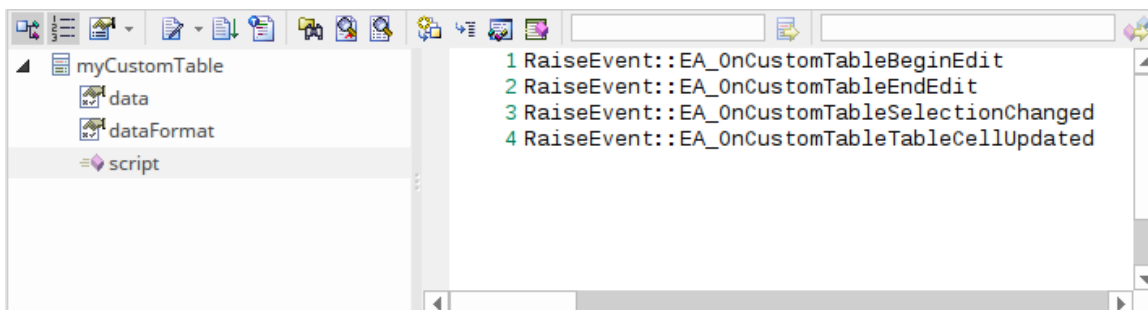
- To contain a script in JavaScript that can be executed from the element context menu; see the *Custom Table Artifact* Help topic, or
- To contain RaiseEvent broadcast calls to trigger actions from an **Add-In** written to read or update the Custom Table

Broadcasts

There are four reserved **Add-In** broadcast events that can only be enabled by listing the event in the 'script' Operation of the Custom Table element. To raise the broadcast events, list any or all of these broadcast calls in the operation named 'script'.

Syntax:

RaiseEvent::EA_OnCustomTableBeginEdit



EA_OnCustomTableBeginEdit

EA_OnCustomTableBeginEdit notifies **Add-Ins** that the Custom Table is beginning edit mode. This broadcast event can only be enabled by the Custom Table's operation 'script' behavior.

Syntax

Function EA_OnCustomTableBeginEdit (Repository As EA.Repository, Info As EA.EventProperties)

The EA_OnCustomTableBeginEdit function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the Custom Table that is under

	<p>edit:</p> <ul style="list-style-type: none">• ObjectID - A long value corresponding to the ElementID of the object
--	---

EA_OnCustomTableEndEdit

EA_OnCustomTableEndEdit notifies **Add-Ins** that a Custom Table element is ending edit mode. This broadcast event can only be enabled by the Custom Table's operation 'script' behavior.

Syntax

Function EA_OnCustomTableEndEdit (Repository As EA.Repository, Info As EA.EventProperties)

The EA_OnCustomTableEndEdit function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the Custom Table that is under

	<p>edit:</p> <ul style="list-style-type: none">• ObjectID - A long value corresponding to the ElementID of the object
--	---

Return Value

This function allows validation of the table data, and returns a Boolean value:

- **True** to save the current data in the grid, or
- **False** to abandon the current data

EA_OnCustomTableSelectionChanged

EA_OnCustomTableSelectionChanged notifies **Add-Ins** that a cell of the Custom Table has changed. This broadcast event can only be enabled by the Custom Table's operation 'script' behavior.

Syntax

Function EA_OnCustomTableSelectionChanged
(Repository As EA.Repository, Info As
EA.EventProperties)

The EA_OnCustomTableSelectionChanged function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these

	<p>EventProperty objects for the Custom Table that has been changed:</p> <ul style="list-style-type: none">• ObjectID - A long value corresponding to the ElementID of the object• RowID - A long value corresponding to the selected row id• ColID - A long value corresponding to the selected column id
--	--

EA_OnCustomTableCellUpdated

EA_OnCustomTableCellUpdated notifies **Add-Ins** that a cell value has been updated. This broadcast event can only be enabled by the Custom Table's operation 'script' behavior.

Syntax

Function EA_OnCustomTableCellUpdated (Repository As EA.Repository, Info As EA.EventProperties)

The EA_OnCustomTableCellUpdated function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the Custom

	<p>Table cell that has been changed:</p> <ul style="list-style-type: none">• ObjectID - A long value corresponding to the ElementID of the object• RowID - A long value corresponding to the selected row id• ColID - A long value corresponding to the selected column id• Value - A variant value of the changed cell data
--	---

Schema Composer Events

Enterprise Architect **Add-Ins** can respond to events associated with the **Schema Composer** to provide custom schema export formats.

The requirements for an **Add-In** to participate consist of implementing these three functions:

- EA_IsSchemaExporter
- EA_GetProfileInfo
- EA_GenerateFromSchema

EA_GenerateFromSchema

Respond to a 'Generate' request from the **Schema Composer** when using the profile type specified by the **EA_IsSchemaExporter** event. The **SchemaComposer** object can be used to traverse the schema. Export formats that have been requested by the user for generation will be listed in the **exports** parameter.

Syntax

Sub **EA_GenerateFromSchema** (Repository as **EA.Repository**, composer as **EA.SchemaComposer**, exports as **String**)

Parameter	Details
Repository	Type: EA.Repository Direction: IN Description: An EA.Repository object representing the currently open model. Poll its members to retrieve model data and user interface status information.
composer	Type: EA.SchemaComposer Direction: IN Description: Provides access to the types defined in the schema currently being

	generated. Use the <i>SchemaTypes</i> attribute to enumerate through the types and output to the appropriate export format.
exports	Type: String Direction: IN Description: Comma-separated list of export formats that the user has requested in the 'Generate' dialog.

Return Value

None.

EA_GetProfileInfo

Add-Ins can optionally implement this function to define the capabilities of the **Schema Composer** when working with the profile type specified by the `EA_IsSchemaExporter` event.

Syntax

Sub `EA_GetProfileInfo` (Repository as `EA.Repository`, profile as `EA.SchemaProfile`)

Parameter	Details
Repository	Type: <code>EA.Repository</code> Direction: IN Description: An <code>EA.Repository</code> object representing the currently open model. Poll its members to retrieve model data and user interface status information.
profile	Type: <code>EA.SchemaProfile</code> Direction: IN Description: An <code>EA.SchemaProfile</code> object representing the currently active profile type. Call the <i>SetCapability</i> function to enable or disable various capabilities of the Schema Composer . Call the

	<i>AddExportFormat</i> function to define additional export formats that this profile will support.
--	---

Return Value

None.

EA_IsSchemaExporter

Enterprise Architect **Add-Ins** can integrate with the **Schema Composer** by providing alternatives to offer users for the generation of schemas and sub models.

The **Add-In** must implement this function to be listed in the Schema Composer.

Syntax

Function EA_IsSchemaExporter(Repository as EA.Repository, ByRef displayName as String) As Boolean

Parameter	Details
Repository	Type: EA.Repository Direction: IN Description: An EA.Repository object representing the currently open model. Poll its members to retrieve model data and user interface status information.
displayName	Type: String Direction: OUT Description: The name of the custom schema set that will be provided by this Add-In .

Return Value

Return **True** to indicate that this **Add-In** will provide schema export functionality and be listed as a Schema Set when defining a new profile in the **Schema Composer**.

Compartment Events

Enterprise Architect **Add-Ins** can respond to various events associated with user-generated element compartments.

Compartment Broadcast Events

Event
EA_QueryAvailableCompartments
EA_GetCompartmentData

EA_QueryAvailableCompartments

This event occurs when Enterprise Architect's diagrams are refreshed. It is a request for the **Add-In** to provide a list of user-defined compartments.

The EA_GetCompartmentData event then queries each object for the data to display in each user-defined compartment.

Syntax

Function EA_QueryAvailableCompartments (Repository As EA.Repository) As Variant

The EA_QueryAvailableCompartments function syntax contains this parameter.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Return Value

A String containing a comma-separated list of user-defined compartments.

Example

```
Function EA_QueryAvailableCompartments(Repository As  
EA.Repository) As Variant
```

```
    Dim sReturn As String
```

```
    sReturn = ""
```

```
    If m_FirstCompartmentVisible = True Then
```

```
        sReturn = sReturn + "first,"
```

```
    End If
```

```
    If m_SecondCompartmentVisible = True Then
```

```
        sReturn = sReturn + "second,"
```

```
    End If
```

```
    If m_ThirdCompartmentVisible = True Then
```

```
        sReturn = sReturn + "third,"
```

```
    End If
```

```
    If Len(sReturn) > 0 Then
```

```
        sReturn = Left(sReturn, Len(sReturn)-1)
```

```
    End If
```

```
EA_QueryAvailableCompartments = sReturn  
End Function
```

EA_GetCompartmentData

This event occurs when Enterprise Architect is instructed to redraw an element. It requests that the **Add-In** provide the data to populate the element's compartment.

Syntax

Function EA_GetCompartmentData (Repository As EA.Repository, sCompartment As String, sGUID As String, oType As EA.ObjectType) As Variant

The EA_QueryAvailableCompartments function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
sCompartment	String Direction: IN Description: The name of the compartment for which data is being

	requested.
sGUID	String Direction: IN Description: The GUID of the element for which data is being requested.
oType	ObjectType Direction: IN Description: The type of the element for which data is being requested.

Return Value

A variant containing a formatted string. The format is illustrated in this example:

Example

Function EA_GetCompartmentData(Repository As EA.Repository, sCompartment As String, sGUID As String, oType As EA.ObjectType) As Variant

If Repository Is Nothing Then

Exit Function

End If

Dim sCompartmentData As String

Dim oXML As MSXML2.DOMDocument

Dim Nodes As MSXML2.IXMLDOMNodeList

Dim Node1 As MSXML2.IXMLDOMNode

Dim Node As MSXML2.IXMLDOMNode

Dim sData As String

sCompartmentData = ""

Set oXML = New MSXML2.DOMDocument

sData = ""

On Error GoTo ERR_GetCompartmentData

oXML.loadXML

(Repository.GetTreeXMLByGUID(sGUID))

Set Node1 = oXML.selectSingleNode("//ModelItem")

If Node1 Is Nothing Then

Exit Function

End If

sCompartmentData = sCompartmentData + "Name=" +
sCompartment + ";"

sCompartmentData = sCompartmentData +
"OwnerGUID=" + sGUID + ";"

```
sCompartmentData = sCompartmentData +  
"Options=SkipIfOnDiagram&_eq_^1&_sc_^"  
Select Case sCompartment  
Case "parts"  
Set Nodes =  
Node1.selectNodes("ModelItem(@Metatype=""Part'')")  
For Each Node In Nodes  
sData = sData + "Data&_eq_^" +  
Node.Attributes.getNamedItem("Name").nodeValue +  
"&_sc_^"  
sData = sData + "GUID&_eq_^" +  
Node.Attributes.getNamedItem("GUID").nodeValue +  
"&_sc_^,"  
Next  
Case "ports"  
Set Nodes =  
Node1.selectNodes("ModelItem(@Metatype=""Port'')")  
For Each Node In Nodes  
sData = sData + "Data&_eq_^" +  
Node.Attributes.getNamedItem("Name").nodeValue +  
"&_sc_^"  
sData = sData + "GUID&_eq_^" +  
Node.Attributes.getNamedItem("GUID").nodeValue +  
"&_sc_^,"  
Next  
End Select  
If there is no data to display, then don't return any
```

compartment data

 If sData <> "" Then

 sCompartmentData = sCompartmentData +
"CompartmentData=" + sData + ";"

 Else

 sCompartmentData = ""

 End If

EA_GetCompartmentData = sCompartmentData

Exit Function

ERR_GetCompartmentData:

EA_GetCompartmentData = ""

End Function

Context Item Events

Enterprise Architect **Add-Ins** can respond to events associated with changing context.

