



Enterprise Architect

User Guide Series

Automation

Author: Sparx Systems

Date: 30/06/2017

Version: 1.0

Table of Contents

Automation	8
Hybrid Scripting	10
C# Example	12
Java Example	14
Scripting	16
Scripts Tab	18
Console Tab	20
Script Group Properties	22
Script Editor	24
Session Object	27
Script Debugging	28
Enterprise Architect Object Model	29
Using the Automation Interface	30
Connect to the Interface	31
Set References In Visual Basic	34
Examples and Tips	35
Call from Enterprise Architect	37
Available Resources	39
Reference	40
Interface Overview	41
App Object	43
Enumerations	44
ConstLayoutStyles	46
CreateBaselineFlag	47
CreateModelType	48
DocumentBreak	49
DocumentPageOrientation	50
DocumentType	51
EAEditionTypes	52
EnumRelationSetType	53
ExportPackageXMIFlag	54
MDGMenus	55
MessageFlag	56
ObjectType	57
PropType	59
ReloadType	60
ScenarioDiagramType	61
ScenarioStepType	62
ScenarioTestType	63
XMIType	64
Repository Package	65
Author Class	66
Client Class	67
Collection Class	69
The AddNew Function	71
Datatype Class	75
EventProperties Class	78

EventProperty Class	79
ModelWatcher Class	80
Package Class	81
ProjectIssues Class	90
ProjectResource Class	92
ProjectRole Class	94
PropertyType Class	96
Reference Class	98
Repository Class	100
Stereotype Class	119
Task Class	121
Term Class	123
Element Package	125
Constraint Class	127
Effort Class	129
Element Class	131
File Class	145
Issue (Maintenance) Class	147
Metric Class	149
Requirement Class	151
Resource Class	153
Risk Class	155
Scenario Class	157
ScenarioExtension Class	159
ScenarioStep Class	161
TaggedValue Class	163
Test Class	165
Element Features Package	167
Attribute Class	168
AttributeConstraint Class	173
AttributeTag Class	175
CustomProperties Collection	177
EmbeddedElements Collection	178
Method Class	179
MethodConstraint Class	183
MethodTag Class	185
Parameter Class	187
ParamTag Class	190
Partitions Collection	192
Properties Class	193
TemplateParameter Class	195
Transitions Collection	197
Connector Package	198
Connector Class	199
ConnectorConstraint Class	205
ConnectorEnd Class	207
ConnectorTag Class	210
RoleTag Class	212
TemplateBinding Class	214
Diagram Package	216
Diagram Class	217

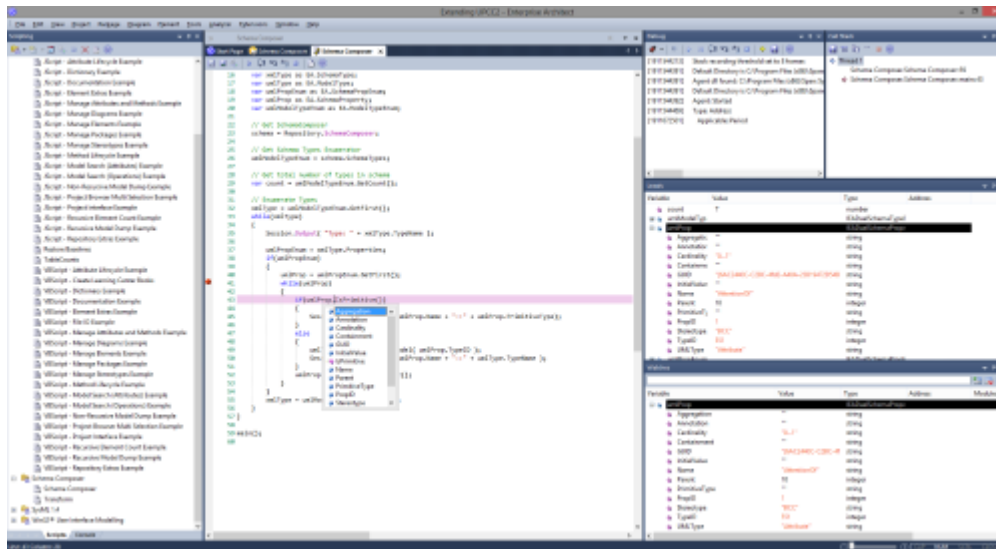
DiagramLinks Class	225
DiagramObject Class	228
SwimlaneDef Class	234
Swimlanes Class	236
Swimlane Class	238
Project Interface Package	239
Project Class	240
Document Generator Interface Package	256
DocumentGenerator Class	257
Mail Interface Package	262
MailInterface Class	263
Simulation Package	266
Simulation Class	267
Schema Composer Package	269
SchemaProperty Class	270
SchemaComposer Class	272
SchemaProfile Class	274
ModelType Class	275
ModelTypeEnum Class	277
SchemaType Class	278
SchemaTypeEnum Class	279
SchemaPropEnum Class	280
SearchType Enumeration	281
Code Samples	282
Open the Repository	283
Iterate Through a .EAP File	284
Add and Manage Packages	285
Add and Manage Elements	287
Add a Connector	288
Add and Manage Diagrams	290
Add and Delete Features	291
Element Extras	292
Repository Extras	296
Stereotypes	299
Work With Attributes	300
Work With Methods	302
Enterprise Architect Add-In Model	304
The Add-In Manager	305
Add-In Tasks	306
Create Add-Ins	307
Define Menu Items	308
Deploy Add-Ins	310
Tricks and Traps	312
Add-In Search	314
XML Format (Search Data)	315
Add-In Events	317
EA_Connect	318
EA_Disconnect	319
EA_GetMenuItems	320
EA_GetMenuState	321
EA_GetRibbonCategory	323

EA_MenuClick	324
EA_OnOutputItemClicked	326
EA_OnOutputItemDoubleClicked	327
EA_ShowHelp	328
Broadcast Events	329
Schema Composer Broadcasts	331
EA_GenerateFromSchema	332
EA_GetProfileInfo	333
EA_IsSchemaExporter	334
Add-In License Management Events	335
EA_AddinLicenseValidate	336
EA_AddinLicenseGetDescription	337
EA_GetSharedAddinName	338
Compartment Events	340
EA_QueryAvailableCompartments	341
EA_GetCompartmentData	343
Context Item Events	346
EA_OnContextItemChanged	347
EA_OnContextItemDoubleClicked	348
EA_OnNotifyContextItemModified	349
EA_FileClose	350
EA_FileNew	351
EA_FileOpen	352
EA_OnPostCloseDiagram	353
EA_OnPostInitialized	354
EA_OnPostOpenDiagram	355
EA_OnPostTransform	356
EA_OnPreExitInstance	357
EA_OnRetrieveModelTemplate	358
EA_OnTabChanged	360
Model Validation Broadcasts	361
EA_OnInitializeUserRules	362
EA_OnStartValidation	363
EA_OnEndValidation	364
EA_OnRunElementRule	365
EA_OnRunPackageRule	366
EA_OnRunDiagramRule	367
EA_OnRunConnectorRule	368
EA_OnRunAttributeRule	369
EA_OnRunMethodRule	370
EA_OnRunParameterRule	371
Model Validation Example	372
Post-New Events	378
EA_OnPostNewElement	379
EA_OnPostNewConnector	380
EA_OnPostNewDiagram	381
EA_OnPostNewDiagramObject	382
EA_OnPostNewAttribute	383
EA_OnPostNewMethod	384
EA_OnPostNewPackage	385
EA_OnPostNewGlossaryTerm	386

Pre-Deletion Events	387
EA_OnPreDeleteElement	388
EA_OnPreDeleteAttribute	389
EA_OnPreDeleteMethod	390
EA_OnPreDeleteConnector	391
EA_OnPreDeleteDiagram	392
EA_OnPreDeleteDiagramObject	393
EA_OnPreDeletePackage	394
EA_OnPreDeleteGlossaryTerm	395
Pre New-Object Events	396
EA_OnPreNewElement	397
EA_OnPreNewConnector	398
EA_OnPreNewDiagram	399
EA_OnPreNewDiagramObject	400
EA_OnPreDropFromTree	401
EA_OnPreNewAttribute	402
EA_OnPreNewMethod	403
EA_OnPreNewPackage	404
EA_OnPreNewGlossaryTerm	405
Tagged Value Broadcasts	406
EA_OnAttributeTagEdit	407
EA_OnConnectorTagEdit	408
EA_OnElementTagEdit	409
EA_OnMethodTagEdit	410
Technology Events	411
EA_OnInitializeTechnologies	412
EA_OnPreActivateTechnology	413
EA_OnPostActivateTechnology	414
EA_OnPreDeleteTechnology	415
EA_OnDeleteTechnology	416
EA_OnImportTechnology	417
Custom Views	418
Create a Custom View	419
Add a Portal	420
Custom Docked Window	421
MDG Add-Ins	422
MDG Events	423
MDG_Build Project	424
MDG_Connect	425
MDG_Disconnect	426
MDG_GetConnectedPackages	427
MDG_GetProperty	428
MDG_Merge	429
MDG_NewClass	432
MDG_PostGenerate	433
MDG_PostMerge	434
MDG_PreGenerate	435
MDG_PreMerge	436
MDG_PreReverse	437
MDG_RunExe	438
MDG_View	439

Workflow Scripts	440
Workflow Script Functions	442
Functions - Validate and Control User Input	443
Functions - Create a Search With User Tasks	445
Filled Workflow Data Structures	446
Workflow Data Structures You Fill	448
Functions You Call	449




Automation





Enterprise Architect has a formidable set of built-in features for working with models, but it also provides a range of environments for accessing and manipulating the contents of a repository programmatically. This is an extremely powerful facility that gives you unlimited ability to query and manipulate models, add to the Enterprise Architect user interface, generate reports, and even create support for new modeling languages. The Automation Interface gives you access to the Object Model, which is an easy to use and well defined set of objects with properties and methods that can be used to query and manipulate the repository and its contents, shielding the programmer from having to know the underlying repository data structures.

The automation interface is available from a scripting framework built into the Enterprise Architect user interface, through external scripting environments, or through Add-Ins that can be built in a wide range of programming languages.

Facilities

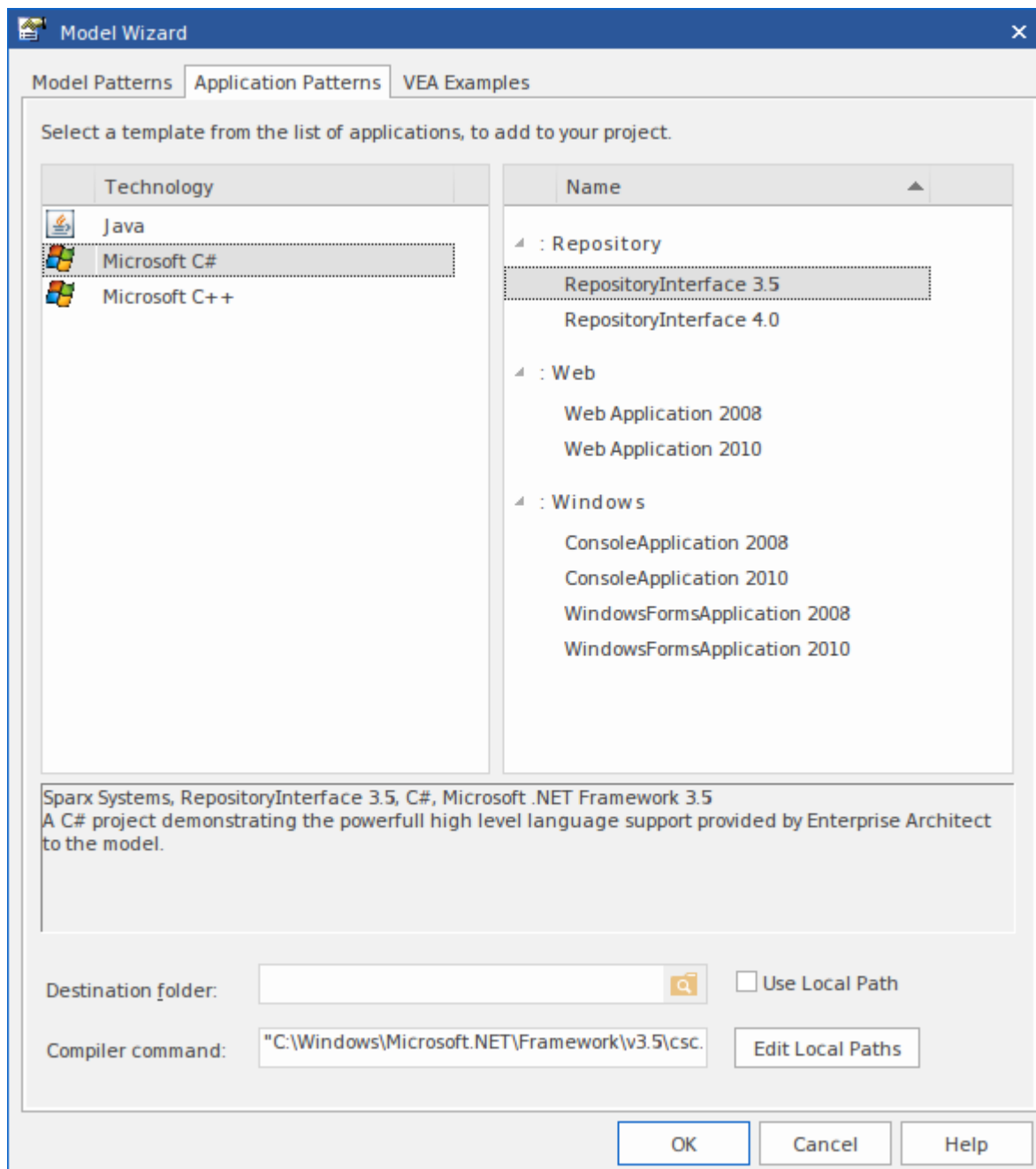
Facility	Description
 <p>Scripting</p>	Learn about the flexible and easy-to-use scripting capability to programmatically inspect and/or modify elements within your currently open model.
 <p>Object Model</p>	Discover the Enterprise Architect Object model. Write your own custom programs that access the information stored in Enterprise Architect.
 <p>Add-In Model</p>	The Enterprise Architect Add-In model helps you build on the features provided by the Automation Interface to enable you to extend the Enterprise Architect user interface.
MDG Add-Ins	MDG Add-Ins are specialized types of Add-Ins that have additional features and extra requirements. MDG Add-Ins are focused on generation, synchronization and general processes concerned with converting models to code and code to models.