Context Item Broadcast Events

Event
EA_OnContextItemChanged
EA_OnContextItemDoubleClicked
EA_OnNotifyContextItemModified

EA_OnContextItemChanged

EA_OnContextItemChanged notifies **Add-Ins** that a different item is now in context.

This event occurs after a user has selected an item anywhere in the Enterprise Architect GUI. Add-Ins that require knowledge of the current item in context can subscribe to this broadcast function. If `ot = otRepository`, then this function behaves in the same way as EA_FileOpen.

Syntax

Sub EA_OnContextItemChanged (Repository As EA.Repository, GUID As String, ot as EA.ObjectType)

The EA_OnContextItemChanged function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
GUID	String

	<p>Direction: IN</p> <p>Description: Contains the GUID of the new context item. The value corresponds to these properties, depending on the value of the ot parameter:</p> <ul style="list-style-type: none">• ot (ObjectType) - GUID value• otElement - Element.ElementGUID• otPackage - Package.PackageGUID• otDiagram - Diagram.DiagramGUID• otAttribute - Attribute.AttributeGUID• otMethod - Method.MethodGUID• otConnector - Connector.ConnectorGUID• otRepository - NOT APPLICABLE, the GUID is an empty string
ot	<p>EA.ObjectType</p> <p>Direction: IN</p> <p>Description: Specifies the type of the new context item.</p>

Return Value

None.

EA_OnContextItemDoubleClicked

EA_OnContextItemDoubleClicked notifies **Add-Ins** that the user has double-clicked the item currently in context.

This event occurs when a user has double-clicked (or pressed the **Enter key**) on the item in context, either in a diagram, in the **Browser window** or in a custom compartment. Add-Ins to handle events can subscribe to this broadcast function.

Syntax

Function EA_OnContextItemDoubleClicked (Repository As EA.Repository, GUID As String, ot as EA.ObjectType)

The EA_OnContextItemDoubleClicked function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
GUID	String

	<p>Direction: IN</p> <p>Description: Contains the GUID of the new context item. The value corresponds to these properties, depending on the value of the ot parameter:</p> <ul style="list-style-type: none">• otElement - Element.ElementGUID• otPackage - Package.PackageGUID• otDiagram - Diagram.DiagramGUID• otAttribute - Attribute.AttributeGUID• otMethod - Method.MethodGUID• otConnector - Connector.ConnectorGUID
ot	<p>EA.ObjectType</p> <p>Direction: IN</p> <p>Description: Specifies the type of the new context item.</p>

Return Value

- Return **True** to notify Enterprise Architect that the double-click event has been handled by an **Add-In**
- Return **False** to enable Enterprise Architect to continue processing the event

EA_OnNotifyContextItemModified

EA_OnNotifyContextItemModified notifies **Add-Ins** that the current context item has been modified.

This event occurs when a user has modified the context item. Add-Ins that require knowledge of when an item has been modified can subscribe to this broadcast function.

Syntax

Sub EA_OnNotifyContextItemModified (Repository As EA.Repository, GUID As String, ot as EA.ObjectType)

The EA_OnNotifyContextItemModified function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
GUID	String Direction: IN

	<p>Description: Contains the GUID of the new context item. The value corresponds to these properties, depending on the value of the ot parameter:</p> <ul style="list-style-type: none">• ot(ObjectType) - GUID value• otElement - Element.ElementGUID• otPackage - Package.PackageGUID• otDiagram - Diagram.DiagramGUID• otAttribute - Attribute.AttributeGUID• otMethod - Method.MethodGUID• otConnector - Connector.ConnectorGUID
ot	<p>EA.ObjectType</p> <p>Direction: IN</p> <p>Description: Specifies the type of the new context item.</p>

Return Value

None.

EA_FileClose

The EA_FileClose event enables the **Add-In** to respond to a File Close event. When Enterprise Architect closes an opened Model file, this event is raised and passed to all **Add-Ins** implementing this method.

This event occurs when the model currently opened within Enterprise Architect is about to be closed (when another model is about to be opened or when Enterprise Architect is about to shutdown).

Syntax

Sub EA_FileClose (Repository As EA.Repository)

The EA_FileClose function syntax contains this parameter:

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the Enterprise Architect model about to be closed. Poll its members to retrieve model data and user interface status information.

Return Value

None.

EA_FileNew

The EA_FileNew event enables the **Add-In** to respond to a File New event. When Enterprise Architect creates a new model file, this event is raised and passed to all **Add-Ins** implementing this method.

The event occurs when the model being viewed by the Enterprise Architect user changes, for whatever reason (through user interaction or Add-In activity).

Syntax

Sub EA_FileNew (Repository As EA.Repository)

The EA_FileNew function syntax contains this parameter.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Return Value

None.

EA_FileOpen

The EA_FileOpen event enables the **Add-In** to respond to a File Open event. When Enterprise Architect opens a new model file, this event is raised and passed to all **Add-Ins** implementing this method.

The event occurs when the model being viewed by the Enterprise Architect user changes, for whatever reason (through user interaction or Add-In activity).

Syntax

Sub EA_FileOpen (Repository As EA.Repository)

The EA_FileOpen function syntax contains this parameter.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Return Value

None.

EA_OnPostCloseDiagram

EA_OnPostCloseDiagram notifies **Add-Ins** that a diagram has been closed.

Syntax

Function EA_OnPostCloseDiagram (Repository As EA.Repository, DiagramID As Integer)

The EA_OnPostCloseDiagram function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the Enterprise Architect model about to be closed. Poll its members to retrieve model data and user interface status information.
DiagramID	Integer Direction: IN Description: Contains the Diagram ID of the diagram that was closed.

Return Value

None.

EA_OnPostInitialized

EA_OnPostInitialized notifies **Add-Ins** that the Repository object has finished loading and any necessary initialization steps can now be performed on the object.

For example, the **Add-In** can create an 'Output' tab using Repository.CreateOutputTab.

Syntax

Sub EA_OnPostInitialized (Repository As EA.Repository)

The EA_OnPostInitialized function syntax contains this parameter.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Return Value

None.

EA_OnPostOpenDiagram

EA_OnPostOpenDiagram notifies **Add-Ins** that a diagram has been opened.

Syntax

Function EA_OnPostOpenDiagram (Repository As EA.Repository, DiagramID As Integer)

The EA_OnPostOpenDiagram function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
DiagramID	Integer Direction: IN Description: Contains the Diagram ID of the diagram that was opened.

Return Value

None.

EA_OnPostTransform

EA_OnPostTransform notifies **Add-Ins** that an MDG transformation has taken place with the output in the specified target Package.

This event occurs when a user runs an MDG transform on one or more target Packages; the notification is provided for each transform/target Package immediately after all transform processes have completed.

Syntax

Function EA_OnPostTransform (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPostTransform function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties

	<p>Direction: IN</p> <p>Description: Contains these EventProperty Objects for the transform performed:</p> <ul style="list-style-type: none">• Transform: A string value corresponding to the name of the transform used• PackageID: A long value corresponding to Package.PackageID of the destination Package
--	--

Return Value

Reserved for future use.

EA_OnPreExitInstance

EA_OnPreExitInstance is not currently used.

Syntax

Sub EA_OnPreExitInstance (Repository As EA.Repository)

The EA_OnPreExitInstance function syntax contains this parameter.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Return Value

None.

EA_OnRetrieveModelTemplate

EA_OnRetrieveModelTemplate requests that an **Add-In** pass a model template to Enterprise Architect. This event occurs when a user executes the 'Add a New Model Using Wizard' command to add a model that has been defined by an MDG Technology.

Syntax

Function EA_OnRetrieveModelTemplate (Repository As EA.Repository, sLocation As String) As String

The EA_OnRetrieveModelTemplate function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
sLocation	String Direction: IN Description: The name of the template

	requested; this should match the location attribute in the <ModelTemplates> section of an MDG Technology File.
--	--

Return Value

Return a string containing the XMI export of the model that is being used as a template.

Return an empty string if access to the template is denied; the **Add-In** is to handle user notification of the error.

Example

```
Public Function EA_OnRetrieveModelTemplate(ByRef Rep
As EA.Repository, ByRef sLocation As String) As String
Dim sTemplate As String
Select Case sLocation
Case "Templates\Template1.xml"
sTemplate = My.Resources.Template1
Case "Templates\Template2.xml"
sTemplate = My.Resources.Template2
Case "Templates\Template3.xml"
sTemplate = My.Resources.Template3
```

Case Else

MsgBox("Path for " & sLocation & " not found")

sTemplate = ""

End Select

EA_OnRetrieveModelTemplate = sTemplate

End Function

EA_OnTabChanged

EA_OnTabChanged notifies **Add-Ins** that the currently open tab has changed.

Diagrams do not generate the message when they are first opened - use the broadcast event EA_OnPostOpenDiagram for this purpose.

Syntax

Function EA_OnTabChanged (Repository As EA.Repository, TabName As String, DiagramID As Integer)

The EA_OnTabChanges function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
TabName	String Direction: IN

	Description: The name of the tab to which focus has been switched.
DiagramID	Long Direction: IN Description: The diagram ID, or 0 if switched to an Add-In tab.

Return Value

None

EA_LoadWindowManager

Enterprise Architect provides a set of Portals, each of which is a collection of shortcuts and information on performing specific areas of work on a project. The Portals help both new and experienced users quickly identify and set up the facilities they most often use in their assigned tasks.

You can add your own Portal to the system-installed set, to provide a convenient and concise call-up of one or more groups of facilities available in your **Add-In**.

Syntax

Function EA_Connect (Repository As EA.Repository) As String

The EA_Connect function syntax has this parameter:

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Example Code

```
public String EA_LoadWindowManager(EA.Repository
Repository)
{
    return Resource1.WindowManager;
}
```

Where Resource1.WindowManager is a resource file with these contents:

```
<?xml version="1.0" encoding="UTF-8"?>
<perspectives>
  <perspective name="Add-In">
    <category name="Add-In" type="commandlist"
projectrequired="true">
      <item name="Hello World" command="CallAddin"
addin="CS_AddinFramework" function="HelloWorld"/>
      <item name="Model Dump" command="CallScript"
group="Local Scripts" script="JScript - Recursive Model
Dump Example"/>
    </category>
    <category name="Open Diagrams"
type="currentdiagramlist" state = "open"/>
    <category name="Recent Diagrams"
type="recentdiagramlist" state = "open"/>
```

```
<category name="Other Windows"  
type="otherwindowlist" state = "open"/>
```

```
</perspective>
```

```
</perspectives>
```

Note that the Add-In cannot specify the icon used.

Model Validation Events

Perform Model Validation from an Add-In

Using Enterprise Architect broadcasts, it is possible to define a set of rules that are evaluated when the user instructs Enterprise Architect to perform model validation. An **Add-In** that performs model validation would involve these broadcast events.

Command	Detail
EA_OnInitializeUserRules	EA_OnInitializeUserRules is intercepted in order to define rule categories and rules.
EA_OnStartValidation	EA_OnStartValidation can be intercepted to perform any required processing prior to validation.
EA_OnEndValidation	EA_OnEndValidation can be intercepted to perform any required clean-up after validation has completed.
Validate Request	These functions intercept each request to validate an individual element, Package, diagram, connector, attribute and method.