	
<p data-bbox="220 331 421 389">Code Samples and Reference</p> 	<p data-bbox="518 331 1385 362">Access the wealth of knowledge and samples to help you complete your Add-In.</p>

Hybrid Scripting

Hybrid scripting extends the capabilities of the standard scripting environment to high level languages such as Java and C#. Hybrid scripting provides a speed advantage over conventional scripting, and also allows script authors to leverage existing skills in popular programming languages.

Access



Ribbon	Design > Model Wizard > Application Patterns
Context Menu	Right-click on Package Add a Model using Wizard Application Patterns

Keyboard Shortcuts	Ctrl+Shift+M Application Patterns
Other	Project Browser caption bar menu New Model from Pattern Application Patterns

Features

- Superior execution speed
- Enhanced interoperability
- Full Visual Execution Analyzer support

C# Example

This sample program demonstrates how easy it is to navigate, query and report on the current model using any Microsoft .NET language. This example is written in C#.

When run, it will print the names of every Package in the model you are currently using.

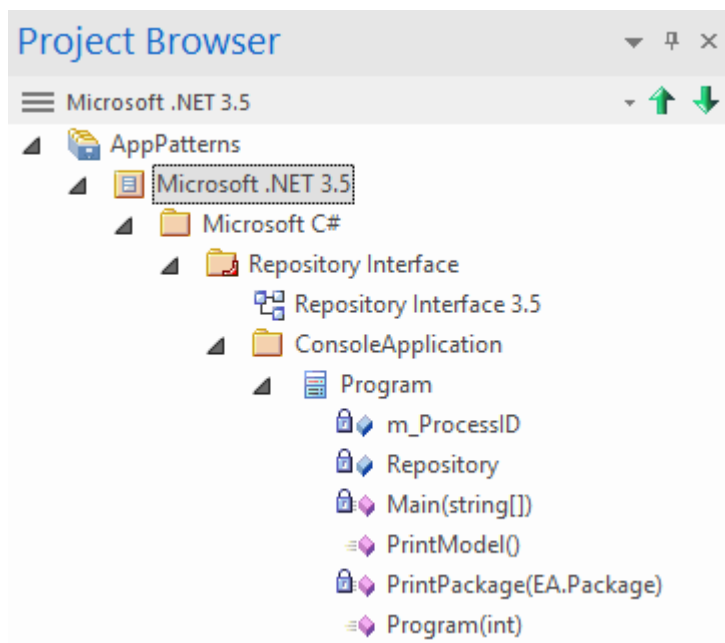
Create the project

In the Project Browser, select the Package in which to create the template and then use the ribbon or context menu to display the Model Wizard; select the 'Application Patterns' tab.

On the 'Application Patterns' tab, select the *Microsoft C# > RepositoryInterface* template. (You can choose from either the 3.5 or the 4.0 framework versions.) Specify the destination folder on the file system where the project template will be created, and click on the OK button.

Open the project

A Package structure similar to this will be created for you.



Expand the structure until you locate the *Repository Interface n.n* diagram and open it.

Overview:

This sample program demonstrates how easy it is to navigate, query and report on the current model using any Microsoft .NET language. This example is written in C#. When run, it will print the names of every Package in the model you are currently using.

Framework:


The build uses the C# compiler from the Microsoft .NET framework.

Version:


4.0

Note:


The links on the right operate on the active Analyzer Script. To use these links make sure you have selected the 'Repository Interface 4.0' script. You can use the Analyzer Scripts link below to do this.

 [Analyzer Scripts](#)


Build the project

 [Build](#)

Run your program

 [Run](#)

Debug the program

 [*DebugRun](#)

Program
- m_ProcessID: int = 0
- Repository: EA.Repository = null
- Main(string[]): void
+ PrintModel(): bool
- PrintPackage(EA.Package): void
+ Program(int)

Build the script

The commands on this diagram will operate on the active build configuration. Before executing them, double-click on the *Analyzer Scripts* link and select the checkbox next to the Repository Interface build configuration.

Run the script

Double-click on the *Run* link to open the Console. The Console will pause after completion so you can read the output from the program; this output will also be sent to the 'Script' tab of the System Output window. You can alter this by changing the code.

Debug the script

Select the 'Program' Class from the Project Browser and press Ctrl+E to open the source code.

Place a breakpoint in one of the functions and then double-click on the *DebugRun* link. When the breakpoint is encountered, the line of code will become highlighted in the editor, as shown:

ConsoleApplication

- Program
 - Repository
 - m_ProcessID
 - Main()
 - PrintModel()
 - PrintPackage()
 - Program()
 - Trace()

```

21      Console.WriteLine(msg);
22    }
23    public Program(int pid)
24    {
25        m_ProcessID = pid;
26        Repository = SparxSystems.Services.GetRepository(m_ProcessID);
27        Trace("Running C# Console Application AppPattern .NET 3.5");
28    }
29    private void PrintPackage(EA.Package package)
30    {
31        Trace(package.Name);
32        EA.Collection packages = package.Packages;
33        for (short ip = 0; ip < packages.Count; ip++)
34        {
35            EA.Package child = (EA.Package)packages.GetAt(ip);
36            PrintPackage(child);
37        }
38    }

```

Java Example

This sample program demonstrates how easy it is to navigate, query and report on the current model using a high-level language such as Java.

When run, it will print the names of every Package in the currently-loaded model.

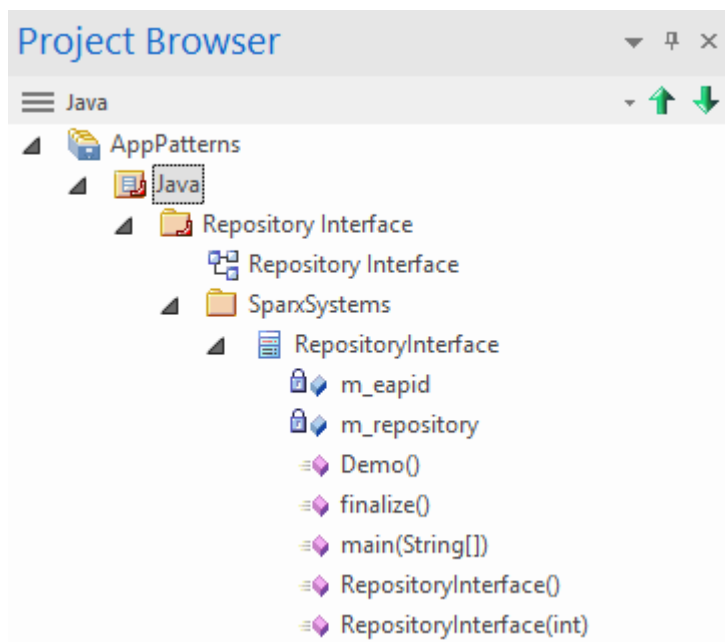
Create the project

In the Project Browser, select the Package in which to create the template, then use the ribbon or context menu to display the Model Wizard; click on the 'Application Patterns' tab.

From this tab, select the *Java > RepositoryInterface* template. Specify the destination folder on the file system in which the project template will be created, and click on the OK button.

Open the project

A Package structure similar to this will be created for you.



Expand the structure until you locate the 'Repository Interface' diagram and open it.

Overview:
This sample program demonstrates how easy it is to navigate, query and report on the current model using a high level language such as Java. When run, it will print the names of every Package in the currently loaded model.

Framework:
The build uses the compiler from the Java JDK 1.7 x86 framework.

Version:
1.7

Note:
In order to use the Build, Run and Debug links, you must first locate the 'Repository Interface' Analyzer Script generated by the wizard, and make it the active script for the model.
You can use the 'Analyzer Scripts' link to do this.

[Analyzer Scripts](#)

Build the project

[Build](#)

Run your program

[Run](#)

Debug the program

[*DebugRun](#)

RepositoryInterface	
-	m_eapid: int = 0
-	m_repository: org.sparx.Repository = null
+	Demo(): void
+	finalize(): void
~	main(String[]): void
+	RepositoryInterface()
+	RepositoryInterface(int)

Build the script

The commands on the diagram will operate on the active build configuration. Before executing them, double-click on the *Analyzer Scripts* link and select the checkbox next to the Repository Interface build configuration.

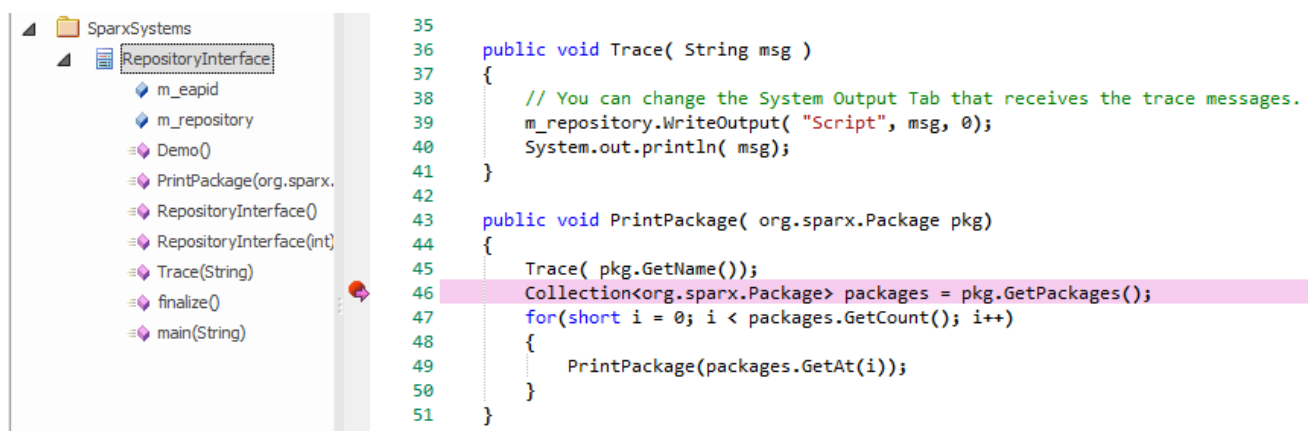
Run the script

Double-click on the *Run* link; a Console will open. The Console will pause after completion so you can read the output. The output from the program will also be output to the 'Script' tab of the System Output window. You can alter this by changing the code.

Debug the script

Select the 'Program' Class from the Project Browser and press Ctrl+E to open the source code.

Place a breakpoint in one of the functions and then double-click on the *DebugRun* link. When the breakpoint is encountered the line of code will become highlighted in the editor, as shown.



Scripting



Enterprise Architect's scripting environment is a flexible and easy to use facility that supports both Javascript and the Microsoft scripting languages JScript and VBScript. When any script runs, it has access to a built in 'Repository' object. Using this script object you can programmatically inspect and/or modify elements within your currently open model. Enterprise Architect also provides feature rich editors, tools to run, debug and manage your scripts. Scripts are modular and can include other scripts by name using the *!include* directive. They can be used for a broad range of purposes from documentation to validation and refactoring, they can be of enormous help with automating time consuming tasks.

Script Engine Support

- Mozilla SpiderMonkey [version 1.8]
- Microsoft Scripting Engine

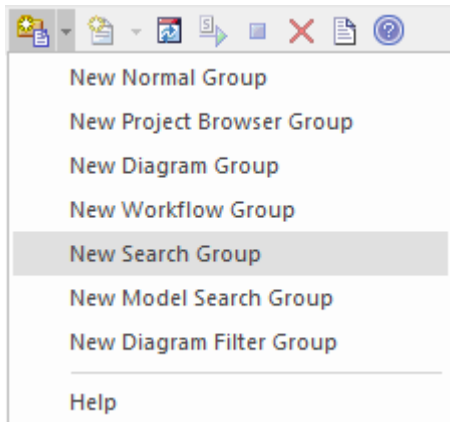
Script Languages

- JavaScript
- JScript
- VBScript

Benefits

- Inspecting and reporting on model and element composition
- Modifying and updating element properties
- Running queries to obtain extended model information
- Modifying diagram layouts
- Being called from report document templates to populate reports
- Creating and implementing process workflows
- Being included in MDG Technologies to augment domain specific languages
- Extensive UI access to scripts through context menus
- Automation Server role for in-process and out-of-process COM clients (Scripting is itself an example of an in-process client; Add-ins are another)
- Element access governance through Workflow security
- Model Search integration

Script Groups



Scripts are managed and contained in groups. Each group has an attribute called 'Type'. This attribute is used to help Enterprise Architect decide how and where the script can be used and from which features it should be made available. The properties of a script group can be viewed from its shortcut menu.

Script Storage

Built in scripts are file based and are installed with Enterprise Architect. They appear under the Local Scripts group.

You cannot edit or delete Local scripts, but you can copy the contents easily enough.

User defined scripts are model based and as such, can be shared by a community. They are listed in the group to which they belong..

Using Scripts

The management interface for Scripting is the Scripting window, which contains the:

- Script Tree View ('Scripts' tab), which you use to review, create and edit scripts
- Script Console ('Console' tab), which you use to operate on an executing script

Other than the Local Scripts, which are file based and installed with Enterprise Architect, all other scripts are stored as model assets and can be shared with its users. Script debuggers can help you with script development and script editors can provide you with information on the automation interfaces available to you. Analyze the execution; for example, by recording a Sequence diagram of the script execution, and halting execution to view local variables.