Validate Element	EA_OnRunElementRule
Validate Package	EA_OnRunPackageRule
Validate Diagram	EA_OnRunDiagramRule
Validate Connector	EA_OnRunConnectorRule
Validate Attribute	EA_OnRunAttributeRule
Validate Method	EA_OnRunMethodRule
Validate Parameter	EA_OnRunParameterRule

EA_OnInitializeUserRules

EA_OnInitializeUserRules is called on Enterprise Architect start-up and requests that the **Add-In** provide Enterprise Architect with a rule category and list of rule IDs for model validation.

This function must be implemented by any Add-In that is to perform its own model validation. It must call Project.DefineRuleCategory once and Project.DefineRule for each rule; these functions are described in the *Project Interface* topic.

Syntax

Sub EA_OnInitializeUserRules (Repository As EA.Repository)

The EA_OnInitializeUserRules function syntax contains this parameter.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

EA_OnStartValidation

EA_OnStartValidation notifies **Add-Ins** that a user has invoked the model validation command from Enterprise Architect.

Syntax

Sub EA_OnStartValidation (Repository As EA.Repository, ParamArray Args() as Variant)

The EA_OnStartValidation function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Args	ParamArray of Variant Direction: IN Description: Contains a list of Rule Categories that are active for the current

	invocation of model validation.
--	---------------------------------

EA_OnEndValidation

EA_OnEndValidation notifies **Add-Ins** that model validation has completed.

Use this event to arrange any clean-up operations arising from the validation.

Syntax

Sub EA_OnEndValidation (Repository As EA.Repository, ParamArray Args() as Variant)

The EA_OnEndValidation function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Args	ParamArray of Variant Direction: IN Description: Contains a list of Rule

	Categories that were active for the invocation of model validation that has just completed.
--	---

EA_OnRunElementRule

This event is triggered once for each rule defined in EA_OnInitializeUserRules to be performed on each element in the selection being validated.

If you don't want to perform the rule defined by RuleID on the given element, then simply return without performing any action.

On performing any validation, if a validation error is found, use the Repository.ProjectInterface.PublishResult method to notify Enterprise Architect.

Syntax

Sub EA_OnRunElementRule (Repository As EA.Repository, RuleID As String, Element As EA.Element)

The EA_OnRunElementRule function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface

	status information.
RuleID	String Direction: IN Description: The ID that was passed into the 'Project.DefineRule' command.
Element	EA.Element Direction: IN Description: The element to potentially perform validation on.

EA_OnRunPackageRule

This event is triggered once for each rule defined in EA_OnInitializeUserRules to be performed on each Package in the selection being validated.

If you don't want to perform the rule defined by RuleID on the given Package, then simply return without performing any action.

On performing any validation, if a validation error is found, use the Repository.ProjectInterface.PublishResult method to notify Enterprise Architect.

Syntax

Sub EA_OnRunPackageRule (Repository As EA.Repository, RuleID As String, PackageID As Long)

The EA_OnRunElementRule function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface

	status information.
RuleID	String Direction: IN Description: The ID that was passed into the 'Project.DefineRule' method.
PackageID	Long Direction: IN Description: The ID of the Package to potentially perform validation on. Use the 'Repository.GetPackageByID' method to retrieve the Package object.

EA_OnRunDiagramRule

This event is triggered once for each rule defined in `EA_OnInitializeUserRules` to be performed on each diagram in the selection being validated.

If you don't want to perform the rule defined by `RuleID` on the given diagram, then simply return without performing any action.

On performing any validation, if a validation error is found, use the `Repository.ProjectInterface.PublishResult` method to notify Enterprise Architect.

Syntax

Sub `EA_OnRunDiagramRule` (`Repository` As `EA.Repository`, `RuleID` As String, `DiagramID` As Long)

The `EA_OnRunDiagramRule` function syntax contains these parameters.

Parameter	Type
Repository	<code>EA.Repository</code> Direction: IN Description: An <code>EA.Repository</code> object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface

	status information.
RuleID	String Direction: IN Description: The ID that was passed into the 'Project.DefineRule' command.
DiagramID	Long Direction: IN Description: The ID of the diagram to potentially perform validation on. Use the Repository.GetDiagramByID method to retrieve the diagram object.

EA_OnRunConnectorRule

This event is triggered once for each rule defined in EA_OnInitializeUserRules to be performed on each connector in the selection being validated.

If you don't want to perform the rule defined by RuleID on the given connector, then simply return without performing any action.

On performing any validation, if a validation error is found, use the Repository.ProjectInterface.PublishResult method to notify Enterprise Architect.

Syntax

Sub EA_OnRunConnectorRule (Repository As EA.Repository, RuleID As String, ConnectorID As Long)

The EA_OnRunConnectorRule function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface

	status information.
RuleID	String Direction: IN Description: The ID that was passed into the 'Project.DefineRule' command.
ConnectorID	Long Direction: IN Description: The ID of the connector to potentially perform validation on. Use the 'Repository.GetConnectorByID' method to retrieve the connector object.

EA_OnRunAttributeRule

This event is triggered once for each rule defined in EA_OnInitializeUserRules to be performed on each attribute in the selection being validated.

If you don't want to perform the rule defined by RuleID on the given attribute, then simply return without performing any action.

On performing any validation, if a validation error is found, use the Repository.ProjectInterface.PublishResult method to notify Enterprise Architect.

Syntax

Sub EA_OnRunAttributeRule (Repository As EA.Repository, RuleID As String, AttributeGUID As String, ObjectID As Long)

The EA_OnRunAttributeRule function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface

	status information.
RuleID	String Direction: IN Description: The ID that was passed into the 'Project.DefineRule' command.
AttributeGUID	String Direction: IN Description: The GUID of the attribute to potentially perform validation on. Use the 'Repository.GetAttributeByGuid' method to retrieve the attribute object.
ObjectID	Long Direction: IN Description: The ID of the object that owns the given attribute. Use the 'Repository.GetElementByID' method to retrieve the object.

EA_OnRunMethodRule

This event is triggered once for each rule defined in EA_OnInitializeUserRules to be performed on each method in the selection being validated.

If you don't want to perform the rule defined by RuleID on the given method, then simply return without performing any action.

On performing any validation, if a validation error is found, use the Repository.ProjectInterface.PublishResult method to notify Enterprise Architect.

Syntax

Sub EA_OnRunMethodRule (Repository As EA.Repository, RuleID As String, MethodGUID As String, ObjectID As Long)

The EA_OnRunMethodRule function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface

	status information.
RuleID	String Direction: IN Description: The ID that was passed into the 'Project.DefineRule' command.
MethodGUID	String Direction: IN Description: The GUID of the method to potentially perform validation on. Use the 'Repository.GetMethodByGuid' method to retrieve the method object.
ObjectID	Long Direction: IN Description: The ID of the object that owns the given method. Use the 'Repository.GetElementByID' method to retrieve the object.

EA_OnRunParameterRule

This event is triggered once for each rule defined in EA_OnInitializeUserRules to be performed on each parameter in the selection being validated.

If you don't want to perform the rule defined by RuleID on the given parameter, then simply return without performing any action.

On performing any validation, if a validation error is found, use the Repository.ProjectInterface.PublishResult method to notify Enterprise Architect.

Syntax

Sub EA_OnRunParameterRule (Repository As EA.Repository, RuleID As String, ParameterGUID As String, MethodGUID As String, ObjectID As Long)

The EA_OnRunMethodRule function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface

	status information.
RuleID	String Direction: IN Description: The ID that was passed into the 'Project.DefineRule' command.
ParameterGUID	String Direction: IN Description: The GUID of the parameter to potentially perform validation on. Use this to retrieve the parameter by iterating through the 'Method.Parameters' collection.
MethodGUID	String Direction: IN Description: The GUID of the method that owns the given parameter. Use the 'Repository.GetMethodByGuid' method to retrieve the method object.
ObjectID	Long Direction: IN Description: The ID of the object that owns the given parameter. Use the 'Repository.GetElementByID' method to retrieve the object.

Model Validation Example

This example code is written in C# and provides a skeleton model validation implementation that you might want to use as a starting point in writing your own model validation rules.

Main.cs

```
using System;
namespace myAddin
{
    public class Main
    {
        public Rules theRules;
        public Main()
        {
            theRules = new Rules();
        }
        public string EA_Connect(EA.Repository Repository)
        {
            return "";
        }
        public void EA_Disconnect()
        {
```

```
        GC.Collect();
        GC.WaitForPendingFinalizers();
    }
    private bool IsProjectOpen(EA.Repository
Repository)
    {
        try
        {
            EA.Collection c = Repository.Models;
            return true;
        }
        catch
        {
            return false;
        }
    }
    public object EA_GetMenuItems(EA.Repository
Repository, string MenuLocation, string MenuName)
    {
        switch (MenuName)
        {
            case "":
                return "-&myAddin";
            case "-&myAddin":
                string() ar = { "&Test" };
        }
    }
}
```

```
        return ar;
    }
    return "";
}

public void EA_GetMenuState(EA.Repository
Repository, string MenuLocation, string MenuName,
    string ItemName, ref bool IsEnabled, ref bool
IsChecked)
{
    // if no open project, disable all menu options
    if (IsProjectOpen(Repository))
        IsEnabled = true;
    else
        IsEnabled = false;
}

public void EA_MenuClick(EA.Repository
Repository, string MenuLocation, string MenuName, string
ItemName)
{
    switch (ItemName)
    {
        case "&Test";
            DoTest(Repository);
            break;
    }
}
```

```
    }  
    public void  
EA_OnInitializeUserRules(EA.Repository Repository)  
    {  
        if (Repository != null)  
        {  
            theRules.ConfigureCategories(Repository);  
            theRules.ConfigureRules(Repository);  
        }  
    }  
    public void EA_OnRunElementRule(EA.Repository  
Repository, string RuleID, EA.Element element)  
    {  
        theRules.RunElementRule(Repository, RuleID,  
element);  
    }  
    public void EA_OnRunDiagramRule(EA.Repository  
Repository, string RuleID, long lDiagramID)  
    {  
        theRules.RunDiagramRule(Repository, RuleID,  
lDiagramID);  
    }  
    public void  
EA_OnRunConnectorRule(EA.Repository Repository,  
string RuleID, long lConnectorID)  
    {
```

```
        theRules.RunConnectorRule(Repository, RuleID,
lConnectorID);
    }
    public void EA_OnRunAttributeRule(EA.Repository
Repository, string RuleID, string AttGUID, long lObjectID)
    {
        return;
    }
    public void EA_OnDeleteTechnology(EA.Repository
Repository, EA.EventProperties Info)
    {
        return;
    }
    public void EA_OnImportTechnology(EA.Repository
Repository, EA.EventProperties Info)
    {
        return;
    }
    private void DoTest(EA.Repository Rep)
    {
        // TODO: insert test code here
    }
}
}
```