Notes

- This facility is available in the Corporate, Business and Software Engineering, Systems Engineering and Ultimate editions
- If you intend to use the Scripting facility under Crossover/WINE, you must also install Internet Explorer version 6.0 or above

Scripts Tab

The 'Scripts' tab is composed of a toolbar and a view of all scripts according to group. The script groups and their scripts also have context menus that provide some or all of these options:

- 'Group Properties' - to display or edit script group properties in the 'Script Group Properties' dialog
- 'Run Script' - to execute the selected script (or press Ctrl while you double-click on the script name)
- 'Edit Script' - to update the selected script (or double-click on the script name to display the 'Script Editor', which usually displays a script template, determined by the user group type as assigned on creation or on the 'Script Group Properties' dialog)
- 'Rename Script' - to change the name of the selected group or script
- 'New VBScript/JScript/JavaScript' - add a new script to the selected user group
- 'Import Workflow Script' - to display the 'Browser' dialog through which you locate and select a workflow script source (.vbs) file to import into the Workflow script folder
- 'Delete Group/Script' - to delete the selected user group or script









You can also move or copy a script from one user scripts folder to another; to:








- Move a script, highlight it in the 'Scripts' tab and drag it into the user scripts folder it now belongs to
- Copy a script, highlight it in the 'Scripts' tab and press Ctrl while you drag it into the user scripts folder in which to duplicate it

Access

Ribbon	Code > Tools > Scripting > Scripts
--------	------------------------------------

Script Toolbar

Icon	Action
	<p>Create a new script group; this option displays a short menu of the types of script group you can create, namely:</p> <ul style="list-style-type: none"> • Normal Group () • Project Browser Group () • Diagram Group () • Workflow Group () • Search Group () • Model Search Group <p>The new group is added to the end of the list in the Scripting window, with the 'New group' text highlighted so that you can type in the group name.</p>
	<p>Create a new script file in the selected script group; displays a short menu of the types of script you can create, namely:</p> <ul style="list-style-type: none"> • VBScript ()

	<ul style="list-style-type: none"> • JScript () • JavaScript () <p>The new script is added to the end of the list in the selected group, with the 'New script' text highlighted so that you can type in the script name.</p>
	Refresh the script tree in the Scripting window; this icon also reloads any changes made to a workflow script.
	<p>Compile and execute the selected script.</p> <p>The output from the script is written to the 'Script' tab of the System Output window, which you display using the View Script Output button.</p>
	Stop an executing script; the icon is disabled if no script is executing.
	<p>Delete a script from the model; you cannot use this icon to delete a script group (see the earlier 'Context Menu' item), scripts in the 'Local Scripts' group, or a script that is executing.</p> <p>The system prompts you to confirm the deletion only if the 'Confirm Deletes' checkbox is selected in the 'Project Browser' panel of the 'General' page of the 'Preferences' dialog; if this option is not selected, no prompt is displayed.</p> <p>Script deletion is permanent - scripts cannot be recovered.</p>
	Display the System Output window with the results of the most recently executed script displayed in the 'Script' tab.

Notes

- This facility is available in the Corporate, Business and Software Engineering, Systems Engineering and Ultimate editions
- If you add, delete or change a script, you might have to reload the model in order for the changes to take effect
- If you select to delete a script group that contains scripts, the system always prompts you to confirm the action regardless of any system settings for delete operations; be certain that you intend to delete the group and its scripts before confirming the deletion - deletion of script groups and scripts is permanent

Console Tab

The script console is a tab of the Scripting window; it is a command line interpreter through which you can quickly enable a script engine and enter commands to act on the script.

You type the commands in the field at the bottom of the tab; when you press the Enter key, the script console executes the commands and displays any output immediately.

You can input two types of command:

- Console commands
- Script commands

Access

Ribbon	Code > Tools > Scripting > Console
--------	------------------------------------

Console Commands

Console commands are preceded by the ! character and instruct the console to perform an action.

The available console commands are provided here; to list these commands on the 'Console' tab itself, type ? in the console field (without the preceding ! character) and press the Enter key.

- c(lear) - clears the console display
- sa(ve) - saves the console display to a file
- h(elp) - prints a list of commands, as for ?
- VB - opens a VBScript console
- JA - opens a JavaScript console
- JS - opens a JScript console
- st(op) - closes any script running console
- i(nclude) name - executes the named script item; name is of the format GroupName.ScriptName (spaces are allowed in names)
- ? - (without the !) lists commands
- ?name - Outputs the value of a variable name (only if a script console is opened).

Script Commands

A script command is script code that depends on the script engine. Script commands can be executed only once a script console has been created.

Examples:

These lines, entered into the console, create a VBScript console and then execute the script 'MyScript' in the user group 'MyGroup':

```
>!VB
```

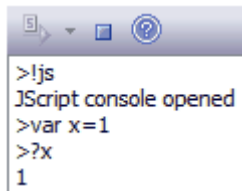
```
>!i MyGroup.MyScript
```

These lines, entered into the console, create a JScript console and then create a variable called x with the value 1:

```
>!JS
```



```
>var x = 1
```

This image shows the result of entering the above JScript example; remember that you can use ?<variable name> to get the current value of any item you have created during the console session.



Console Tab Toolbar

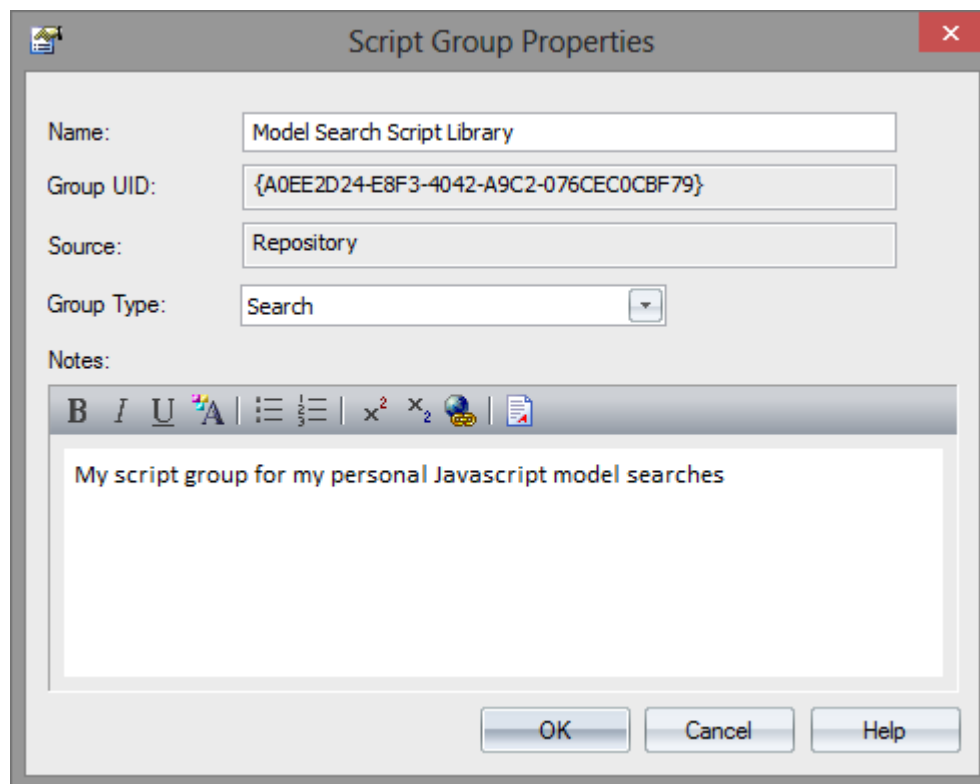
The 'Console' tab has two operations available through the toolbar:

- Open Console () - click on the down-arrow and select to open a VBScript console, JScript console or JavaScript console
- Stop Script () - click to stop an executing script and close the current console

Notes

- This facility is available in the Corporate, Business and Software Engineering, Systems Engineering and Ultimate editions
- You can save the output of the console to an external .txt file; right-click on the console window, select the 'Save As' option, browse for an appropriate file location and specify the file name

Script Group Properties









When you create a script you develop it within a script group, the properties of which determine how that script is to be made available to the user - through the Project Browser context menu to operate on objects of a specific type, or through a diagram context menu. You create a Script Group using the first icon on the 'Scripts' tab toolbar.

Access

Ribbon	Code > Tools > Scripting > Scripts > right-click on [Group name] > Group Properties
--------	---

Define the Script Group Properties

Field/Button	Action
Name	Type in the name of the script group.
Group UID	(Read only) The automatically assigned GUID for the group.
Source	(Read only) The location of the template used to create the script.
Group Type	Click on the drop-down arrow and select the type of script contained in the group;

	<p>this can be one of:</p> <ul style="list-style-type: none"> • Normal - () General model scripts • Project Browser - () Scripts that are listed in and can be executed from the Project Browser 'Scripts' context menu option • Workflow - () Scripts executed by Enterprise Architect's workflow engine; you can create only VB scripts of this type • Search - () Scripts that can be executed as model searches; these scripts are listed in the 'Search' field of the Model Search window, in the last category in the list • Diagram - () Scripts that can be executed from the 'Scripts' submenu of the diagram context menu • Find in Project - () Scripts that can be executed from the 'Scripts' submenu of a context menu within the Model Search view, on the results of a successfully-executed SQL search that includes CLASSGUID and CLASSTYPE, or a Query-built search • Element - Scripts that can be executed from the 'Scripts' submenu of element context menus; accessible from the Project Browser, Diagram, Model Search, Element List, Package Browser and Gantt views • Package - Scripts that can be executed from the 'Scripts' submenu of Package context menus; accessible from the Project Browser • Diagram - Scripts that can be executed from the 'Scripts' context menu option for diagrams; accessible from the Project Browser and diagrams • Link - Scripts that can be executed from the 'Scripts' context menu option for connectors; accessible from diagrams
Notes	Type in any comments you need regarding this script group.

Script Editor

Using the Script Editor you can perform a number of operations on an open script file, such as:

- Save changes to the current script
- Save the current script under a different name
- Run the script
- Debug the script
- Stop the executing script
- View the script output in the 'Scripts' tab of the System Output window

The editor is based on, and provides the facilities of, the common Code Editor in the application work area.

Access

Ribbon	Code > Tools > Scripting > Scripts > right-click on [script name] > Edit Script or Code > Tools > Scripting > Scripts > double-click on [script name]
--------	--

Facilities

Facility	Detail
Scripting Objects	<p>Enterprise Architect adds to the available functionality and features of the editor script language by providing inbuilt objects; these are either Type Libraries providing Intelli-sense for editing purposes, or Runtime objects providing access to objects of the types described in the Type Libraries.</p> <p>The available Intelli-sense scripting objects are:</p> <ul style="list-style-type: none"> • EA • MathLib • System <p>The runtime scripting objects are:</p> <ul style="list-style-type: none"> • Repository (Type: IDualRepository, an instance of EA.Repository, the Enterprise Architect Automation Interface) • Maths (Type: IMath, an instance of MathLib; this exposes functions from the Cephes mathematical library for use in scripts) • Session (Type: ISession, an instance of System)
Script Editing Intelli-sense (Required Syntax)	<p>Intelli-sense is available not only in the 'Script Editor', but also in the 'Script Console'; Intelli-sense at its most basic is presented for the inbuilt functionality of the script engine.</p> <p>For Intelli-sense on the additional Enterprise Architect scripting objects (as listed) you must declare variables according to syntax that specifies a type; it is not necessary to use this syntax to execute a script properly, it is only present so that the correct Intelli-sense can be displayed for an item.</p> <p>The syntax can be seen in, for example:</p>

	<p>Dim e as EA.Element</p> <p>Then when you type, in this case, e., the editor displays a list of member functions and properties of e's type.</p> <p>You select one of these to complete the line of script; you might, therefore, type:</p> <p>VBTrace(e.</p> <p>As you type the period, the editor presents the appropriate list and you might double-click on, for example, Abstract; this is inserted in the line, and you continue to type or select the rest of the statement, in this case adding the end space and parenthesis:</p> <p>VBTrace(e.Abstract)</p>
Keystrokes	<p>In the Script Editor or Console, Intelli-sense is presented on these keystrokes.</p> <ul style="list-style-type: none"> • Press . (period) after an item to list any members for that item's type • Press Ctrl+Spacebar on a word to list any Intelli-sense items with a name starting with the string at the point the keys were pressed • Press Ctrl+Spacebar when not on a word to display any available top level Intelli-sense items - these are the Intelli-sense objects described above plus any built-in methods and properties of the current scripting language
Include script libraries	<p>An <i>Include</i> statement (!INC) allows a script to reference constants, functions and variables defined by another script accessible within the Scripting Window. Include statements are typically used at the beginning of a script.</p> <p>To include a script library, use this syntax:</p> <p>!INC [Script Group Name].[Script Name]</p> <p>For example:</p> <p>!INC Local Scripts.EAConstants-VBScript</p>
Using Inbuilt Math Functions	<p>Various mathematical functions are available within the Script Editor, through the use of the inbuilt Maths object.</p> <p>You can access the Maths object within the Script Editor by typing 'Maths' followed by a period. The Intelli-sense feature displays a list of the available mathematical functions provided by the Cephes Mathematical Library. For example:</p> <p>Session.Output "The square root of 9 is " & Maths.sqrt(9)</p> <p>Session.Output "2^10 = " & Maths.pow(2,10)</p> <p>The Maths object is available in the Business and Software Engineering, Systems Engineering and Ultimate editions of Enterprise Architect.</p>
Using COM / ActiveX Objects	<p>VBScript, JScript and Javascript can each create and work with ActiveX / COM objects. This can help you to work with external libraries, or to interact with other applications external to Enterprise Architect. For example, the Scripting.FileSystemObject Class can be used to read and write files on the local machine. The syntax for creating a new object varies slightly for each language, as illustrated by these examples:</p> <p>VBScript:</p> <p>set fsObject = CreateObject("Scripting.FileSystemObject")</p> <p>JScript:</p> <p>fsObject = new ActiveXObject("Scripting.FileSystemObject");</p> <p>Javascript:</p> <p>fsObject = new COMObject("Scripting.FileSystemObject");</p>

Using Javascript with out-of-process COM servers	<p>Users of Javascript in Enterprise Architect can access out-of-process COM servers. The application must be registered on the machine as providing local server support. The syntax for creating or obtaining a reference to an out-of-process server is:</p> <pre>var server = new COMObject(<i>progID</i>, true);</pre> <p>where <i>progID</i> is the registered program ID for the COM component ('Excel.Application', for example).</p>
System Script Library	<p>When Enterprise Architect is installed on your system, it includes a default script library that provides a number of helpful scripting functions, varying from simple string functions to functions for defining your own CSV or XMI import and export. To use the script library you must enable it in the 'MDG Technologies' dialog ('Configure > Technology > Manage').</p> <p>Scroll through the list of technologies, and select the 'Enabled' checkbox against 'EAScriptLib'.</p>

Notes

- The Script Editor is available in the Corporate, Business and Software Engineering, Systems Engineering and Ultimate editions
- Enterprise Architect scripting supports declaring variables to match the Enterprise Architect types; this enables the editor to present Intelli-sense, but is not necessary for executing the script

Session Object

The Session runtime object provides a common input/feedback mechanism across all script languages, giving access to objects of the types described in the System Type library. It is available through both the 'Scripts' tab and the script 'Console' tab to any script run within Enterprise Architect.