Rules.cs

```
using System;
using System.Collections;
namespace myAddin
{
    public class Rules
    {
        private string m_sCategoryID;
        private System.Collections.ArrayList m_RuleIDs;
        private System.Collections.ArrayList m_RuleIDEx;
        private const string cRule01 = "Rule01";
        private const string cRule02 = "Rule02";
        private const string cRule03 = "Rule03";
        // TODO: expand this list as much as necessary
        public Rules()
        {
            m_RuleIDs = new System.Collections.ArrayList();
            m_RuleIDEx = new
System.Collections.ArrayList();
        }
        private string LookupMap(string sKey)
        {
            return DoLookupMap(sKey, m_RuleIDs,
```



```
m_RuleIDEx);
    }
    private string LookupMapEx(string sRule)
    {
        return DoLookupMap(sRule, m_RuleIDEx,
m_RuleIDs);
    }
    private string DoLookupMap(string sKey, ArrayList
arrValues, ArrayList arrKeys)
    {
        if (arrKeys.Contains(sKey))
            return
arrValues(arrKeys.IndexOf(sKey)).ToString();
        else
            return "";
    }
    private void AddToMap(string sRuleID, string sKey)
    {
        m_RuleIDs.Add(sRuleID);
        m_RuleIDEx.Add(sKey);
    }
    private string GetRuleStr(string sRuleID)
    {
        switch (sRuleID)
        {
```

```
        case cRule01:
            return "Error Message 01";
        case cRule02:
            return "Error Message 02";
        case cRule03:
            return "Error Message 03";
        // TODO: add extra cases as much as necessary
    }
    return "";
}

public void ConfigureCategories(EA.Repository
Repository)
{
    EA.Project Project =
Repository.GetProjectInterface();
    m_sCategoryID =
Project.DefineRuleCategory("Enterprise Collaboration
Architecture (ECA) Rules");
}

public void ConfigureRules(EA.Repository
Repository)
{
    EA.Project Project =
Repository.GetProjectInterface();
    AddToMap(Project.DefineRule(m_sCategoryID,
EA.EnumMVErrorType.mvError, GetRuleStr(cRule01)),
```

```
cRule01);
    AddToMap(Project.DefineRule(m_sCategoryID,
EA.EnumMVErrorType.mvError, GetRuleStr(cRule02)),
cRule02);
    AddToMap(Project.DefineRule(m_sCategoryID,
EA.EnumMVErrorType.mvError, GetRuleStr(cRule03)),
cRule03);
    // TODO: expand this list
}
public void RunConnectorRule(EA.Repository
Repository, string sRuleID, long lConnectorID)
{
    EA.Connector Connector =
Repository.GetConnectorByID((int)lConnectorID);
    if (Connector != null)
    {
        switch (LookupMapEx(sRuleID))
        {
            case cRule02:
                // TODO: perform rule 2 check
                break;
                // TODO: add more cases
        }
    }
}
public void RunDiagramRule(EA.Repository
```

```
Repository, string sRuleID, long lDiagramID)
{
    EA.Diagram Diagram =
Repository.GetDiagramByID((int)lDiagramID);
    if (Diagram != null)
    {
        switch (LookupMapEx(sRuleID))
        {
            case cRule03:
                // TODO: perform rule 3 check
                break;
            // TODO: add more cases
        }
    }
}

public void RunElementRule(EA.Repository
Repository, string sRuleID, EA.Element Element)
{
    if (Element != null)
    {
        switch (LookupMapEx(sRuleID))
        {
            case cRule01:
                DoRule01(Repository, Element);
                break;
```

```
        // TODO: add more cases
    }
}
}
private void DoRule01(EA.Repository Repository,
EA.Element Element)
{
    if (Element.Stereotype != "myStereotype")
        return;
    // TODO: validation logic here
    // report validation errors
    EA.Project Project =
Repository.GetProjectInterface();
    Project.PublishResult(LookupMap(cRule01),
EA.EnumMVErrorType.mvError, GetRuleStr(cRule01));
}
}
}
```

Post-New Events

Enterprise Architect **Add-Ins** can respond to the creation of new elements, connectors, objects, attributes, methods and Packages using these broadcast events:

Post-New Broadcast Events

Event
EA_OnPostNewElement
EA_OnPostNewConnector
EA_OnPostNewDiagram
EA_OnPostNewDiagramObject
EA_OnPostNewAttribute
EA_OnPostNewMethod
EA_OnPostNewPackage
EA_OnPostNewGlossaryTerm

EA_OnPostNewElement

EA_OnPostNewElement notifies **Add-Ins** that a new element has been created on a diagram. It enables Add-Ins to modify the element upon creation.

This event occurs after a user has dragged a new element from the Toolbox or 'Resources' tab of the **Browser window** onto a diagram. The notification is provided immediately after the element is added to the model.

Set Repository.SuppressEADialogs to **True** to suppress Enterprise Architect from showing its default 'Properties' dialog.

Syntax

Function EA_OnPostNewElement (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPostNewElement function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface

	status information.
Info	<p>EA.EventProperties</p> <p>Direction: IN</p> <p>Description: Contains this EventProperty object for the new element:</p> <ul style="list-style-type: none">• ElementID: A long value corresponding to Element.ElementID

Return Value

Return **True** if the element has been updated during this notification. Return **False** otherwise.

EA_OnPostNewConnector

EA_OnPostNewConnector notifies **Add-Ins** that a new connector has been created on a diagram. It enables Add-Ins to modify the connector upon creation.

This event occurs after a user has dragged a new connector from the Toolbox or 'Resources' tab of the **Browser window** onto a diagram. The notification is provided immediately after the connector is added to the model.

Syntax

Function EA_OnPostNewConnector (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPostNewConnector function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties

	<p>Direction: IN</p> <p>Description: Contains this EventProperty object for the new connector:</p> <ul style="list-style-type: none">• ConnectorID: A long value corresponding to Connector.ConnectorID
--	---

Return Value

Return **True** if the connector has been updated during this notification. Return **False** otherwise.

EA_OnPostNewDiagram

EA_OnPostNewDiagram notifies **Add-Ins** that a new diagram has been created. It enables Add-Ins to modify the diagram upon creation.

Syntax

Function EA_OnPostNewDiagram (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPostNewDiagram function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the new diagram:

- | | |
|--|--|
| | <ul style="list-style-type: none">• DiagramID: A long value corresponding to Diagram.PackageID |
|--|--|

Return Value

Return **True** if the diagram has been updated during this notification. Return **False** otherwise.

EA_OnPostNewDiagramObject

EA_OnPostNewDiagramObject notifies **Add-Ins** that a new object has been created on a diagram. It enables Add-Ins to modify the object upon creation.

This event occurs after a user has dragged a new object directly from the **Browser window** or from the 'Resources' tab of the Browser window onto a diagram. The notification is provided immediately after the object is added to the diagram.

Syntax

Function EA_OnPostNewDiagramObject (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPostNewDiagramObject function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Info	<p>EA.EventProperties</p> <p>Direction: IN</p> <p>Description: Contains these EventProperty objects for the new element:</p> <ul style="list-style-type: none">• ID: A long value corresponding to the ElementID of the object that has been added to the diagram• DiagramID: A long value corresponding to the DiagramID of the diagram to which the object has been added• DUID: A string value for the DUID; can be used with Diagram.GetDiagramObjectByID to retrieve the new DiagramObject
------	---

Return Value

Return **True** if the element has been updated during this notification. Return **False** otherwise.

EA_OnPostNewAttribute

EA_OnPostNewAttribute notifies **Add-Ins** that a new attribute has been created on a diagram. It enables Add-Ins to modify the attribute upon creation.

This event occurs when a user creates a new attribute on an element by either drag-and-dropping from the **Browser window**, using the 'Attributes' tab of the **Features window**, or using the in-place editor on the diagram. The notification is provided immediately after the attribute is created.

Syntax

Function EA_OnPostNewAttribute (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPostNewAttribute function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Info	<p>EA.EventProperties</p> <p>Direction: IN</p> <p>Description: Contains this EventProperty object for the new attribute:</p> <ul style="list-style-type: none">• AttributeID: A long value corresponding to Attribute.AttributeID
------	---

Return Value

Return **True** if the attribute has been updated during this notification. Return **False** otherwise.

EA_OnPostNewMethod

EA_OnPostNewMethod notifies **Add-Ins** that a new method has been created on a diagram. It enables Add-Ins to modify the method upon creation.

This event occurs when a user creates a new method on an element by either drag-dropping from the **Browser window**, using the method's 'Properties' dialog, or using the in-place editor on the diagram. The notification is provided immediately after the method is created.

Syntax

Function EA_OnPostNewMethod (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPostNewMethod function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Info	<p>EA.EventProperties</p> <p>Direction: IN</p> <p>Description: Contains this EventProperty object for the new method:</p> <ul style="list-style-type: none">• MethodID: A long value corresponding to Method.MethodID
------	---

Return Value

Return **True** if the method has been updated during this notification. Return **False** otherwise.

EA_OnPostNewPackage

EA_OnPostNewPackage notifies **Add-Ins** that a new Package has been created on a diagram. It enables Add-Ins to modify the Package upon creation.

This event occurs when a user drags a new Package from the Toolbox or 'Resources' tab of the **Browser window** onto a diagram, or by selecting the **New Package icon** from the Browser window.

Syntax

Function EA_OnPostNewPackage (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPostNewPackage function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties

	<p>Direction: IN</p> <p>Description: Contains this EventProperty object for the new Package:</p> <ul style="list-style-type: none">• PackageID: A long value corresponding to Package.PackageID
--	---

Return Value

Return **True** if the Package has been updated during this notification. Return **False** otherwise.

EA_OnPostNewGlossaryTerm

EA_OnPostNewGlossaryTerm notifies **Add-Ins** that a new glossary term has been created. It enables Add-Ins to modify the glossary term upon creation.

The notification is provided immediately after the glossary term is added to the model.

Syntax

Function EA_OnPostNewGlossaryTerm (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPostNewGlossaryTerm function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN

	<p>Description: Contains these EventProperty objects for the new glossary term:</p> <ul style="list-style-type: none">• TermID: A string value corresponding to Term.TermID• Term: A string value corresponding to the name of the glossary term being created• Meaning: A string value corresponding to meaning of the glossary term being created
--	---

Return Value

Return **True** if the glossary term has been updated during this notification. Return **False** otherwise.

Pre-Deletion Events

Enterprise Architect **Add-Ins** can respond to requests to delete elements, attributes, methods, connectors, diagrams, Packages and glossary terms using these broadcast events:

Pre-Deletion Broadcast Events

Event
EA_OnPreDeleteElement
EA_OnPreDeleteAttribute
EA_OnPreDeleteMethod
EA_OnPreDeleteConnector
EA_OnPreDeleteDiagram
EA_OnPreDeletePackage
EA_OnPreDeleteGlossaryTerm
EA_OnPreDeleteTechnology (Deprecated)

EA_OnPreDeleteElement

EA_OnPreDeleteElement notifies **Add-Ins** that an element is to be deleted from the model. It enables Add-Ins to permit or deny deletion of the element.

This event occurs when a user deletes an element from the **Browser window** or on a diagram. The notification is provided immediately before the element is deleted, so that the **Add-In** can disable deletion of the element.

Syntax

Function EA_OnPreDeleteElement (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreDeleteElement function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties

	<p>Direction: IN</p> <p>Description: Contains this EventProperty object for the element to be deleted:</p> <ul style="list-style-type: none">• ElementID: A long value corresponding to Element.ElementID
--	---

Return Value

- Return **True** to enable deletion of the element from the model
- Return **False** to disable deletion of the element

EA_OnPreDeleteAttribute

EA_OnPreDeleteAttribute notifies **Add-Ins** that an attribute is to be deleted from the model. It enables Add-Ins to permit or deny deletion of the attribute.

This event occurs when a user attempts to permanently delete an attribute from the **Browser window**. The notification is provided immediately before the attribute is deleted, so that the **Add-In** can disable deletion of the attribute.