Properties


Properties	Detail
Attributes	<ul style="list-style-type: none"> • UserName - Returns the current windows username (read only) • Version - Returns the version of this object (read only)
Methods	<ul style="list-style-type: none"> • Input(string Prompt) - displays a dialog box prompting the user to input a value; returns the string value that was entered by the user • Output(string Output) - writes text to the current default output location; during: <ul style="list-style-type: none"> - Normal script execution, output is written to the 'Script' tab of the System Output window - Script Debugging, output is written to the Debug window - Use of the Script Console, output is written to the Console • Prompt(string Prompt, long PromptType) - displays a modal dialog containing the specified prompt text and button types; returns the 'PromptResult' value corresponding to the button that the user clicked
PromptType values	<ul style="list-style-type: none"> • promptOK = 1 • promptYESNO = 2 • promptYESNOCANCEL = 3 • promptOKCANCEL = 4
PromptResult values	<ul style="list-style-type: none"> • resultOK = 1 • resultCancel = 2 • resultYes = 3 • resultNo = 4
Session.Prompt Example	(VBScript) If (Session.Prompt("Continue?", promptYESNO) = resultYes) Then...

Script Debugging


Script debugging aids in the development and maintenance of model scripts, and monitoring their activity at the time of execution. While debugging a script, you can:

- Control execution flow using the 'Debug', 'Step Over', 'Step Into', 'Step Out' and 'Stop Script' buttons on the Script Editor toolbar
- Set Breakpoints, Recording Markers and Tracepoint Markers
- Use the Debug window to view output generated by the script
- Use the Locals window to inspect values of variables, including objects from the Automation Interface
- Use the Record & Analyze window to record a Sequence diagram of the script execution

Access

Ribbon	Code > Tools > Scripting > Scripts > right-click on [script name] > Debug Script
Other	Script Editor window toolbar : Click on the  toolbar icon

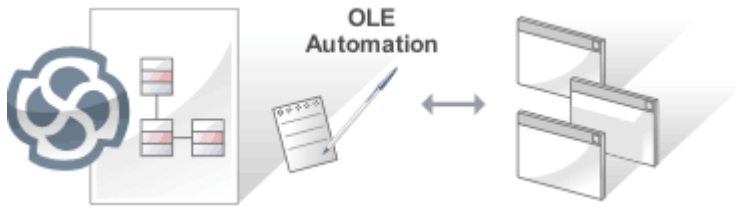
Begin debugging a model script

Step	Action
1	Open a model script in the Script Editor.
2	Set any Breakpoints on the appropriate line(s) of code.
3	Click on the  toolbar icon (Debug).

Notes

- Script debugging is supported for VBScript, JScript and Javascript
- VBScript and JScript require the Microsoft Process Debug Manager to be installed on the local machine; this is available through various Microsoft products including the free 'Microsoft Script Debugger'
- Breakpoints are not saved for scripts and will not persist when the script is next opened
- While debugging, script output is redirected to the Debug window

Enterprise Architect Object Model



The Enterprise Architect Object Model gives the scripter or programmer access to the underlying objects that you can use to query or manipulate the repository. The Object Model is accessible either from internal or external scripting environments or through Add-Ins. This is a powerful feature that ensures that a programmer is insulated from the underlying database where the repository is stored, protecting them from changes to the database structure or content. The objects are grouped into Packages and contain a useful, extensive and well documented set of properties and methods that are intuitive to use and allow access to elements, features, diagrams and project meta-data.

Automation provides a way for other applications to access the information in an Enterprise Architect model using Windows OLE Automation (ActiveX). Typically this involves scripting clients such as MS Word or Visual Basic, or using scripts created within Enterprise Architect using the Scripting window.

The Automation Interface provides a way of accessing the internals of Enterprise Architect models. Examples of things you can do using the Automation Interface include:

- Perform repetitive tasks, such as update the version number for all elements in a model
- Generate code from a StateMachine diagram
- Produce custom reports
- Perform ad hoc queries

Features

Feature	Description
Connecting to the Automation Interface	All development environments capable of generating ActiveX COM clients should be able to connect to the Enterprise Architect Automation Interface. This guide provides detailed instructions on connecting to the interface using Microsoft Visual Basic 6.0, Borland Delphi 7.0, Microsoft C# and Java. There are also more detailed steps on how to set-up Visual Basic; the principles are applicable to other languages.
Examples and Tips	Instruction on how to use the Automation Interface is provided by means of sample code. See pointers to the samples and other available resources. Also, consult the extensive Reference Section.
Calling Executables from Enterprise Architect	Enterprise Architect can be set up to call an external application. You can pass parameters on the current position selected in the Project Browser to the application being called. For instructions, go to the Call from Enterprise Architect topic. A more sophisticated method is to create Add-Ins, which are discussed in a separate section.

Using the Automation Interface

This section provides instructions on how to connect to and use the Automation Interface, including:

- Connecting to the interface
- Setting references in Visual Basic
- Examples and Tips

Connect to the Interface

All development environments capable of generating ActiveX Com clients can connect to the Enterprise Architect Automation Interface.

By way of example, these sections describe how to connect using several such tools. The procedure might vary slightly with different versions of these products.

Microsoft Visual Basic 6.0

Step	Action
1	Create a new project.
2	Select the 'Project References' menu option.
3	Select Enterprise Architect Object Model 2.0 from the list. If this does not appear, go to the command line and re-register Enterprise Architect using: EA.exe /unregister then EA.exe /register
4	See the general library documentation on the use of Classes. This example creates and opens a repository object: Public Sub ShowRepository() Dim MyRep As New EA.Repository MyRep.OpenFile "c:\eatest.eap" End Sub

Borland Delphi 7.0

Step	Action
1	Create a new project.
2	Select the 'Project Import Type Library' menu option.
3	Select Enterprise Architect Object Model 2.0 from the list. If this does not appear, go to the command line and re-register Enterprise Architect using: EA.exe /unregister then EA.exe /register
4	Click on the Create Unit button.

5	Include EA_TLB in Project1's Uses clause.
6	<p>See the general library documentation on the use of Classes. This example creates and opens a repository object:</p> <pre> procedure TForm1.Button1Click(Sender: TObject); var r: TRepository; b: boolean; begin r:= TRepository.Create(nil); b:= r.OpenFile('c:\eatest.eap'); end; </pre>

Microsoft C#

Step	Action
1	Select the 'Visual Studio Project Add Reference' menu option.
2	Click on the 'Browse' tab.
3	<p>Navigate to the folder in which you installed Enterprise Architect; usually:</p> <p>Program Files/Sparx Systems/EA</p> <p>Select</p> <p>Interop.EA.dll</p>
4	<p>See the general library documentation on the use of Classes. This example creates and opens a repository object:</p> <pre> private void button1_Click(object sender, System.EventArgs e) { EA.Repository r = new EA.Repository(); r.OpenFile("c:\\eatest.eap"); } </pre>

Java

Step	Action
1	<p>Copy the file:</p> <p>SSJavaCOM.dll</p> <p>from the Java API subdirectory of your installed directory, usually:</p> <p>Program Files/Sparx Systems/EA</p>

	into any location within the Windows PATH windows\system32 directory.
2	Copy the file eaapi.jar from the Java API subdirectory of your installed directory, usually: Program Files/Sparx Systems/EA to a location in the Java CLASSPATH or where the Java class loader can find it at run time.
3	All of the Classes described in the documentation are in the Package org.sparx. See the general library documentation for their use. This example creates and opens a repository object: <pre>public void OpenRepository() { org.sparx.Repository r = new org.sparx.Repository(); r.OpenFile("c:\\eatest.eap"); }</pre>

Set References In Visual Basic

It is possible to use the Enterprise Architect ActiveX interface with Visual Basic (VB). Use is ensured for Visual Basic version 6, but might vary slightly with versions other than version 6.

It is assumed that you have accessed VB through a Microsoft Application such as VB 6.0, MS Word or MS Access. If the code is not called from within Word, the Word VB reference must also be set.

On creating a new VB project, you set a reference to an Enterprise Architect Type Library and a Word Type Library.

Set References

Step	Action
1	Select the 'Tools References' menu option.
2	Select the 'Enterprise Architect Object Model 2.10' checkbox from the list.
3	Do the same for VB or VB Word: select the checkbox for the 'Microsoft Word 10.0 Object Library'.
4	Click on the OK button.

Notes

- If 'Enterprise Architect Object Model 2.10' does not appear in the list, go to the command line and manually re-enter Enterprise Architect using:
 - (To unregister Enterprise Architect) ea.exe /unregister
 - (To register Enterprise Architect) ea.exe /register
- Visual Basic 5/6 users should also note that the version number of the Enterprise Architect interface is stored in the VBP project file in a form similar to this:
 Reference=*\G{64FB2BF4-9EFA-11D2-8307-C45586000000}#2.2#0#..\..\..\Program Files\
 Sparx Systems\EA\EA.TLB#Enterprise Architect Object Model 2.02
 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
 Reference to objects in Enterprise Architect and Word should now be available in the Object Browser, which can be accessed from the main menu by pressing F2
 The drop-down list on the top-left of the window should now include Enterprise Architect and Word; if MS-Project is installed, also set this up

Examples and Tips

Points to consider

Subject	Points
Examples	<p>Instructions for using the interface are provided through sample code. There are several sets of examples:</p> <ul style="list-style-type: none"> • VB 6 and C# examples are available in the Code Samples folder under your Enterprise Architect installation (default: C:\Program Files\Sparx Systems\EA\Code Samples) • Enterprise Architect can be set up to call an external application • Several VB.NET code snippets are provided in the reference section • A comprehensive example of using Visual Basic to create MS Word documentation is available from the internet at www.sparxsystems.com/resources/developers/autint_vb.html • Additional samples are available from the Sparx Systems website; see the <i>Available Resources</i> topic
Tips and Tricks	<p>Also note these tips and tricks:</p> <ul style="list-style-type: none"> • An instance of the Enterprise Architect (EA.exe) process is executed when you initialize a new repository object - this process must remain running in order to perform automation tasks; if the main window is visible, you can safely minimize it, but it must remain running • The Enterprise Architect ActiveX Interface is a functional interface rather than a data interface; when you load data through the interface there is a noticeable delay as Enterprise Architect user interface elements (such as Windows and menus) are loaded and the specified database connection is established • Collections use a zero-based index; for example, Repository.Models(0) represents the first model in the repository • During the development of your client software your program might terminate unexpectedly and leave EA.exe running in such a state that it is unable to support further interface calls; if your program terminates abnormally, ensure that Enterprise Architect is not left running in the background (see the Windows 'Task Manager / Process' tab) • A handle to a currently running instance of Enterprise Architect can be obtained through the use of a GetObject() call (see the reference page for the App object); accessing your Enterprise Architect model via the App object enables querying the current User Interface status, such as using GetContextItem() on the Repository object to detect the current selection by the user, allowing for rapid prototyping and testing
Enterprise Architect Not Closing	<p>After all processing by an automation controller is complete, it is recommended to call CloseFile() and Exit() on the Repository object, then set all references to the repository object to null.</p> <pre>repository.CloseFile(); repository.Exit(); repository = null;</pre> <p>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</p>

	<p>release of the COM pointers, call the memory management functions:</p> <p>GC.Collect();</p> <p>GC.WaitForPendingFinalizers();</p> <p>There are additional concerns when controlling a running instance of Enterprise Architect that loads Add-Ins - see the <i>Tricks and Traps</i> topic for details.</p>
--	---

Call from Enterprise Architect

Enterprise Architect can be set up to call an external application. You can pass parameters on the current position selected in the Project Browser to the application being called. This helps you to:

- Add a command line for an application
- Define parameters to pass to this application

The parameters required for running the AutInt executable are:

- The Enterprise Architect file parameter \$f and
- The current PackageID \$p

Hence the arguments should simply contain: \$f,\$p.

Once this has been set up, the application can be called from the 'Extend' ribbon in Enterprise Architect using the 'Extend > <YourApplication>' option.

Access

Ribbon	Start > Workspace > Preferences > Customize > Tools
--------	---

Parameters to pass information to external applications

Parameter	Description
\$d	Diagram ID Notes: ID for accessing associated diagram.
\$D	Diagram GUID Notes: GUID for accessing the associated diagram.
\$e	Comma separated list of element IDs Notes: All elements selected in the current diagram.
\$E	Comma separated list of element GUIDs Notes: All elements selected in the current diagram.
\$f	Project Name Notes: For example: C:\projects\EAexample.eap.
\$F	Calling Application (Enterprise Architect) Notes: 'Enterprise Architect'.
\$p	Current Package ID Notes: For example: 144.
\$P	Package GUID

	Notes: GUID for accessing this Package.
--	---

Available Resources

Resources

Available resources include:

Resource	Download Link
VB 6 Add-In for generating MS Word documentation.	www.sparxsystems.com/resources/developers/autint_vb.html
VB 6 Add-In to display a custom ActiveX graph control within the Enterprise Architect window as a new view.	www.sparxsystems.com/resources/developers/autint_vb_custom_view.html
A basic Add-In framework written in C#. Useful as a starting point for authoring your own custom Enterprise Architect Add-In.	www.sparxsystems.com/bin/CS_AddinFramework.zip
An extension on the CS_AddinFramework example showing how to export Tagged Values to a .csv file.	www.sparxsystems.com/bin/CS_AddinTaggedCSV.zip
A basic Add-In skeleton written in Delphi.	www.sparxsystems.com/bin/DelphiDemo.zip
A simple example Add-In written in C#.	www.sparxsystems.com/bin/CS_Sample.zip

Reference

This section provides detailed information on all the objects available in the object model provided by the Automation Interface, including:

Object Groups

Group
App Object
Enumerations
Repository Package
Element Package
Element Features Package
Connector Package
Diagram Package
Project Interface Package
Document Generator Interface Package
Mail Interface Package
Code Samples