Syntax

Function EA_OnPreDeleteAttribute (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreDeleteAttribute function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Info	<p>EA.EventProperties</p> <p>Direction: IN</p> <p>Description: Contains this EventProperty object for the attribute to be deleted:</p> <ul style="list-style-type: none">• AttributeID: A long value corresponding to Attribute.AttributeID
------	---

Return Value

- Return **True** to enable deletion of the attribute from the model
- Return **False** to disable deletion of the attribute

EA_OnPreDeleteMethod

EA_OnPreDeleteMethod notifies **Add-Ins** that a method (operation) is to be deleted from the model. It enables Add-Ins to permit or deny deletion of the method.

This event occurs when a user attempts to permanently delete a method from the **Browser window**. The notification is provided immediately before the method is deleted, so that the **Add-In** can disable deletion of the method.

Syntax

Function EA_OnPreDeleteMethod (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreDeleteMethod function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Info	<p>EA.EventProperties</p> <p>Direction: IN</p> <p>Description: Contains this EventProperty object for the method to be deleted:</p> <ul style="list-style-type: none">• MethodID: A long value corresponding to Method.MethodID
------	---

Return Value

- Return **True** to enable deletion of the method from the model
- Return **False** to disable deletion of the method

EA_OnPreDeleteConnector

EA_OnPreDeleteConnector notifies **Add-Ins** that a connector is to be deleted from the model. It enables Add-Ins to permit or deny deletion of the connector.

This event occurs when a user attempts to permanently delete a connector on a diagram. The notification is provided immediately before the connector is deleted, so that the **Add-In** can disable deletion of the connector.

Syntax

Function EA_OnPreDeleteConnector (Repository As EA.Repository, Info As EA.EventProperties) As Boolean
The EA_OnPreDeleteConnector function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties

	<p>Direction: IN</p> <p>Description: Contains this EventProperty object for the connector to be deleted:</p> <ul style="list-style-type: none">• ConnectorID: A long value corresponding to Connector.ConnectorID
--	---

Return Value

- Return **True** to enable deletion of the connector from the model
- Return **False** to disable deletion of the connector

EA_OnPreDeleteDiagram

EA_OnPreDeleteDiagram notifies **Add-Ins** that a diagram is to be deleted from the model. It enables Add-Ins to permit or deny deletion of the diagram.

This event occurs when a user attempts to permanently delete a diagram from the **Browser window**. The notification is provided immediately before the diagram is deleted, so that the **Add-In** can disable deletion of the diagram.

Syntax

Function EA_OnPreDeleteDiagram (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreDeleteDiagram function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Info	<p>EA.EventProperties</p> <p>Direction: IN</p> <p>Description: Contains this EventProperty object for the diagram to be deleted:</p> <ul style="list-style-type: none">• DiagramID: A long value corresponding to Diagram.DiagramID
------	---

Return Value

- Return **True** to enable deletion of the diagram from the model
- Return **False** to disable deletion of the diagram

EA_OnPreDeleteDiagramObject

EA_OnPreDeleteDiagramObject notifies **Add-Ins** that a diagram object is to be deleted from the model. It enables Add-Ins to permit or deny deletion of the element.

This event occurs when a user attempts to permanently delete an element from a diagram. The notification is provided immediately before the element is deleted, so that the **Add-In** can disable deletion of the element.

Syntax

Function EA_OnPreDeleteDiagramObject (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreDeleteDiagramObject function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties

	<p>Direction: IN</p> <p>Description: Contains this EventProperty object for the element to be deleted:</p> <ul style="list-style-type: none">• ID: A long value corresponding to DiagramObject.ElementID
--	--

Return Value

- Return **True** to enable deletion of the element from the model
- Return **False** to disable deletion of the element

EA_OnPreDeletePackage

EA_OnPreDeletePackage notifies **Add-Ins** that a Package is to be deleted from the model. It enables Add-Ins to permit or deny deletion of the Package.

This event occurs when a user attempts to permanently delete a Package from the **Browser window**. The notification is provided immediately before the Package is deleted, so that the **Add-In** can disable deletion of the Package.

Syntax

Function EA_OnPreDeletePackage (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreDeletePackage function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Info	<p>EA.EventProperties</p> <p>Direction: IN</p> <p>Description: Contains this EventProperty object for the Package to be deleted:</p> <ul style="list-style-type: none">• PackageID: A long value corresponding to Package.PackageID
------	---

Return Value

- Return **True** to enable deletion of the Package from the model
- Return **False** to disable deletion of the Package

EA_OnPreDeleteGlossaryTerm

EA_OnPreDeleteGlossaryTerm notifies **Add-Ins** that a glossary term is to be deleted from the model. It enables Add-Ins to permit or deny deletion of the glossary term.

The notification is provided immediately before the glossary term is deleted, so that the **Add-In** can disable deletion of the glossary term.

Syntax

Function EA_OnPreDeleteGlossaryTerm (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreDeleteGlossaryTerm function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties

	<p>Direction: IN</p> <p>Description: Contains this EventProperty object for the glossary term to be deleted:</p> <ul style="list-style-type: none">• TermID: A long value corresponding to Term.TermID
--	--

Return Value

- Return **True** to enable deletion of the glossary term from the model
- Return **False** to disable deletion of the glossary term

Pre New-Object Events

When you create an **Add-In**, you can include broadcast events to intercept and respond to requests to create new objects, including elements, connectors, diagram objects, attributes, methods and Packages.

Events to intercept

Event
Creation of a new element
Creation of a new connector
Creation of a new diagram
Creation of a new diagram object
Creation of a new element by dropping onto a diagram from the Browser window .
Creation of a new attribute
Creation of a new method
Creation of a new Package

Creation of a new glossary term

EA_OnPreNewElement

EA_OnPreNewElement notifies **Add-Ins** that a new element is about to be created on a diagram. It enables Add-Ins to permit or deny creation of the new element.

This event occurs when a user drags a new element from the Toolbox or 'Resources' tab of the **Browser window** onto a diagram. The notification is provided immediately before the element is created, so that the **Add-In** can disable addition of the element.

Syntax

Function EA_OnPreNewElement (Repository As EA.Repository, Info As EA.EventProperties) As Boolean
The EA_OnPreNewElement function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Info	<p>EA.EventProperties</p> <p>Direction: IN</p> <p>Description: Contains these EventProperty objects for the element to be created:</p> <ul style="list-style-type: none">• Type: A string value corresponding to Element.Type• FQStereotype: A string value corresponding to Element.FQStereotype• Stereotype: A string value corresponding to Element.Stereotype• ParentID: A long value corresponding to Element.ParentID• DiagramID: A long value corresponding to the ID of the diagram to which the element is being added
------	---

Return Value

- Return **True** to enable addition of the new element to the model
- Return **False** to disable addition of the new element

EA_OnPreNewConnector

EA_OnPreNewConnector notifies **Add-Ins** that a new connector is about to be created on a diagram. It enables Add-Ins to permit or deny creation of a new connector.

This event occurs when a user drags a new connector from the Toolbox or 'Resources' tab of the **Browser window**, onto a diagram. The notification is provided immediately before the connector is created, so that the **Add-In** can disable addition of the connector.

Syntax

Function EA_OnPreNewConnector (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreNewConnector function syntax contains these elements:

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Info	<p>EA.EventProperties</p> <p>Direction: IN</p> <p>Description: Contains these EventProperty objects for the connector to be created:</p> <ul style="list-style-type: none">• Type: A string value corresponding to Connector.Type• Subtype: A string value corresponding to Connector.Subtype• Stereotype: A string value corresponding to Connector.Stereotype• ClientID: A long value corresponding to Connector.ClientID• SupplierID: A long value corresponding to Connector.SupplierID• DiagramID: A long value corresponding to Connector.DiagramID
------	--

Return Value

- Return **True** to enable addition of the new connector to the model
- Return **False** to disable addition of the new connector

EA_OnPreNewDiagram

EA_OnPreNewDiagram notifies **Add-Ins** that a new diagram is about to be created. It enables Add-Ins to permit or deny creation of the new diagram.

The notification is provided immediately before the diagram is created, so that the **Add-In** can disable addition of the diagram.

Syntax

Function EA_OnPreNewDiagram (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreNewDiagram function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties

	<p>Direction: IN</p> <p>Description: Contains these EventProperty objects for the diagram to be created:</p> <ul style="list-style-type: none">• Type: A string value corresponding to Diagram.Type• ParentID: A long value corresponding to Diagram.ParentID• PackageID: A long value corresponding to Diagram.PackageID
--	---

Return Value

- Return **True** to enable addition of the new diagram to the model
- Return **False** to disable addition of the new diagram

EA_OnPreNewDiagramObject

EA_OnPreNewDiagramObject notifies **Add-Ins** that a new diagram object is about to be dropped on a diagram. It enables Add-Ins to permit or deny creation of the new object.

This event occurs when a user drags an object directly from the Enterprise Architect **Browser window** or from the 'Resources' tab of the Browser window onto a diagram. The notification is provided immediately before the object is created, so that the **Add-In** can disable addition of the object.

Syntax

Function EA_OnPreNewDiagramObject (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreNewDiagramObject function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface

	status information.
Info	<p>EA.EventProperties</p> <p>Direction: IN</p> <p>Description: Contains these EventProperty objects for the object to be created:</p> <ul style="list-style-type: none">• Type: A string value corresponding to the Type of object being added to the diagram• Stereotype: A string value corresponding to the Stereotype of the object being added to the diagram• ID: A long value corresponding to the ID of the element, Package or diagram being added to the diagram• DiagramID: A long value corresponding to the ID of the diagram to which the object is being added

Return Value

- Return **True** to enable addition of the object to the model
- Return **False** to disable addition of the object

EA_OnPreDropFromTree

When a user drags any kind of element from the **Browser window** onto a diagram, EA_OnPreDropFromTree notifies the **Add-In** that a new item is about to be dropped onto a diagram. The notification is provided immediately before the element is dropped, so that the Add-In can override the default action that would be taken for this drag.

Syntax

Function EA_OnPreDropFromTree (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreDropFromTree function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN

	<p>Description: Contains these EventProperty objects for the element to be created:</p> <ul style="list-style-type: none">• ID: A long value of the type being dropped• Type: A string value corresponding to type of element being dropped• DiagramID: A long value corresponding to the ID of the diagram to which the element is being added• PositionX: The X coordinate into which the element is being dropped• PositionY: The Y coordinate into which the element is being dropped• DroppedID: A long value corresponding to the ID of the element the item has been dropped onto
--	--

Return Value

- Return **True** to allow the default behavior to be executed
- Return **False** to override this behavior

EA_OnPreNewAttribute

EA_OnPreNewAttribute notifies **Add-Ins** that a new attribute is about to be created on an element. It enables Add-Ins to permit or deny creation of the new attribute.

This event occurs when a user creates a new attribute on an element by either drag-dropping from the **Browser window**, using the 'Attributes' tab of the **Features window**, or using the in-place editor on the diagram. The notification is provided immediately before the attribute is created, so that the **Add-In** can disable addition of the attribute.

Syntax

Function EA_OnPreNewAttribute (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreNewAttribute function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Info	<p>EA.EventProperties</p> <p>Direction: IN</p> <p>Description: Contains these EventProperty objects for the attribute to be created:</p> <ul style="list-style-type: none">• Type: A string value corresponding to Attribute.Type• Stereotype: A string value corresponding to Attribute.Stereotype• ParentID: A long value corresponding to Attribute.ParentID• ClassifierID: A long value corresponding to Attribute.ClassifierID
------	--

Return Value

- Return **True** to enable addition of the new attribute to the model
- Return **False** to disable addition of the new attribute

EA_OnPreNewMethod

EA_OnPreNewMethod notifies **Add-Ins** that a new method is about to be created on an element. It enables Add-Ins to permit or deny creation of the new method.

This event occurs when a user creates a new method on an element by either drag-dropping from the **Browser window**, using the 'Operations' tab of the **Features window**, or using the in-place editor on the diagram. The notification is provided immediately before the method is created, so that the **Add-In** can disable addition of the method.

Syntax

Function EA_OnPreNewMethod (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreNewMethod function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Info	<p>EA.EventProperties</p> <p>Direction: IN</p> <p>Description: Contains these EventProperty objects for the method to be created:</p> <ul style="list-style-type: none">• ReturnType: A string value corresponding to Method.ReturnType• Stereotype: A string value corresponding to Method.Stereotype• ParentID: A long value corresponding to Method.ParentID• ClassifierID: A long value corresponding to Method.ClassifierID
------	---

Return Value

- Return **True** to enable addition of the new method to the model
- Return **False** to disable addition of the new method

EA_OnPreNewPackage

EA_OnPreNewPackage notifies **Add-Ins** that a new Package is about to be created in the model. It enables Add-Ins to permit or deny creation of the new Package.

This event occurs when a user drags a new Package from the Toolbox or 'Resources' tab of the **Browser window** onto a diagram, or by selecting the **New Package icon** from the Browser window. The notification is provided immediately before the Package is created, so that the **Add-In** can disable addition of the Package.

Syntax

Function EA_OnPreNewPackage (Repository As EA.Repository, Info As EA.EventProperties) As Boolean
The EA_OnPreNewPackage function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Info	<p>EA.EventProperties</p> <p>Direction: IN</p> <p>Description: Contains these EventProperty objects for the Package to be created:</p> <ul style="list-style-type: none">• Stereotype: A string value corresponding to Package.Stereotype• ParentID: A long value corresponding to Package.ParentID• DiagramID: A long value corresponding to the ID of the diagram to which the Package is being added
------	---

Return Value

- Return **True** to enable addition of the new Package to the model
- Return **False** to disable addition of the new Package

EA_OnPreNewGlossaryTerm

EA_OnPreNewGlossaryTerm notifies **Add-Ins** that a new glossary term is about to be created. It enables Add-Ins to permit or deny creation of the new glossary term.

The notification is provided immediately before the glossary term is created, so that the **Add-In** can disable addition of the element.

Syntax

Function EA_OnPreNewGlossaryTerm (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreNewGlossaryTerm function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties

	<p>Direction: IN</p> <p>Description: Contains these EventProperty objects for the glossary term to be created:</p> <ul style="list-style-type: none">• TermID: A string value corresponding to Term.TermID• Term: A string value corresponding to the name of the glossary term being created• Meaning: A string value corresponding to meaning of the glossary term being created
--	--

Return Value

- Return **True** to enable addition of the new glossary term to the model
- Return **False** to disable addition of the new glossary term


Tagged Value Events

Enterprise Architect includes the Addin Broadcast Tagged Value type that allows an **Add-In** to respond to attempts to edit it. The function that is called depends on the type of object the Tagged Value is on.

Tagged Value Events

Event
EA_OnAttributeTagEdit
EA_OnConnectorTagEdit
EA_OnElementTagEdit
EA_OnMethodTagEdit

EA_OnAttributeTagEdit

EA_OnAttributeTagEdit is called when the user clicks the  button for a Tagged Value of type AddinBroadcast on an attribute.

The **Add-In** displays fields to show and change the value and notes; this function provides the initial values for the Tagged Value notes and value, and takes on any changes on exit of the function.

Syntax


Sub EA_OnAttributeTagEdit (Repository As EA.Repository, AttributeID As Long, String TagName, String TagValue, String TagNotes)

The EA_OnAttributeTagEdit function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

AttributeID	Long Direction: IN Description: The ID of the attribute that this Tagged Value is on.
TagName	String Direction: IN Description: The name of the Tagged Value to edit.
TagValue	String Direction: INOUT Description: The current value of the tag; if the value is updated, the new value is stored in the repository on exit of the function.
TagNotes	String Direction: INOUT Description: The current value of the Tagged Value notes; if the value is updated, the new value is stored in the repository on exit of the function.

EA_OnConnectorTagEdit

EA_OnConnectorTagEdit is called when the user clicks the  button for a Tagged Value of type AddinBroadcast on a connector.

The **Add-In** displays fields to show and change the value and notes; this function provides the initial values for the Tagged Value notes and value, and takes on any changes on exit of the function.

Syntax


Sub EA_OnConnectorTagEdit (Repository As EA.Repository, ConnectorID As Long, String TagName, String TagValue, String TagNotes)

The EA_OnConnectorTagEdit function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

ConnectorID	Long Direction: IN Description: The ID of the connector that this Tagged Value is on.
TagName	String Direction: IN Description: The name of the Tagged Value to edit.
TagValue	String Direction: INOUT Description: The current value of the tag; if the value is updated, the new value is stored in the repository on exit of the function.
TagNotes	String Direction: INOUT Description: The current value of the Tagged Value notes; if the value is updated, the new value is stored in the repository on exit of the function.

EA_OnElementTagEdit

EA_OnElementTagEdit is called when the user clicks the  button for a Tagged Value of type AddinBroadcast on an element.

The **Add-In** displays fields to show and change the value and notes; this function provides the initial values for the Tagged Value notes and value, and takes on any changes on exit of the function.

Syntax


Sub EA_OnElementTagEdit (Repository As EA.Repository, ObjectID As Long, String TagName, String TagValue, String TagNotes)

The EA_OnElementTagEdit function syntax contains these elements:

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

ObjectID	Long Direction: IN Description: The ID of the object (element) that this Tagged Value is on.
TagName	String Direction: IN Description: The name of the Tagged Value to edit.
TagValue	String Direction: INOUT Description: The current value of the tag; if the value is updated, the new value is stored in the repository on exit of the function.
TagNotes	String Direction: INOUT Description: The current value of the Tagged Value notes; if the value is updated, the new value is stored in the repository on exit of the function.

EA_OnMethodTagEdit

EA_OnMethodTagEdit is called when the user clicks the  button for a Tagged Value of type AddinBroadcast on an operation.

The **Add-In** displays fields to show and change the value and notes; this function provides the initial values for the Tagged Value notes and value, and takes on any changes on exit of the function.

Syntax

Sub EA_OnMethodTagEdit (Repository As EA.Repository, MethodID As Long, String TagName, String TagValue, String TagNotes)

The EA_OnMethodTagEdit function syntax contains these elements:

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

MethodID	Long Direction: IN Description: The ID of the method that this Tagged Value is on.
TagName	String Direction: IN Description: The name of the Tagged Value to edit.
TagValue	String Direction: INOUT Description: The current value of the tag; if the value is updated, the new value is stored in the repository on exit of the function.
TagNotes	String Direction: INOUT Description: The current value of the Tagged Value notes; if the value is updated, the new value is stored in the repository on exit of the function.

Technology Events

Enterprise Architect **Add-Ins** can respond to events associated with the use of MDG Technologies.

Technology Broadcast Events

Event
EA_OnInitializeTechnologies
EA_OnPreActivateTechnology
EA_OnPostActivateTechnology
EA_OnPreDeleteTechnology (Deprecated)
EA_OnDeleteTechnology (Deprecated)
EA_OnImportTechnology (Deprecated)

EA_OnInitializeTechnologies

EA_OnInitializeTechnologies requests that an **Add-In** pass an MDG Technology to Enterprise Architect for loading.

This event occurs on Enterprise Architect start up. Return your technology XML to this function and Enterprise Architect loads and enables it.

Syntax

Function EA_OnInitializeTechnologies (Repository As EA.Repository) As Object

The EA_OnInitializeTechnologies function syntax contains this parameter:

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Return Value

Return the MDG Technology as a single XML string.

Example

```
Public Function EA_OnInitializeTechnologies(ByVal  
Repository As EA.Repository) As Object
```

```
    EA_OnInitializeTechnologies =  
My.Resources.MyTechnology
```

```
End Function
```

EA_OnPreActivateTechnology

EA_OnPreActivateTechnology notifies **Add-Ins** that an MDG Technology resource is about to be activated in the model.

This event occurs when a user selects to activate an MDG Technology resource in the model (by clicking on the **Set Active button** on the 'MDG Technologies' dialog or by selecting the technology in the list box in the Default Tools toolbar).

The notification is provided immediately after the user attempts to activate the MDG Technology, so that the **Add-In** can permit or disable activation of the Technology.

Syntax

Function EA_OnPreActivateTechnology (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreActivateTechnology function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to

	retrieve model data and user interface status information.
Info	<p>EA.EventProperties</p> <p>Direction: IN</p> <p>Description: Contains these EventProperty objects for the MDG Technology to be activated:</p> <ul style="list-style-type: none">• TechnologyID: A string value corresponding to the MDG Technology ID

Return Value

- Return **True** to enable activation of the MDG Technology resource in the model
- Return **False** to disable activation of the MDG Technology resource

EA_OnPostActivateTechnology

EA_OnPostActivateTechnology notifies **Add-Ins** that an MDG Technology resource has been activated in the model.

This event occurs when a user activates an MDG Technology resource in the model (by clicking on the **Set Active button** on the 'MDG Technologies' dialog, or by selecting the technology in the list box in the Default Tools toolbar).

The notification is provided immediately after the user succeeds in activating the MDG Technology, so that the **Add-In** can update the Technology if necessary.

Syntax

Function EA_OnPostActivateTechnology (Repository As EA.Repository, Info As EA.EventProperties)

The EA_OnPostActivateTechnology function syntax contains these parameters:

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface

	status information.
Info	<p>EA.EventProperties</p> <p>Direction: IN</p> <p>Description: Contains these EventProperty objects for the MDG Technology to be activated:</p> <ul style="list-style-type: none">• TechnologyID: A string value corresponding to the MDG Technology ID

Return Value

Return **True** if the MDG Technology resource is updated during this notification. Return **False** otherwise.

EA_OnPreDeleteTechnology

Deprecated - refers to deleting a technology through the 'Resources' tab of the **Browser window**; this process is no longer recommended. See *Deploy An MDG Technology* for information on recommended methods for using technologies.

EA_OnPreDeleteTechnology notifies **Add-Ins** that an MDG Technology resource is about to be deleted from the model. This event occurs when a user deletes an MDG Technology resource from the model.

The notification is provided immediately after the user confirms their request to delete the MDG Technology, so that the **Add-In** can disable deletion of the MDG Technology.

Related Broadcast Events

Event
EA_OnInitializeTechnologies
EA_OnPreActivateTechnology
EA_OnPostActivateTechnology

Syntax

Function EA_OnPreDeleteTechnology (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreDeleteTechnology function syntax contains these elements:

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the MDG Technology to be deleted: <ul style="list-style-type: none">TechnologyID: A string value corresponding to the MDG Technology ID

Return Value

- Return **True** to enable deletion of the MDG Technology resource from the model
- Return **False** to disable deletion of the MDG Technology resource

EA_OnDeleteTechnology

Deprecated - refers to deleting a technology through the 'Resources' tab of the **Browser window**; this process is no longer recommended. See *Deploy An MDG Technology* for information of recommended methods for using technologies.