Interface Overview

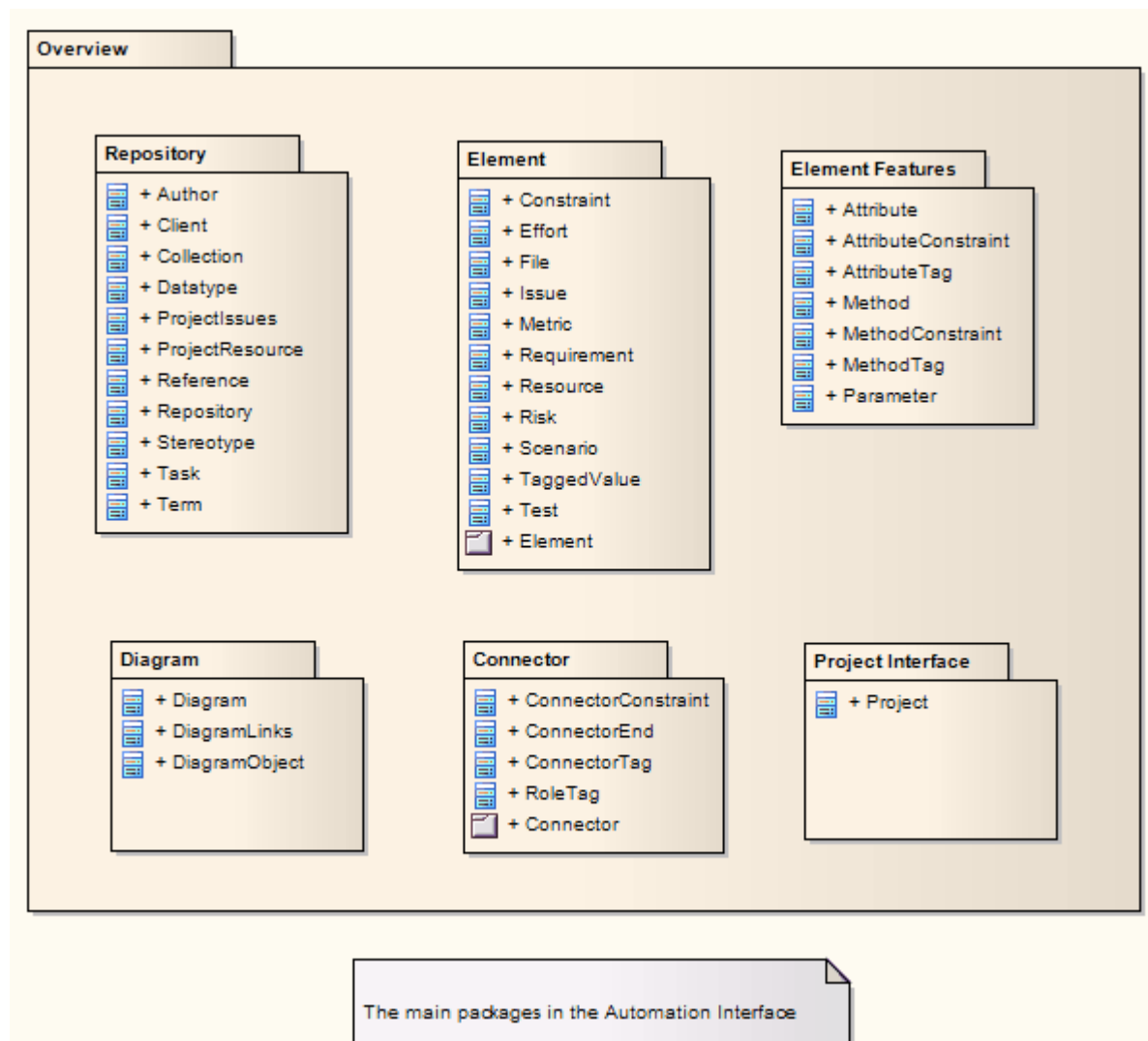
This section provides an overview of the main components of the Automation Interface.

Main Packages of Automation Interface

Package	Detail
Repository Package	Represents the model as a whole and provides entry to model Packages and collections.
Element Package	Identifies the basic structural units (such as Class, Use Case and Object).
Element Features Package	Identifies the attributes and operations defined on an element.
Diagram Package	Describes the visible drawings contained in the model.
Connector Package	Defines the relationships between elements.

Packages and Contents

This diagram illustrates the main interface Packages and their associated contents. Each UML element in this User Guide can be created by Automation and can be accessed either through the various collections that exist or, in some cases, directly.



The Repository Class is the starting point for all use of the Automation Interface. It contains the high level system objects and entry point into the model itself using the Models collection and the other system-level collections.

App Object

The App object represents a running instance of Enterprise Architect. Its object provides access to the Automation Interface.

Attributes

Attribute	Type
Project	Project Notes: Read only Provides a handle to the Project Interface.
Repository	Repository Notes: Read only Provides a handle to the Repository object.
Visible	Boolean Notes: Read/Write Whether or not the application is visible.

GetObject() Support

The App object is creatable and a handle can be obtained by creating one. In addition, clients can use the equivalent of Visual Basic's GetObject() to obtain a reference to a currently running instance of Enterprise Architect.

Use this method to more quickly test changes to Add-Ins and external clients, as the Enterprise Architect application and data files do not have to be constantly re-loaded.

For example:

```
Dim App as EA.App
Set App = GetObject("EA.App")
MsgBox App.Repository.Models.Count
```

Another example, which uses the App object without saving it to a variable:

```
Dim Rep as EA.Repository
Set Rep = GetObject( "EA.App").Repository
MsgBox Rep.ConnectionString
```

Enumerations

These enumerations are defined by the Automation Interface:

Automation Interface Enumerations

Enumeration	Link
Constant Layout Styles	Constant Layout Styles
Create Baseline Flag	Create Baseline Flag
Create Model Type	Create Model Type
Document Break	Document Break
Document Page Orientation	Document Page Orientation
Document Type	Document Type
Enterprise Architect Edition Types	Enterprise Architect Edition Types
Enumeration Relation Set Type	Enumeration Relation Set Type
Export Package XMI Flag	Export Package XMI Flag
Mail Interface Message Flag	Mail Interface Message Flag
MDG Menus	MDG Menus
Object Type	Object Type
PropType	PropType
Reload Type	Reload Type
Scenario Diagram Type	Scenario Diagram Type
Scenario Step Type	Scenario Step Type
Scenario Test Type	Scenario Test Type
XMI Type	XMI Type

ConstLayoutStyles

The enum values defined here are used exclusively for the 'Lay Out a Diagram' method. You use these values to define the layout options as provided by the 'Layout > Tools > Layout Diagram' ribbon option.

Enum Values

Value	Meaning
IsCrossReduceAggressive	Perform aggressive Cross-reduction in the layout process (time consuming).
IsCycleRemoveDFS	Use the Depth First Cycle Removal algorithm.
IsCycleRemoveGreedy	Use the Greedy Cycle Removal algorithm.
IsDiagramDefault	Use existing layout options specified for this diagram.
IsInitializeDFSIn	Initialize the layout using the Depth First Search Inward algorithm.
IsInitializeNaive	Initialize the layout using the Naïve Initialize Indices algorithm.
IsInitializeDFSOut	Initialize the layout using the Depth First Search Outward algorithm.
IsLayeringLongestPathSink	Layer the diagram using the Longest Path Sink algorithm.
IsLayeringLongestPathSource	Layer the diagram using the Longest Path Source algorithm.
IsLayeringOptimalLinkLength	Layer the diagram using the Optimal Link Length algorithm.
IsLayoutDirectionDown	Direct connectors to point down.
IsLayoutDirectionLeft	Direct connectors to point left.
IsLayoutDirectionRight	Direct connectors to point right.
IsLayoutDirectionUp	Direct connectors to point up.
IsProgramDefault	Use factory default layout options as specified by Enterprise Architect.

CreateBaselineFlag

The CreateBaselineFlag enumeration is used in Baseline Management, when creating a Baseline.

Enum Values

Value	Meaning
cbSaveToStub	Baseline this Package with only immediate children (child Packages are included as stubs only).

CreateModelType

The CreateModelType enumeration is used in the CreateModel method on the Repository Class.

Enum Values

Value	Meaning
cmEAPFromBase	Create a copy of the EABase model file to the specified file path.
cmEAPFromSQLRepository	Create a .eap file shortcut to an SQL-based repository; requires user interaction to provide SQL connection details.

DocumentBreak

The DocumentBreak enumeration is used in the InsertBreak method on the DocumentGenerator Class.

Enum Values

Value	Meaning
breakPage	Insert a page break in the document.
breakSection	Insert a section break in the document.

DocumentPageOrientation

The DocumentPageOrientation enumeration is used in the SetPageOrientation method on the DocumentGenerator Class.

Enum Values

Value	Meaning
pagePortrait	Sets the current page orientation to Portrait.
pageLandscpae	Sets the current page orientation to Landscape.

DocumentType

The DocumentType enumeration is used in the SaveDocument method on the DocumentGenerator Class.

Enum Values

Value	Meaning
dtRTF	Save the document file to disk as an RTF document.
dtHTML	Save the document file to disk as a HTML document.
dtPDF	Save the document file to disk as a PDF document.
dtDOCX	Save the document file to disk as a DOCX document.

EAEditionTypes

The EAEditionTypes enumeration identifies the current level of licensed functionality available.

```
EAEditionTypes theEdition = theRepository.GetEAEdition();  
if (theEdition == EAEditionTypes.piDesktop)  
...  
else if (theEdition == EAEditionTypes.piProfessional)  
...  
...
```

The enumeration defines these formal values:

- piLite
- piDesktop
- piProfessional
- piCorporate
- piBusiness
- piSystemEng
- piUltimate

There is no separate value for the trial edition; the Repository.GetEAEdition() function returns the appropriate EAEditionTypes value for whichever edition the user has selected to trial.

EnumRelationSetType

This enumeration represents values returned from the GetRelationSet method of the Element object.

Enum Values

Value	Meaning
rsDependEnd	List of elements that depend on the current element.
rsDependStart	List of elements that the current element depends on.
rsGeneralizeEnd	List of elements that are generalized by the current element.
rsGeneralizeStart	List of elements that the current element generalizes.
rsParents	List of all parent elements of the current element.
rsRealizeEnd	List of elements that are realized by the current element.
rsRealizeStart	List of elements that the current element realizes.

ExportPackageXMIFlag

The ExportPackageXMIFlag enumeration is used in Package control, when exporting to XMI.

Enum Values

Value	Meaning
epExcludeEAExtensions	Export this Package without any tool specific information.
epSaveToStub	Export this Package with only immediate children (child Packages are included as stubs only).

MDGMenus

Use this enumeration when providing the 'HiddenMenus' property to MDG_GetProperty.

These options are exclusive of one another and can be read or added to hide more than one menu.

Enum Values

Value	Meaning
mgBuildProject	'Hide Build Project' menu option.
mgMerge	'Hide Merge' menu option.
mgRun	'Hide Run' menu option.

MessageFlag

The MessageFlag enumeration is used in both the SendMailMessage and ComposeMailMessage methods of the MailInterface, to specify a flag to attach to the message.

Enum Values

Value	Meaning
mfNone	Do not flag the message.
mfComplete	Flag the message as 'Complete'.
mfPurple	Flag the message with a 'Purple' flag.
mfOrange	Flag the message with an 'Orange' flag.
mfGreen	Flag the message with a 'Green' flag.
mfYellow	Flag the message with a 'Yellow' flag.
mfBlue	Flag the message with a 'Blue' flag.
mfRed	Flag the message with a 'Red' flag.

ObjectType

The ObjectType enumeration identifies Enterprise Architect object types even when referenced through a Dispatch interface. For example:

```
var treeSelectedType = Repository.GetTreeSelectedItemType();
switch (treeSelectedType)
{
    case otElement :
    {
        // Code for when an element is selected
        var theElement as EA.Element;
        theElement = Repository.GetTreeSelectedObject();
        break;
    }
    case otPackage :
    {
        // Code for when a Package is selected
        var thePackage as EA.Package;
        thePackage = Repository.GetTreeSelectedObject();
        break;
    }
}
```

Valid Enumeration Values

otAttribute
otAttributeConstraint
otAttributeTag
otAuthor
otClient
otCollection
otConnector
otConnectorConstraint
otConnectorEnd
otConnectorTag
otConstraint
otCustomProperty
otDatatype
otDiagram
otDiagramLink
otDiagramObject
otEffort

otElement
otEventProperties
otEventProperty
otFile
otIssue
otMailInterface
otMethod
otMethodConstraint
otMethodTag
otMetric
otModel
otNone
otPackage
otParameter
otParamTag
otPartition
otProject
otProjectIssues
otProjectResource
otProperties
otProperty
otPropertyType
otReference
otRepository
otRequirement
otResource
otRisk
otRoleTag
otScenario
otScenarioExtension
otScenarioStep
otStereotype
otSwimlane
otSwimlaneDef
otSwimlanes
otTaggedValue
otTask
otTerm
otTest
otTransition

PropType

The PropType enumeration gives the automation programmer an indication of what sort of data is going to be stored by this property.

Enum Values

Value	Meaning
ptArray	An array containing values of any type.
ptBoolean	True or False.
ptEnum	A string being an entry in the semi-colon separated list specified in the validation field of the Property.
ptFloatingPoint	4 or 8 byte floating point value.
ptInteger	16-bit or 32-bit signed integer.
ptString	Unicode string.

ReloadType

The ReloadType enumeration represents values returned from the GetReloadItem and PeekReloadItem methods of the ModelWatcher Class. It has four possible values, which define the type of change that was made to a model.

Enum Values

Value	Meaning
rtElement	The Item parameter represents a particular element that must be reloaded.
rtEntireModel	Entire model must be reloaded to ensure that all changes are reloaded.
rtNone	No change in the model.
rtPackage	The Item parameter represents a particular Package that must be reloaded.

ScenarioDiagramType

The ScenarioDiagramType enumeration provides these enumeration values to the Project.GenerateDiagramFromScenario() method. They specify the type of diagram to generate.

Enum Values

Value	Meaning
sdActivity	Generate an Activity diagram.
sdActivityWithAction	Generate an Activity diagram with an Action.
sdActivityWithActionPin	Generate an Activity diagram with an ActionPin.
sdActivityWithActivityParameter	Generate an Activity diagram with an ActivityParameter.
sdRobustness	Generate a Robustness diagram.
sdRuleFlow	Generate a RuleFlow diagram.
sdSequence	Generate a Sequence diagram.
sdState	Generate a StateMachine diagram.

ScenarioStepType

The ScenarioStepType enumeration is used to identify the steps of a scenario, and the entity performing the step.

Enum Values

Value	Meaning
stActor	Identify that the step is an action performed by an actor.
stSystem	Identify that the step is an action performed by the system.

ScenarioTestType

The ScenarioTestType enumeration provides these enumeration values to the Project.GenerateTestFromScenario() method, to specify the type of test to generate.

Enum Values

Value	Meaning
stHorizontalTestSuite	Generate a horizontal Test Suite diagram.
stVerticalTestSuite	Generate a vertical Test Suite diagram.
stExternal	Generate an external Test Case element.
stInternal	Generate an internal test.

XMIType

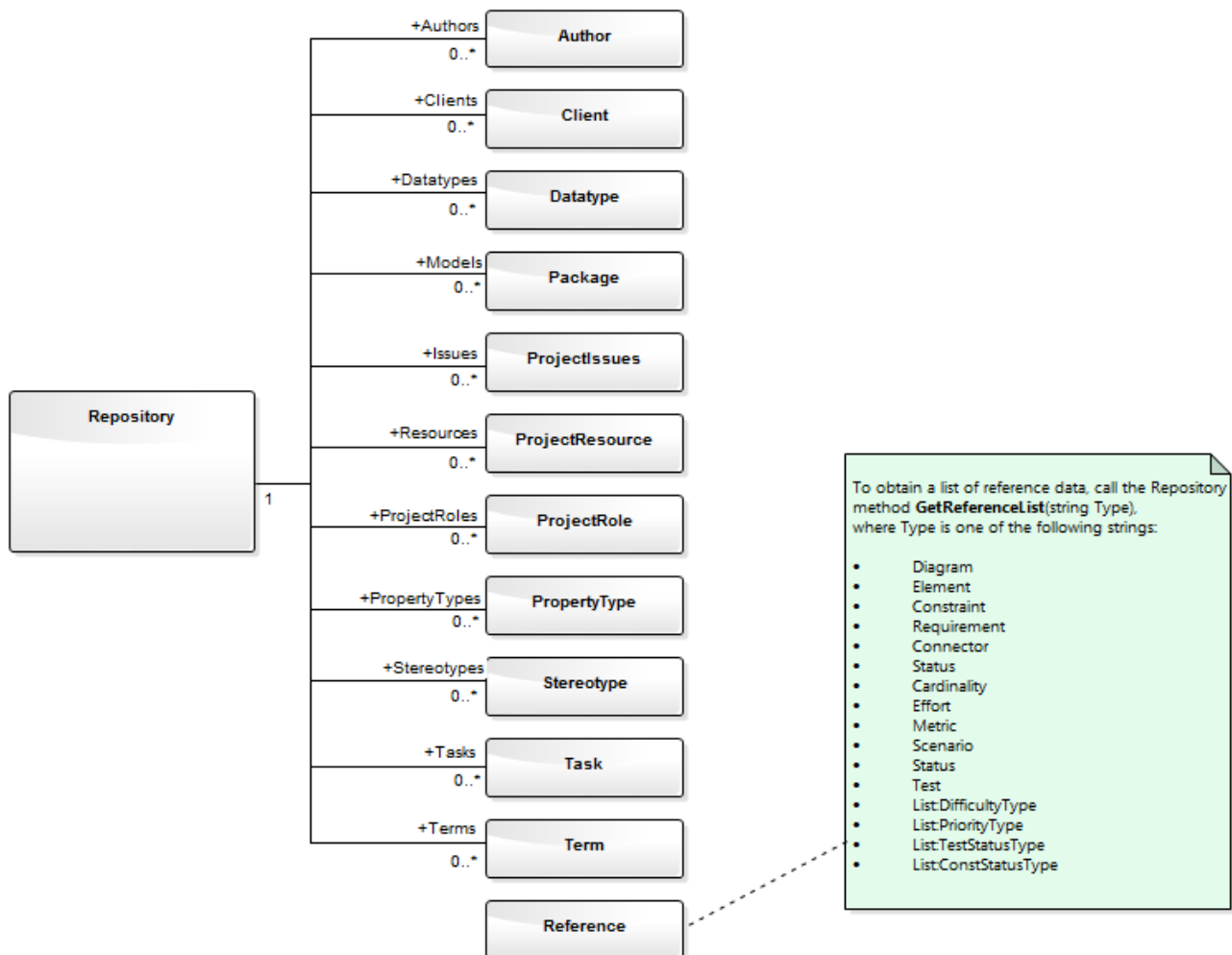
These enumeration values are used in the `Project.ExportPackageXMI()` and `Project.ExportPackageXMIEx()` methods, to specify the XMI export type.

- `xmiEADefault = 0`
- `xmiRoseDefault = 1`
- `xmiEA10 = 2`
- `xmiEA11 = 3`
- `xmiEA12 = 4`
- `xmiRose10 = 5`
- `xmiRose11 = 6`
- `xmiRose12 = 7`
- `xmiMOF13 = 8`
- `xmiMOF14 = 9`
- `xmiEA20 = 10`
- `xmiEA21 = 11`
- `xmiEA211 = 12`
- `xmiEA212 = 13`
- `xmiEA22 = 14`
- `xmiEA23 = 15`
- `xmiEA24 = 16`
- `xmiEA241 = 17`
- `xmiEA242 = 18`
- `xmiEA251 = 19`
- `xmiEcore = 20`
- `xmiBPMN20 = 21`
- `xmiXPDL22 = 22`

Repository Package

The Repository Package contains the high level system objects and the entry point into the model itself, using the Models collection and the other system level collections.

This diagram shows the collections of the Repository interface. Association Target roles correspond to member variable names in the Repository interface. The associated Classes represent the object type used in each collection.



Author Class

An Author object represents a named model author. Authors can be accessed using the Repository Authors collection.

Associated table in .EAP file

t_authors

Author Attributes

Attribute	Remarks
Name	String Notes: Read/Write The Author name.
Notes	String Notes: Read/Write Notes about the author.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Roles	String Notes: Read/Write Roles the author might play in this project.

Author Methods

Method	Remarks
GetLastError ()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update ()	Boolean Notes: Updates the current Author object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Client Class

A Client represents one or more people or organizations related to the project. Clients can be accessed using the Repository Clients collection.

Associated table in .EAP file

t_clients

Client Attributes

Attribute	Remarks
EMail	String Notes: Read/Write The client's email address.
Fax	String Notes: Read/Write The client's fax number.
Mobile	String Notes: Read/Write The client's mobile phone number, if available.
Name	String Notes: Read/Write The client's name.
Notes	String Notes: Read/Write Notes about the client.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through the Dispatch interface.
Organization	String Notes: Read/Write The client's associated organization.
Phone1	String Notes: Read/Write The client's main phone number.

Phone2	String Notes: Read/Write The client's second phone number.
Roles	String Notes: Read/Write Roles this client might play in the project.

Client Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Updates the current Client object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Collection Class

Collection is the main collection Class used by all elements within the Automation Interface. It contains methods to iterate through the collection, refresh the collection and delete an item from the collection.

It is important to realize that when AddNew is called, the item is not automatically added to the current collection. The typical steps are:

- Call AddNew to add a new item
- Modify the item as required
- Call Update on the item to save it to the database
- Call Refresh on the collection to include it in the current set

Delete is the same; until Refresh is called, the collection still contains a reference to the deleted item, which should not be called.

Each method can be used to iterate through the collection for languages that support this type of construct.

Collection Attributes

Attribute	Remarks
Count	Short Notes: Read only The number of objects referenced by this list.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

Collection Methods

Method	Remarks
AddNew(string Name, string Type)	Object Notes: Adds a new item to the current collection. The interface is the same for all collections; you must provide a Name and Type argument. What these arguments are used for depends on the actual collection being accessed. For example, when adding a new element to the Elements collection, the Type string can be either a basic UML element type or a fully qualified element type (stereotype) defined by a profile, such as SysML::Requirement, differentiating it from a standard requirement. Also note that you must call Update() on the returned object to complete the AddNew. If Update() is not called the object is left in an indeterminate state. Parameters: <ul style="list-style-type: none">• Name: String• Type: String (up to 30 characters long)

Delete(short index)	<p>Void</p> <p>Notes: Deletes the item at the selected reference.</p> <p>Parameters:</p> <ul style="list-style-type: none"> index: Short
DeleteAt(short index, boolean Refresh)	<p>Void</p> <p>Notes: Deletes the item at the selected index. The second parameter is currently unused.</p> <p>Parameters:</p> <ul style="list-style-type: none"> index: Short Refresh: Boolean
GetAt(short index)	<p>Object</p> <p>Notes: Retrieves the array object using a numerical index. If the index is out of bounds, an error occurs.</p> <p>Parameters:</p> <ul style="list-style-type: none"> index: Short
GetByName(string Name)	<p>Object</p> <p>Notes: Gets an item in the current collection by name. Supported for Model, Package, Element, Diagram and element TaggedValue collections.</p> <p>If the collection does not contain any items (or, for the Tagged Value collection, if the collection contains items but the method cannot locate an object with the specified name) the method returns a null value. For other collections, if the method is unable to find an object with the specified name, it raises an exception.</p> <p>Parameters:</p> <ul style="list-style-type: none"> Name: String
GetLastError()	<p>String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p>
Refresh()	<p>Void</p> <p>Notes: Refreshes the collection by re-querying the model and reloading the collection. Should be called after adding a new item or after deleting an item.</p>
Update()	<p>Boolean</p> <p>Notes: Updates the current Collection object after modification or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>

The AddNew Function

The AddNew() function is used widely across the API to add new objects to a Collection. In all cases you must provide a Name and Type argument, but what these arguments are used for depends on the actual collection being accessed. For example, when adding a new element to the Elements collection, the 'Type' string can be either a basic UML element type or a fully qualified element type (stereotype) defined by a profile, such as SysML::Requirement differentiated from a standard requirement.

AddNew Attribute Arguments

This table provides guidance in specifying the AddNew arguments for each of the object attributes.

Attribute	Arguments
AttributeConstraints	Name - The name of the constraint. Type - The constraint type
Attributes	Name - The name of the attribute. Type - The attribute type.
AttributesEx	Name - The name of the attribute. Type - The attribute type.
AttributeTags	Name - The fully-qualified name, or plain text. Type - The value of the Tagged Value.
Authors	Name - The author name. Type - The author role.
Clients	Name - The client name. Type - The client role.
ConnectorConstraints	Name - The name of the constraint. Type - The constraint type.
ConnectorConveyedItems	Name - The GUID of an element. Type - <i>Not used</i> . Note: This does not return an object.
Connectors	Name - The name of the connector. Type - The connector type (for example 'Realization').
ConnectorTags	Name - The fully-qualified name, or plain text. Type - The value of the Tagged Value.
Constraints	Name - The name of the constraint. Type - The constraint type.
ConstraintsEx	Name - The name of the constraint.

	Type - The constraint type.
CustomProperties	You cannot create these.
DataTypes	Name - The datatype name. Type - The datatype type.
DiagramLinks	Name - <i>Not used</i> . Type - The style string (such as 'l=200;r=400;t=200;b=600;') (You might prefer to leave the Type empty and use the Functions on this interface for size, colors and so on).
DiagramObjects	Name - This can either be an empty string, or it can specify the initial Left, Right, Top and Bottom values for the new DiagramObject. For example: <code>diagram.DiagramObjects.AddNew("l=200;r=400;t=200;b=600;", "")</code> Note: Top and Bottom values should be specified here as positive numbers, but will be set in the repository as negative values. Type - Unused.
Diagrams	Name - The name of the diagram. Type - This can be either a standard UML metaclass type (such as 'Class' or 'UseCase') or a fully-qualified metatype defined by an MDG Technology (such as 'BPMN2.0::BusinessProcess' or 'SysML1.4::Block').
Efforts	Name - The name of the effort. Type - The effort type.
Elements	Name - The name of the new element. If the repository has an auto-name counter defined for the element type being created, pass an empty string to use the auto-name counter instead. Type - Can be either a standard UML metaclass type (such as 'Class' or 'UseCase') or a fully-qualified metatype defined by an MDG Technology (such as 'BPMN2.0::BusinessProcess' or 'SysML1.4::Block').
Files	Name - The full pathname of the file. Type - The file type (such as 'Local File' or 'Web Address').
Issues	Name - The name of the issue. Type - The problem type, (such as 'Issue' or 'Defect')
MethodPostConditions	Name - The name of the constraint. Type - The constraint type
MethodPreconditions	Name - The name of the constraint. Type - The constraint type.
Methods	Name - The name of the method. Type - The return value of the method.
MethodsEx	Name - The name of the method.

	Type - The return value of the method.
MethodTags	Name - The fully-qualified name, or plain text. Type - The value of the Tagged Value.
Metrics	Name - The name of the metric. Type - The metric type.
Models	Name - The name of the model. Type - Unused.
Packages	Name - The name of the Package. Type - Unused.
Parameters	Name - The parameter name. Type - The parameter type.
ParamTags	Name - The fully-qualified name or plain text. Type - The value of the Tagged Value.
Partitions	Name - The partition name. Type - The partition note.
ProjectIssues	Name - The name of the issue. Type - The issue type (such as 'Request', 'Defect', or 'Release')
ProjectResources	Name - The resource name. Type - The resource role.
ProjectRole	Name - The role name. Type - <i>Not used</i> .
PropertyTypes	Name - The tag name. Type - The description (limited to 50 characters).
Requirements	Name - The name of the requirement. Type - The requirement type.
RequirementsEx	Name - The name of the requirement. Type - The requirement type.
Resources	Name - The resource name. Type - The resource role.
Risks	Name - The name of the risk. Type - The risk type.
ScenarioExtension	Name - The extension name. Type - The scenario type

ScenarioStep	Name - The step name. Type - The ScenarioStep type value.
Scenarios	Name - The name of the scenario. Type - The scenario type.
Stereotypes	Name - The stereotype name. Type - The element this applies to. Note: You can only support multiple elements from within a Profile.
Tasks	Name - The task name. Type - The task type.
TemplateBindings	Name - The formal name of the binding. Type - The actual name of the binding or element GUID.
TemplateParameters	Name - The parameter name. Type - The parameter type
Terms	Name - The term name. Type - The term type.
Tests	Name - The name of the test. Type - The test type.
Transitions	Name - The transition name. Type - The transition value.

Datatype Class

A Datatype is a named type that can be associated with attribute or method types. It typically is related to either code engineering or database modeling. Datatypes also indicate which language or database system they relate to. Datatypes can be accessed using the Repository Datatypes collection.