EA_OnDeleteTechnology notifies **Add-Ins** that an MDG Technology resource has been deleted from the model.

This event occurs after a user has deleted an MDG Technology resource from the model. Add-Ins that require an MDG Technology resource to be loaded can catch this event to disable certain functionality.

Related Events

Event
EA_OnInitializeTechnologies
EA_OnPreActivateTechnology
EA_OnPostActivateTechnology

Syntax

Sub EA_OnDeleteTechnology (Repository As EA.Repository, Info As EA.EventProperties)

The EA_OnDeleteTechnology function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects: <ul style="list-style-type: none">• TechnologyID: A string value corresponding to the MDG Technology ID

Return Value

None.

EA_OnImportTechnology

Deprecated - refers to importing a technology into the 'Resources' tab of the **Browser window**; this process is no longer recommended. See *Deploy An MDG Technology* for information of recommended methods for using technologies.

EA_OnImportTechnology notifies **Add-Ins** that you have imported an MDG Technology resource into the model.

This event occurs after you have imported an MDG Technology resource into the model. Add-Ins that require an MDG Technology resource to be loaded can catch this **Add-In** to enable certain functionality.

Related Events

Event
EA_OnInitializeTechnologies
EA_OnPreActivateTechnology
EA_OnPostActivateTechnology

Syntax

Sub EA_OnImportTechnology (Repository As EA.Repository, Info As EA.EventProperties)

The EA_OnImportTechnology function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects: <ul style="list-style-type: none">• TechnologyID: A string value corresponding to the MDG Technology ID

Return Value

None.

Custom Views

Enterprise Architect enables custom windows to be inserted as a tab within the **Diagram View** that appears at the center of the Enterprise Architect frame.

Creating a custom view helps you to easily display a custom interface within Enterprise Architect, alongside other diagrams and built-in views for quick and easy access.

Uses for this facility include:

- Reports and graphs showing summary data of the model
- Alternative views of a diagram
- Alternative views of the model
- Views of external data related to model data
- Documentation tools

Bear in mind that the 'Open Diagrams in Single Window' option in the 'Application Look' dialog will swap diagrams in the Diagram View rather than open more diagram tabs.

Create a Custom View

A custom view must be designed as an ActiveX Custom Control and inserted via the **Automation Interface**.

ActiveX Custom Controls can be created using most well-known programming tools, including Microsoft Visual Studio. See the documentation provided by the relevant vendor on how to create a custom control to produce an OCX file.

Once the custom control has been created and registered on the target system, it can be added through the AddTab() method of the Repository object. While it is possible to call AddTab() from any automation client, it is likely that you would call it from an **Add-In**, and that the Add-In is defined in the same OCX that provides the custom view.

This is a C# code example:

```
public class Addin
{
    UserControl1 m_MyControl;
    public void EA_Connect(EA.Repository Rep)
    {
    }
    public object EA_GetMenuItems(EA.Repository
Repository, string Location, string MenuName)
    {
        if(MenuName == "")
```

```
        return "-&C# Control Demo";
    else
    {
        String() ret = {"Show Custom View", "Show
Button"};
        return ret;
    }
}

public void EA_MenuClick(EA.Repository Rep,
string Location, string MenuName, string ItemName)
{
    if(ItemName == "Show Custom View")
        m_MyControl = (UserControl1)
Rep.AddTab("C# Demo","ContDemo.UserControl1");
    else if(ItemName == "Show Button")
        m_MyControl.ShowButton();
}
}
```

Custom Docked Window

Custom docked windows can be added into the Enterprise Architect user interface. Once added, they can be shown and docked in the same way as other built-in Enterprise Architect docked windows.

A custom docked window must be designed as an ActiveX Custom Control and inserted via the **Automation Interface**. ActiveX Custom Controls can be created using most well-known programming tools, including Microsoft Visual Studio. See the documentation provided by the relevant vendor on how to create a custom control to produce an OCX file.

Once the custom control has been created and registered on the target system, it can be added using the AddWindow() method of the Repository object. While it is possible to call AddWindow() from any automation client, it is likely that you would call it from an **Add-In**, and that the Add-In is defined in the same OCX that provides the custom view.

To view custom docked windows that have been added, select the 'Specialize > **Add-Ins** > Windows' ribbon option.

Custom docked windows can also be made visible by the automation client or Add-In using the ShowAddinWindow() method, or hidden by using the HideAddinWindow() method.

This is an example in C# code:

```
public class Addin
```

```
{
    UserControl1 m_MyControl;
    public void EA_Connect(EA.Repository Rep)
    {
        m_MyControl = (UserControl1)
Rep.AddWindow
        ("C# Demo","ContDemo.UserControl1");
    }
    public object EA_GetMenuItems(EA.Repository
Repository, string Location, string MenuName)
    {
        if(MenuName == "")
            return "-&C# Control Demo";
        else
        {
            String() ret = {"Show Window", "Show
Button"};
            return ret;
        }
    }
    public void EA_MenuClick(EA.Repository Rep,
string Location, string MenuName, string ItemName)
    {
        if(ItemName == "Show Window")
            Rep.ShowAddinWindow("C# Demo");
    }
}
```

```
        else if(ItemName == "Show Button")
            m_MyControl.ShowButton();
        }
    }
```

MDG Add-Ins

MDG Add-Ins are specialized types of **Add-In** that have additional features and extra requirements, for Add-In authors who want to contribute to Enterprise Architect's goal of Model Driven Generation.

One of the additional responsibilities of an MDG Add-In is to take ownership of a branch of an Enterprise Architect model, which is done through the MDG_Connect event.

Unlike general Add-In events, MDG Add-In events are only sent to the Add-In that has taken ownership of an Enterprise Architect model branch on a particular workstation.

MDG Add-Ins identify themselves as such during EA_Connect by returning the string 'MDG'.

Unlike ordinary Add-Ins, responding to MDG Add-In events is not optional, and methods must be published for each of the MDG Events.

MDG Events

An MDG **Add-In** must respond to all MDG Events. These events usually identify processes such as Build, Run, Synchronize, PreMerge and PostMerge, amongst others.

An MDG Link Add-In is expected to implement some form of forward and reverse engineering capability within Enterprise Architect, and as such requires access to a specific set of events, all to do with generation, synchronization and general processes concerned with converting models to code and code to models.

MDGAdd-In Events

Event
MDG_BuildProject
MDG_Connect
MDG_Disconnect
MDG_GetConnectedPackages
MDG_GetProperty
MDG_Merge

MDG_NewClass
MDG_PostGenerate
MDG_PostMerge
MDG_PreGenerate
MDG_PreMerge
MDG_PreReverse
MDG_RunExe
MDG_View

MDG_BuildProject

Add-Ins can use MDG_BuildProject to handle file changes caused by generation. This function is called in response to a user selecting the 'Execute > Source > Build > Build' ribbon option.

Respond to this event by compiling the project source files into a running application.

Syntax

Sub MDG_BuildProject (Repository As EA.Repository,
PackageGuid As String)

The MDG_BuildProject function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
PackageGuid	String

	<p>Direction: IN</p> <p>Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.</p>
--	--

Return Value

None.

MDG_Connect

An **Add-In** uses MDG_Connect to handle a user driven request to connect a model branch to an external application. The function is called when the user attempts to connect a particular Enterprise Architect Package to an as yet unspecified external project. The Add-In calls the event to interact with the user to specify such a project.

The Add-In is responsible for retaining the connection details, which should be stored on a per-user or per-workstation basis. That is, users who share a common Enterprise Architect model over a network should be able to connect and disconnect to external projects independently of one another.

The Add-In should therefore not store connection details in an Enterprise Architect repository. A suitable place to store such details would be:

SHGetFolderPath(..CSIDL_APPDATA..)\\AddinName

The PackageGuid parameter is the same identifier as is required for most events relating to the MDG Add-In.

Therefore it is recommended that the connection details be indexed using the PackageGuid value.

The PackageID parameter is provided to aid fast retrieval of Package details from Enterprise Architect, should this be required.

Syntax

Function MDG_Connect (Repository As EA.Repository, PackageID as Long, PackageGuid As String) As Long

The MDG_Connect function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
PackageID	Long Direction: IN Description: The PackageID of the Enterprise Architect Package the user has requested to have connected to an external project.
PackageGuid	String Direction: IN Description: The unique ID identifying the project provided by the Add-In when a connection to a project branch of an Enterprise Architect model was first

	established.
--	--------------

Return Value

- Return a non-zero to indicate that a connection has been made
- Return a zero to indicate that the user has not nominated a project and connection should not proceed

MDG_Disconnect

Add-Ins can use MDG_Disconnect to respond to user requests to disconnect the model branch from an external project.

This function is called when the user attempts to disconnect an associated external project. The **Add-In** is required to delete the details of the connection.

Syntax

Function MDG_Disconnect (Repository As EA.Repository, PackageGuid As String) As Long

The MDG_Disconnect function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
PackageGuid	String

	<p>Direction: IN</p> <p>Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.</p>
--	--

Return Value

- Return a non-zero to indicate that a disconnection has occurred, enabling Enterprise Architect to update the user interface
- Return a zero to indicate that the user has not disconnected from an external project

MDG_GetConnectedPackages

Add-Ins can use MDG_GetConnectedPackages to return a list of current connections between Enterprise Architect and an external application.

This function is called when the **Add-In** is first loaded, and is expected to return a list of the available connections to external projects for this Add-In.

Syntax

Function MDG_GetConnectedPackages (Repository As EA.Repository) As Variant

The MDG_GetConnectedPackages function syntax contains this parameter.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Return Value

Returns an array of GUID strings representing individual Enterprise Architect Packages.

MDG_GetProperty

MDG_GetProperty provides miscellaneous **Add-In** details to Enterprise Architect.

This function is called by Enterprise Architect to poll the Add-In for information relating to the PropertyName. This event should occur in as short a **duration** as possible, as Enterprise Architect does not cache the information provided by the function.

Values corresponding to these PropertyNames must be provided:

- IconID - Return the name of a DLL and a resource identifier in the format #ResID, where the resource ID indicates an icon
c:\program files\myapp\myapp.dll#101
- Language - Return the default language that Classes should be assigned when they are created in Enterprise Architect
- HiddenMenus - Return one or more values from the MDGMenus enumeration to hide menus that do not apply to your Add-In
if(PropertyName == "HiddenMenus")
return mgBuildProject + mgRun;

Syntax

Function MDG_GetProperty (Repository As EA.Repository,

PackageGuid As String, PropertyName As String) As Variant

The MDG_GetProperty function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In .
PropertyName	String Direction: IN Description: The name of the property that is used by Enterprise Architect. See the start of this topic for the possible values.

Return Value

See the start of this topic.

MDG_Merge

Add-Ins can use MDG_Merge to jointly handle changes to both the model branch and the code project that the model branch is connected to.

This event should be called whenever the user has asked to merge their model branch with its connected code project, or whenever the user has established a new connection to a code project.

The purpose of this event is to make the **Add-In** interact with the user to perform a merge between the model branch and the connected project.