Associated table in .EAP file

t_datatypes

Datatype Attributes

Attribute	Remarks
DatatypeID	Long Notes: Read/Write The instance ID for this datatype within the current model; this is system maintained.
DefaultLen	Long Notes: Read/Write The default length (DDL only).
DefaultPrec	Long Notes: Read/Write The default precision (DDL only).
DefaultScale	Long Notes: Read/Write The default scale (DDL only).
GenericType	String Notes: Read/Write The associated generic type for this data type.
HasLength	String Notes: Read/Write Indicates whether the datatype has a length component.
MaxLen	Long Notes: Read/Write The maximum length (DDL only).
MaxPrec	Long Notes: Read/Write

	The maximum precision (DDL only).
MaxScale	Long Notes: Read/Write The maximum scale (DDL only).
Name	String Notes: Read/Write The datatype name (such as integer). This appears in the related drop-down datatype lists where appropriate.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Product	String Notes: Read/Write The datatype product, such as Java, C++ or Oracle.
Size	Long Notes: Read/Write The datatype size.
Type	String Notes: Read/Write The type can be DDL for database datatypes or Code for language datatypes.
UserDefined	Long Notes: Read/Write Indicates if the datatype is a user defined type or system generated. Datatypes distributed with Enterprise Architect are all system generated. Datatypes created in the 'Datatype' dialog are marked 1 (True).

Datatype Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Updates the current Datatype object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

EventProperties Class

An EventProperties object is passed to BroadcastFunctions to facilitate parameter passing.

EventProperties Attributes

Attribute	Remarks
Count	Long Notes: Read only The number of parameters being passed to this broadcast event.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

EventProperties Methods

Method	Remarks
Get(object Index)	EventProperty Class Notes: Read only Returns an EventProperty in the list, raising an error if Index is out of range. Parameters: <ul style="list-style-type: none">• Index: Variant - can either be a number representing a zero-based index into the array, or a string representing the name of the EventProperty: for example, Props.Get(3) or Props.Get("ObjectID")

EventProperty Class

EventProperty objects are always part of an EventProperties collection, and are passed to Add-In methods responding to broadcast events.

EventProperty Attributes

Attribute	Remarks
Description	String Notes: An explanation of what this property represents.
Name	String Notes: A string distinguishing this property from others in the list.
ObjectType	ObjectType Notes: Distinguishes objects referenced through a Dispatch interface.
Value	Variant Notes: A string, number or object reference representing the property value.

ModelWatcher Class

The ModelWatcher object enables an automation client to track changes in a particular model.

ModelWatcher Attributes

Attribute	Remarks
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

ModelWatcher Methods

Methods	Remarks
GetReloadItem (object Item)	ReloadType Notes: The object that must be reloaded in order to see all changes is returned through the Item parameter. If there are no changes or the entire model must be reloaded, this value is returned as null (C#) or Nothing (VB). Calling this method clears the records so that the next time it is called the return values refer only to new changes. Returns a value from the ReloadType enumeration that specifies which type of change, if any, has occurred. Parameters: <ul style="list-style-type: none">Item: Object
PeekReloadItem	ReloadType Notes: This method behaves identically to 'GetReloadItem()' but does not clear the change record.

Notes

- After your model has been loaded, you only create the ModelWatcher once; if you reload the model, or load another model, the created ModelWatcher is still valid

Package Class

A Package object corresponds to a Package element in the Enterprise Architect Project Browser. Packages can be accessed either through the Repository Models collection (a Model is a special form of Package) or through the Package Packages collection.

Note that a Package has an Element object as an attribute; this corresponds to an Enterprise Architect Package element in the t_object table and is used to associate additional information (such as scenarios and constraints) with the logical Package.

To set additional information for a Package, reference the Element object directly. Also note that if you add a Package to a diagram, you should add an instance of the element (not the Package itself) to the DiagramObject Class for a diagram.

Associated table in .EAP file

t_package

Package Attributes

Attribute	Remarks
Alias	String Notes: Read only Alias
BatchLoad	Long Notes: Read/Write Flag to indicate that the Package is batch loaded during batch import from controlled Packages. Not currently used.
BatchSave	Long Notes: Read/Write Boolean value to indicate whether the Package is included in the batch XMI export list or not.
CodePath	String Notes: Read/Write The path where associated source code is found. Not currently used.
Connectors	Collection Notes: Read only The collection of connectors.
Created	Date Notes: Read/Write Date the Package was created.

Diagrams	<p>Collection</p> <p>Notes: Read only</p> <p>A collection of diagrams contained in this Package.</p>
Element	<p>Element</p> <p>Notes: Read only</p> <p>The associated element object; use to get/set common information such as Stereotype, Complexity, Alias, Author, Constraints, Tagged Values and Scenarios.</p>
Elements	<p>Collection</p> <p>Notes: Read only</p> <p>A collection of elements that belong to this Package.</p>
Flags	<p>String</p> <p>Notes: Read/Write</p> <p>Extended information about the Package.</p>
IsControlled	<p>Boolean</p> <p>Notes: Read/Write</p> <p>Indicates if the Package has been marked as Controlled.</p>
IsModel	<p>Boolean</p> <p>Notes: Read only</p> <p>Indicates if the Package is a model or a Package.</p>
IsNamespace	<p>Boolean</p> <p>Notes: Read/Write</p> <p>True indicates that 'Package is a Namespace root'.</p> <p>Use 0 and 1 to set False and True.</p>
IsProtected	<p>Boolean</p> <p>Notes: Read/Write</p> <p>Indicates if the Package has been marked as 'Protected'.</p>
IsVersionControlled	<p>Boolean</p> <p>Notes: Read only</p> <p>Indicates whether or not this Package is under version control.</p>
LastLoadDate	<p>Date</p> <p>Notes: Read/Write</p> <p>The date XML was last loaded for the Package.</p>
LastSaveDate	<p>Date</p> <p>Notes: Read/Write</p> <p>The date XML was last saved from the Package.</p>
LogXML	<p>Boolean</p>

	Notes: Read/Write Indicates if XML export information is to be logged.
Modified	Date Notes: Read/Write Date the Package was last modified.
Name	String Notes: Read/Write The name of the Package.
Notes	String Notes: Read/Write Notes about this Package.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Owner	String Notes: Read/Write. The Package owner when using controlled Packages.
PackageGUID	Variant Notes: Read only The global Package ID; valid across models.
PackageID	Long Notes: Read only The local Package ID number. Valid only in this model file.
Packages	Collection Notes: Read only A collection of contained Packages that can be walked through.
ParentID	Long Notes: Read/Write The ID of the Package that is the parent of this one. 0 indicates that this Package is a model (that is, it has no parent).
TreePos	Long Notes: Read/Write The relative position in the tree compared to other Packages (use to sort Packages).
UMLVersion	String Notes: Read/Write

	The UML version for XMI export purposes.
UseDTD	Boolean Notes: Read/Write Indicates if a DTD is to be used when exporting XML.
Version	String Notes: Read/Write The version of the Package.
XMLPath	String Notes: Read/Write The path to which the XML is saved when using controlled Packages.

Package Methods

Method	Remarks
ApplyGroupLock (string aGroupName)	Boolean Notes: Applies a group lock to the Package object, for the specified group, on behalf of the current user. Returns True if the operation is successful; returns False if the operation is unsuccessful. Use GetLastError() to retrieve error information. Parameters: <ul style="list-style-type: none"> aGroupName: String - The name of the security group for which to apply the lock
ApplyGroupLockRecursive (string aGroupName, boolean IncludeElements, boolean IncludeDiagrams, boolean IncludeSubPackages)	Boolean Notes: Applies a group lock to the Package object, object, and all of the Package, diagrams and elements contained within that Package, for the specified group, on behalf of the current user. Returns True if the operation is successful; returns False if the operation is unsuccessful. Use 'GetLastError()' to retrieve error information. Parameters <ul style="list-style-type: none"> aGroupName: String - The name of the security group for which to apply the lock IncludeElements: Boolean - Recursively apply group lock to child elements IncludeDiagrams: Boolean - Recursively apply group lock to child diagrams IncludeSubPackages: Boolean - Recursively apply group lock to child Packages
ApplyUserLock ()	Boolean Notes: Applies a user lock to the Package object for the current user. Returns True if the operation is successful; returns False if the operation is unsuccessful. Use 'GetLastError()' to retrieve error information.
ApplyUserLockRecursive	

(boolean IncludeElements, boolean IncludeDiagrams, boolean IncludeSubPackages)	<p>Boolean</p> <p>Notes: Applies user locks to the Package object, and all of the Packages, diagrams and elements contained within that Package.</p> <p>Returns True if the operation is successful; returns False if the operation is unsuccessful. Use GetLastError() to retrieve error information.</p> <p>Parameters</p> <ul style="list-style-type: none"> • IncludeElements: Boolean - Recursively apply user lock to child elements • IncludeDiagrams: Boolean - Recursively apply user lock to child diagrams • IncludeSubPackages: Boolean - Recursively apply user lock to child Packages
Clone	<p>LDISPATCH</p> <p>Notes: Inserts a copy of the Package into the same parent as the original Package. Returns the newly-created Package.</p>
FindObject (string DottedID)	<p>LPDISPATCH</p> <p>Notes: Returns a Package, element, attribute or operation matching the parameter DottedID.</p> <p>If the DottedID is not found, an error is returned: <i>Can't find matching object.</i></p> <p>Parameters</p> <ul style="list-style-type: none"> • DottedID: String - Is in the form 'object.object.object' where object is replaced by the name of a Package, element attribute or operation; examples include MyNamespace.Class1, CStudent.m_Name, MathClass.DoubleIt(int)
GetLastError ()	<p>String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p>
ReleaseUserLock ()	<p>Boolean</p> <p>Notes: Removes an existing User or Group lock from the Package object. Returns True if the operation is successful; returns False if the operation is unsuccessful. Use GetLastError() to retrieve error information.</p>
ReleaseUserLockRecursive ()	<p>Boolean</p> <p>Notes: Releases user locks and group locks from the Package object, and all of the Packages, diagrams and elements contained within that Package.</p> <p>Returns True if the operation is successful; returns False if the operation is unsuccessful. Use GetLastError() to retrieve error information.</p>
SetReadOnly (boolean ReadOnly, boolean IncludeSubPkgs)	<p>Void</p> <p>Notes: Sets a Package Flag to mark a Package as ReadOnly=1.</p> <p>If Project Security is enabled, the user must have Configure Packages permission to use this method.</p> <p>Throws an exception if the operation fails due to the user not having Configure Packages permission; use 'GetLastError()' to retrieve error information.</p> <p>Parameters</p> <ul style="list-style-type: none"> • ReadOnly: Boolean - Sets or clears the Read Only flag on the Package(s); if: False, any Read Only flag is removed from the Package True, a Read Only flag is applied to the Package • IncludeSubPkgs: Boolean - Indicates whether to set/reset the Read Only flag

	<p>on just the object Package, or on the object Package and all of the nested sub-Packages that it contains; if:</p> <p>False, only the flag on the object Package is set or cleared</p> <p>True, flags are set (or cleared, according to the ReadOnly parameter) for the object Package plus all of the nested sub-Packages that it contains</p> <p>When working with version controlled Packages, the Read Only flag can be applied to Packages whether they are checked-in or checked-out.</p> <p>User Security applies to setting this flag - if you are prevented from editing the Package, you are also prevented from setting the flag.</p>
Update ()	<p>Boolean</p> <p>Notes: Updates the current Package object after modification or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p> <p>Note that a Package object also has an element component that must be taken into account; the Package object contains information about the Package attributes such as hierarchy or contents.</p> <p>The element attribute contains information about, for example, Stereotypes, Constraints or Files - all the attributes of a typical element.</p>
VersionControlAdd (string ConfigGuid, string XMLFile, string Comment, boolean KeepCheckedOut)	<p>Void</p> <p>Notes: Places the Package under version control, using the specified Version Control Configuration and the specified XMI filename.</p> <p>Throws an exception if the operation fails; use GetLastError() to retrieve error information.</p> <p>It is recommended that the Package be saved using Update() before calling VersionControlAdd(), so that any outstanding changes are not lost.</p> <p>Parameters</p> <ul style="list-style-type: none"> • ConfigGuid: String - Name corresponding to the Unique ID of the version control configuration to use • XMLFile: String - Name of the XML file to use for this Package; this filename is relative to the Working Copy folder specified for the Config • Comment: String - Log message that is added to the version controlled file's history (where applicable) • KeepCheckedOut: Boolean - Specify True to add to version control and keep the Package checked-out
VersionControlCheckin (string Comment)	<p>Void</p> <p>Notes: Perform checkin of the version controlled Package (also see VersionControlCheckinEx).</p> <p>Throws an exception if the operation fails; use GetLastError() to retrieve error information.</p> <p>Parameters</p> <ul style="list-style-type: none"> • Comment: String - Log message that is added to the version controlled file's history (where applicable)
VersionControlCheckinEx (string Comment, boolean PreserveCrossPkgRefs)	<p>Void</p> <p>Notes: Perform checkin of the version controlled Package.</p> <p>Throws an exception if the operation fails; use GetLastError() to retrieve error information.</p>

	<p>Parameters</p> <ul style="list-style-type: none"> • Comment: String - Log message that is added to the version controlled file's history (where applicable) • PreserveCrossPkgRefs: Boolean - Flag to indicate whether to preserve or discard pre-existing Cross Package References when checking-in; this parameter overrides the setting in the 'Preferences' dialog, 'XML Specifications' page Unsatisfied cross-Package references are preserved or discarded according to this setting, without prompting the user; see <i>Learn more</i>
VersionControlCheckout (string Comment)	<p>Void</p> <p>Notes: Perform checkout of the version controlled Package.</p> <p>Throws an exception if the operation fails; use GetLastError() to retrieve error information.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Comment: String - Log message that is added to the version controlled file's history (where applicable) <p>When working in an environment that uses a Private Model deployment and your model contains a significant number of cross-Package references, it is recommended that you invoke the Repository.ScanXMIAndReconcile() method from time to time, following the re-importation of controlled Packages - for example, after using Package.VersionControlGetLatest() to update a number of Packages, or after performing a number of Package check-outs.</p>
VersionControlGetLatest (boolean ForceImport)	<p>Void</p> <p>Notes: Updates the local working copy of the Package file associated with the object Package, before re-importing the Package data from the Package file.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ForceImport: Boolean - Used if the Package data in the model is found to be up-to-date with respect to the version controlled Package file; if: <ul style="list-style-type: none"> - False, the Package data that exists in the model is accepted as being up-to-date and no attempt is made to re-import data from the Package file - True, the system re-imports the Package from the Package file regardless <p>See also the version control menu option 'Get Latest'.</p> <p>When working in an environment that uses a Private Model deployment and your model contains a significant number of cross-Package references, it is recommended that you invoke the 'Repository.ScanXMIAndReconcile()' method from time to time, following the re-importation of controlled Packages - for example, after using 'Package.VersionControlGetLatest()' to update a number of Packages, or after performing a number of Package check-outs.</p>
VersionControlGetStatus ()	<p>Long</p> <p>Notes: Returns the version control status of the Package, as recorded in the current project database.</p> <p>Throws an exception if the operation fails; use GetLastError() to retrieve error information.</p> <p>Return value maps to this enumerated type:</p> <pre>enum EnumCheckOutStatus { csUncontrolled = 0, csCheckedIn, csCheckedOutToThisUser,</pre>