Syntax

Function MDG_Merge (Repository As EA.Repository, PackageGuid As String, SynchObjects As Variant, SynchType As String, ExportObjects As Variant, ExportFiles As Variant, ImportFiles As Variant, IgnoreLocked As String, Language As String) As Long

The MDG_Merge function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object

	representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In .
SynchObjects	Variant Direction: OUT Description: A string array containing a list of objects (Object ID format) to be jointly synchronized between the model branch and the project. See <i>Object ID Format</i> for the format of the Object IDs.
SynchType	String Direction: OUT Description: The value determining the user-selected type of synchronization to take place. See <i>Synchronize Type</i> for a list of valid values.

ExportObjects	<p>Variant</p> <p>Direction: OUT</p> <p>Description: The string array containing the list of new model objects (in Object ID format) to be exported by Enterprise Architect to the code project.</p>
ExportFiles	<p>Variant</p> <p>Direction: OUT</p> <p>Description: A string array containing the list of files for each model object chosen for export by the Add-In.</p> <p>Each entry in this array must have a corresponding entry in the ExportObjects parameter at the same array index, so ExportFiles(2) must contain the filename of the object by ExportObjects(2).</p>
ImportFiles	<p>Variant</p> <p>Direction: OUT</p> <p>Description: A string array containing the list of code files made available to the code project to be newly imported to the model.</p> <p>Enterprise Architect imports each file listed in this array for import into the connected model branch.</p>

IgnoreLocked	String Direction: OUT Description: A value indicating whether to ignore any files locked by the code project (that is, ' True ' or ' False ').
Language	String Direction: OUT Description: The string value containing the name of the code language supported by the code project connected to the model branch.

Object ID Format

Each of the Object IDs listed in the 'SynchObjects' string arrays should have this format:

`(@namespace)*(#class)*($attribute|%operation|:property)*`

Return Value

- Return a non-zero if the merge operation completed successfully
- Return a zero when the operation has been unsuccessful

Merge

A merge consists of three major operations:

- Export: where newly created model objects are exported into code and made available to the code project
- Import: where newly created code objects, Classes and such things are imported into the model
- Synchronize: where objects available both to the model and in code are jointly updated to reflect changes made in either the model, code project or both

Synchronize Type

The Synchronize operation can take place in one of four different ways. Each of these ways corresponds to a value returned by 'SynchType':

- None: ('SynchType' = 0) No synchronization is to be performed
- Forward: ('SynchType' = 1) Forward synchronization, between the model branch and the code project is to occur
- Reverse: ('SynchType' = 2) Reverse synchronization, between the code project and the model branch is to occur
- Both: ('SynchType' = 3) Reverse, then Forward synchronizations are to occur

MDG_NewClass

Add-Ins can use MDG_NewClass to alter details of a Class before it is created.

This method is called when Enterprise Architect generates a new Class, and requires information relating to assigning the language and file path. The file path should be passed back as a return value and the language should be passed back via the language parameter.

Syntax

Function MDG_NewClass (Repository As EA.Repository, PackageGuid As String, CodeID As String, Language As String) As String

The MDG_NewClass function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In .
CodeID	String Direction: IN Description: A string used to identify the code element before it is created.
Language	String Direction: OUT Description: A string used to identify the programming language for the new Class. The language must be supported by Enterprise Architect.

Return Value

Return a string containing the file path that should be assigned to the Class.

MDG_PostGenerate

Add-Ins can use MDG_PostGenerate to handle file changes caused by generation.

This event is called after Enterprise Architect has prepared text to replace the existing contents of a file. Responding to this event enables the **Add-In** to write to the linked application's user interface rather than modify the file directly.

When the contents of a file are changed, Enterprise Architect passes FileContents as a non-empty string. New files created as a result of code generation are also sent through this mechanism, so the Add-Ins can add new files to the linked project's file list.

When new files are created Enterprise Architect passes FileContents as an empty string. When a non-zero is returned by this function, the Add-In has successfully written the contents of the file. A zero value for the return indicates to Enterprise Architect that the file must be saved.

Syntax

Function MDG_PostGenerate (Repository As EA.Repository, PackageGuid As String, FilePath As String, FileContents As String) As Long

The MDG_PostGenerate function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In .
FilePath	String Direction: IN Description: The path of the file Enterprise Architect intends to overwrite.
FileContents	String Direction: IN Description: A string containing the proposed contents of the file.

Return Value

The return value depends on the type of event that this function is responding to (see introduction). This function is required to handle two separate and distinct cases.

MDG_PostMerge

MDG_PostMerge is called by Enterprise Architect after a merge process has been completed.

File save checking should not be performed with this function, but should be handled by MDG_PreGenerate, MDG_PostGenerate and MDG_PreReverse.

Syntax

Function MDG_PostMerge (Repository As EA.Repository, PackageGuid As String) As Long

The MDG_PostMerge function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
PackageGuid	String Direction: IN

	Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In .
--	--

Return Value

- Return a non-zero to indicate that the post-merge has been successful
- Return a zero if the post-merge process has failed

Enterprise Architect assumes a non-zero return if this method is not implemented.

MDG_PreGenerate

Add-Ins can use MDG_PreGenerate to deal with unsaved changes.

This function is called immediately before Enterprise Architect attempts to generate files from the model. A possible use of this function would be to prompt the user to save unsaved source files.

Return Value

- Return a zero to abort generation
- Return any other value to enable the generation to continue

Syntax

Function MDG_PreGenerate (Repository As EA.Repository, PackageGuid As String) As Long

The MDG_PreGenerate function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN

	<p>Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</p>
PackageGuid	<p>String</p> <p>Direction: IN</p> <p>Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.</p>

MDG_PreMerge

MDG_PreMerge is called after a merge process has been initiated by the user and before Enterprise Architect performs the merge process.

This event is called after a user has performed their interactions with the merge screen and has confirmed the merge with the **OK button**, but before Enterprise Architect performs the merge process using the data provided by the MDG_Merge call, before any changes have been made to the model or the connected project.

This event is made available to provide the **Add-In** with the opportunity to generally set internal Add-In flags to augment the MDG_PreGenerate, MDG_PostGenerate and MDG_PreReverse events.

File save checking should not be performed with this function, but should be handled by MDG_PreGenerate, MDG_PostGenerate and MDG_PreReverse.

Syntax

Function MDG_PreMerge (Repository As EA.Repository, PackageGuid As String) As Long

The MDG_PreMerge function syntax contains these parameters.

Parameter	Type

Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In .

Return Value

- Return a zero to indicate that the merge process can not occur
- Return a non-zero if the merge process proceeds

If this method is not implemented then it is assumed that a merge process is used.

MDG_PreReverse

Add-Ins can use MDG_PreReverse to save file changes before they are imported into Enterprise Architect.

This function operates on a list of files that are about to be reverse-engineered into Enterprise Architect. If the user is working on unsaved versions of these files in an editor, you could either prompt the user or save automatically.

Syntax

Sub MDG_PreReverse (Repository As EA.Repository, PackageGuid As String, FilePaths As Variant)

The MDG_PreReverse function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
PackageGuid	String

	<p>Direction: IN</p> <p>Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.</p>
FilePaths	<p>String array</p> <p>Direction: IN</p> <p>Description: An array of filepaths pointed to the files that are to be reverse engineered.</p>

Return Value

None.

MDG_RunExe

Add-Ins can use MDG_RunExe to run the target application.

This function is called when the user selects the 'Execute > Run > Start > Run' ribbon option.

Respond to this event by launching the compiled application.

Syntax

Sub MDG_RunExe (Repository As EA.Repository,
PackageGuid As String)

The MDG_RunExe function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
PackageGuid	String

	<p>Direction: IN</p> <p>Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.</p>
--	--

Return Value

None.

MDG_View

Add-Ins can use MDG_View to display user specified code elements.

This function is called by Enterprise Architect when the user asks to view a particular code element. The **Add-In** can then present that element in its own way, usually in a code editor.

Syntax

Function MDG_View (Repository As EA.Repository,
PackageGuid As String, CodeID as String) As Long

The MDG_View function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
PackageGuid	String Direction: IN Description: The GUID identifying the

	Enterprise Architect Package sub-tree that is controlled by the Add-In .
CodeID	<p>String</p> <p>Direction: IN</p> <p>Description: Identifies the code element in this format:</p> <p><type>ElementPart<type>ElementPart... where each element is proceeded with a token identifying its type:</p> <ul style="list-style-type: none">@ - namespace# - Class\$ - attribute% - operation <p>For example, if a user has selected the m_Name attribute of Class1 located in namespace Name1, the Class ID would be passed through in this format:</p> <p>@Name1#Class1%m_Name</p>

Return Value

- Return a non-zero value to indicate that the **Add-In** has processed the request

- Return a zero value for Enterprise Architect to employ the standard viewing process, which is to launch the associated source file

Workflow Add-In Events

Enterprise Architect provides this set of four additional events that are sent only to workflow **Add-Ins**.

Workflow Add-In Events

Event
<p>EA_AllowPropertyUpdate</p> <p>This event is sent to workflow Add-Ins after a user has changed a built-in property value.</p>
<p>EA_AllowTagUpdate</p> <p>This event is sent to workflow Add-Ins after a user has changed a Tagged Value.</p>
<p>EA_CanEditProperty</p> <p>This event is sent to workflow Add-Ins when a property is being displayed that allows the property to block all edits.</p>
<p>EA_CanEditTag</p> <p>This event is sent to workflow Add-Ins when a Tagged Value is being displayed that allows the property to block all edits.</p>

EA_AllowPropertyUpdate

This event is sent to workflow **Add-Ins** after a user has changed a built-in property value.

Syntax

Function EA_AllowPropertyUpdate (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects describing the requested property update: <ul style="list-style-type: none">• Type: A string value corresponding to Element.Type

	<ul style="list-style-type: none">• Stereotype: A string value corresponding to <code>Element.Stereotype</code>• PropertyName: The name of the property field to enable or disable• OldValue: The previous value of the property• NewValue: The new value of the property
--	--

Return Value

- Return **False** to prevent this change to the described property
- Return **True** to allow this change

EA_AllowTagUpdate

This event is sent to Workflow **Add-Ins** after a user has changed a Tagged Value.

Syntax

Function EA_AllowTagUpdate (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects describing the requested Tagged Value update: <ul style="list-style-type: none">• Type: A string value corresponding to Element.Type

	<ul style="list-style-type: none">• Stereotype: A string value corresponding to <code>Element.Stereotype</code>• TagName: The name of the Tagged Value field to enable or disable• OldValue: The previous value of the tag• NewValue: The new value of the tag
--	---

Return Value

- Return **False** to prevent this change to the described Tagged Value
- Return **True** to allow this change

EA_CanEditProperty

This event is sent to Workflow **Add-Ins** when a property is being displayed that allows the property to block all edits.

Syntax

Function EA_CanEditProperty (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects describing the property: <ul style="list-style-type: none">• Type: A string value corresponding to Element.Type

	<ul style="list-style-type: none">• Stereotype: A string value corresponding to <code>Element.Stereotype</code>• PropertyName: The name of the property field to enable or disable
--	---

Return Value

- Return **False** to prevent all edits to the described property
- Return **True** to allow changes

EA_CanEditTag

This event is sent to Workflow **Add-Ins** when a Tagged Value is being displayed that allows the property to block all edits.

Syntax

Function EA_CanEditTag (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects describing the Tagged Value: <ul style="list-style-type: none">• Type: A string value corresponding to

	<p>Element.Type</p> <ul style="list-style-type: none">• Stereotype: A string value corresponding to Element.Stereotype• TagName: The name of the tag to enable or disable
--	--

Return Value

- Return **False** to prevent all edits to the described Tagged Value
- Return **True** to allow changes