	<pre> csReadOnlyVersion, csCheckedOutToAnotherUser, csOfflineCheckedIn, csCheckedOutOfflineByUser, csCheckedOutOfflineByOther, csDeleted, }; </pre> <ul style="list-style-type: none"> • csUncontrolled - Either unable to communicate with the version control provider associated with the Package, or the Package file is unknown to the provider • csCheckedIn - The Package is not checked-out to anybody in the current project database • csCheckedOutToThisUser - The Package is marked as checked-out to the current user, in the current project database • csReadOnlyVersion - The Package is marked as read-only; an earlier revision of the Package has been retrieved from version control • csCheckedOutToAnotherUser - The Package is marked as checked-out in the current project database, by a user other than the current user • csOfflineCheckedIn - The Package is not checked-out to anybody in the current project database; however, the version control configuration associated with the Package was unable to connect to the VC server • csCheckedOutOfflineByUser - The Package was 'checked out' in this database, by this user, whilst disconnected from version control • csCheckedOutOfflineByOther - The Package was checked out in this project database, by another user, whilst disconnected from version control • csDeleted - The Package file has been deleted from version control
VersionControlPutLatest (string CheckInComment)	<p>Void</p> <p>Notes: Perform a checkin of the version controlled Package, whilst keeping the Package checked-out.</p> <p>Throws an exception if the operation fails; use GetLastError() to retrieve error information.</p> <p>When a Package that was previously marked as Checked Out Offline, is successfully 'Put' (checkedin) to version control, that Package's flags are updated to clear the Checked Out Offline indicator.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Comment: String - Log message added to the version controlled file's history (where applicable)
VersionControlRemove ()	<p>Void</p> <p>Notes: Removes version control from the Package.</p> <p>Throws an exception if the operation fails; use 'GetLastError()' to retrieve error information.</p>
VersionControlResynchPackageStatus (boolean ClearSettings)	<p>Notes: Synchronizes the version control status of the single object Package recorded in your current model with the Package status reported by your version control provider.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ClearSettings: Boolean - used if the Package file associated with the specified Package is reported by the version control provider as uncontrolled; if

	<p>ClearSettings is:</p> <p>True, the version control settings are cleared from the Package</p> <p>False, the version control settings remain unchanged</p>
--	---

ProjectIssues Class

A ProjectIssue is a system-level Issue that indicates a problem or risk associated with the system as a whole. ProjectIssues can be accessed using the Repository Issues collection.

Associated table in .EAP file

t_issues

ProjectIssues Attributes

Attribute	Remarks
Category	String Notes: Read/Write The category this issue belongs to.
Date	Date Notes: Read/Write The date the issue item was created.
DateResolved	Date Notes: Read/Write The date the issue was resolved.
Name	String Notes: Read/Write The issue name (that is, the issue itself).
IssueID	Long Notes: Read only The ID of this issue.
Notes	String Notes: Read/Write The associated description of the issue.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Owner	String Notes: Read/Write The owner of the issue.

Priority	String Notes: Read/Write The issue priority - Low, Medium or High.
Resolution	String Notes: Read/Write A description of the resolution.
Resolver	String Notes: Read/Write The name of the person resolving the issue.
Severity	String Notes: Read/Write The issue severity - Low, Medium or High.
Status	String Notes: Read/Write The current status of the issue.

ProjectIssues Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Updates the current Issue object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

ProjectResource Class

A Project Resource is a named person who is available to work on the current project in any capacity. ProjectResources can be accessed using the Repository Resources collection.

Associated table in .EAP file

t_resources

ProjectResource Attributes

Attribute	Remarks
Email	String Notes: The resource's email address.
Fax	String Notes: The resource's fax number.
Mobile	Variant Notes: The resource's mobile number, if available.
Name	String Notes: The name of the resource.
Notes	String Notes: A description of the resource, if appropriate.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Organization	Package Class : String Notes: The organization the resource is associated with.
Phone1	Variant Notes: The resource's main telephone number.
Phone2	Variant Notes: The resource's alternative telephone number.
Roles	String Notes: The roles this resource can play in the current project.

ProjectResource Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Updates the current Resource object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

ProjectRole Class

A ProjectRole object represents a named project role. ProjectRoles can be accessed using the Repository ProjectRole collection.

Associated table in .EAP file

t_projectroles

ProjectRole Attributes

Attribute	Remarks
Description	String Notes: Read/Write The project role item description.
Notes	String Notes: Read/Write Notes about the project role item.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Role	String Notes: Read/Write The project role item name.

ProjectRole Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Updates the current ProjectRole object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

PropertyType Class

A PropertyType object represents a defined property that can be applied to UML elements as a Tagged Value. PropertyTypes can be accessed using the Repository PropertyTypes collection.

Each PropertyType corresponds to one of the predefined Tagged Values for the model.

Associated table in .EAP file

t_propertytypes

PropertyType Attributes

Attribute	Remarks
Description	String Notes: Read/Write A short description of the property.
Detail	String Notes: Read/Write Configuration information for the property.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Tag	String Notes: Read/Write The name of the property (Tag Name).

PropertyType Methods:

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Updates the current PropertyType object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Reference Class

This Interface provides access to the various lookup tables within Enterprise Architect. Use the Repository `GetReferenceList()` method to get a handle to a list.

Valid lists are:

- Diagram
- Element
- Constraint
- Requirement
- Connector
- Status
- Cardinality
- Effort
- Metric
- Scenario
- Status
- Test
- List:DifficultyType
- List:PriorityType
- List:TestStatusType
- List:ConstStatusType

Reference Attributes

Attribute	Remarks
Count	Short Notes: A count of items in the list.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Type	String Notes: The list type (for example, DiagramTypes).

Reference Methods

Method	Remarks
GetAt(short Index)	String

	<p>Notes: Get the item at the specified index.</p> <p>Parameters:</p> <ul style="list-style-type: none">• Index: Short - The index of the item to retrieve from the list
GetLastError()	<p>String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p>
Refresh()	<p>Short</p> <p>Notes: Refresh the current list and return the count of items.</p>

Repository Class

The Repository is the main container of all structures such as models, Packages and elements. You can begin accessing the model iteratively using the Models collection. The Repository also has some convenient methods to directly access the structures without having to locate them in the hierarchy first.

Associated table in .EAP file

<none>

Repository Attributes

Attribute	Remarks
Authors	<p>Collection</p> <p>Notes: Read only</p> <p>This is the system Authors collection containing 0 or more Author objects, each of which can be associated with, for example, elements or diagrams as the item author or owner.</p> <p>Use AddNew(), Delete() and GetAt() to manage Authors.</p>
BatchAppend	<p>Boolean</p> <p>Notes: Read/Write</p> <p>Set this property to True when your automation client has to rapidly insert many elements, operations, attributes and/or operation parameters.</p> <p>Set to False when work is complete.</p> <p>This can result in 10- to 20-fold improvement in adding new elements in bulk.</p>
Clients	<p>Collection</p> <p>Notes: Read only</p> <p>A list of Clients associated with the project. You can modify, delete and add new Client objects using this collection.</p>
ConnectionString	<p>String</p> <p>Notes: Read only</p> <p>The filename/connection string of the current Repository.</p> <p>For a connection string, the DBMS repository type is identified by "DBType=n;" where n is a number corresponding to the DBMS type, as shown:</p> <ul style="list-style-type: none"> 0 - MYSQL 1 - SQLSVR 2 - ADOJET 3 - ORACLE 4 - POSTGRES 5 - ASA 8 - ACCESS2007

	9 - FIREBIRD
Datatypes	<p>Collection</p> <p>Notes: Read only</p> <p>The Datatypes collection. This contains a list of Datatype objects, each representing a data type definition for either data modeling or code generation purposes.</p>
EAEdition	<p>EAEditionTypes</p> <p>Notes: Read only</p> <p>Returns the current level of core licensed functionality available.</p> <p>This property returns Corporate when the edition is Business and Software Engineering, Systems Engineering or Ultimate.</p> <p>Use 'EAEditionEx' to identify which of these extended editions is available.</p>
EAEditionEx	<p>EAEditionTypes</p> <p>Notes: Read only</p> <p>Returns the current level of extended licensed functionality available (Business and Software Engineering, Systems Engineering or Ultimate).</p>
EnableCache	<p>Boolean</p> <p>Notes: Read/Write</p> <p>An optimization for pre-loading Package objects when dealing with large sets of automation objects.</p>
EnableUIUpdates	<p>Boolean</p> <p>Notes: Read/Write</p> <p>Set this property to False to improve the performance of changes to the model; for example, bulk addition of elements to a Package. To reveal changes to the user, call 'Repository.RefreshModelView()'.</p>
FlagUpdate	<p>Boolean</p> <p>Notes: Read/Write</p> <p>Instructs Enterprise Architect to update the Repository with the LastUpdate value.</p>
InstanceGUID	<p>String</p> <p>Notes: Read only</p> <p>The identifier string identifying the Enterprise Architect runtime session.</p>
IsSecurityEnabled	<p>Boolean</p> <p>Notes: Read only</p> <p>Indicates whether User Security is enabled for the current repository.</p>
Issues	<p>Collection</p> <p>Notes: Read only</p> <p>The System Issues list. Contains ProjectIssues objects, each detailing a particular issue as it relates to the project as a whole.</p>
LastUpdate	<p>String</p> <p>Notes: Read only</p>

	The identifier string identifying the Enterprise Architect runtime session and the timestamp for when it was set.
LibraryVersion	Long Notes: Read only The build number of the Enterprise Architect runtime.
Models	Collection of type Package Notes: Read only Models are of type Package and belong to a collection of Packages. This is the top level entry point to an Enterprise Architect project file. Each model is a root node in the Project Browser and can contain items such as Views and Packages. A model is a special form of a Package; it has a ParentID of 0. By iterating through all models, you can access all the elements within the project hierarchy. You can also use the AddNew() function to create a new model. A model can be deleted, but remember that everything contained in the model is deleted as well.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through the Dispatch interface.
ProjectGUID	String Notes: Read only Returns the unique ID for the project.
ProjectRoles	Collection Notes: Read only The system Roles collection containing 0 or more Role objects, each of which can be associated with, for example, elements or diagrams as the item author or owner. Use AddNew(), Delete() and GetAt() to manage Roles.
PropertyTypes	Collection Notes: Read only Collection of Property Types available to the Repository.
Resources	Collection Notes: Read only Contains available ProjectResource objects to assign to work items within the project. Use the 'Add New()', 'Modify()' and 'Delete()' functions to manage resources.
Stereotypes	Collection Notes: Read only The Stereotype collection. A list of Stereotype objects that contain information on a stereotype and the elements it can be applied to.
SuppressEADialogs	Boolean Notes: Read/Write Set this property in the EA_OnPostNewElement broadcast event to control whether

	Enterprise Architect should suppress showing the default 'Properties' dialog to the user when an element is created.
SuppressSecurityDialog	Boolean Notes: Read/Write Suppress the login prompt dialog that appears by default when username and password parameters passed to OpenFile2 are invalid. For use by external automation clients only.
Tasks	Collection Notes: Read only A list of system tasks (to do list). Each entry is a Task Item; you can modify, delete and add new tasks.
Terms	Collection Notes: Read only The Project Glossary Terms. Each Term object is an entry in the Glossary. Add, modify and delete Terms to maintain the Glossary.

Repository Methods

Method	Remarks
ActivateDiagram (long DiagramID)	Notes: Activates an already open diagram (that is, makes it the active tab) in the main Enterprise Architect user interface. Parameters: <ul style="list-style-type: none"> DiagramID: Long - the ID of the diagram to make active
ActivatePerspective (string long)	Boolean Notes: Deprecated - no longer in use.
ActivateTab (string Name)	Notes: Activates an open Enterprise Architect tabbed view. Parameters: <ul style="list-style-type: none"> Name: String - the name of the view to activate
ActivateTechnology (string TechnologyID)	Notes: Activates an enabled MDG Technology. Parameters: <ul style="list-style-type: none"> TechnologyID: String - the ID of the Technology to activate, as assigned in the MDG Technology Wizard
ActivateToolbox (string Toolbox, long Options)	Boolean Notes: Activates a Toolbox page in the GUI. The returned value is reserved for future use. Parameters: <ul style="list-style-type: none"> Toolbox: String - the name of the Toolbox page to activate Options: Long - reserved for future use

AddDefinedSearches (string sXML)	<p>Notes: Used to enter a set of defined searches that last in Enterprise Architect for the life of the application; when Enterprise Architect loads again they must be inserted again by your Add-In.</p> <p>Parameters:</p> <ul style="list-style-type: none"> sXML: String - the XML of the defined searches; you can get this XML by performing an export of the searches from the 'Manage Searches' dialog in Enterprise Architect
AddDocumentationPath (string Name, string Path, long Type)	<p>Notes: Provides an Add-In with the ability to insert a book path into the Enterprise Architect installation directory, to display Learning Center pages on user-authored subjects (such as use of the Add-In).</p> <p>Parameters:</p> <ul style="list-style-type: none"> Name: String - the top-level (root) name for the Learning Center documentation hierarchy for the Add-In (for example, Enterprise Architect) Path: String - the directory path to the folder to contain the Learning Center documentation structure (for example, C:\Program Files (86)\Sparx Systems\EA\Books) Type: Long - reserved for future use; set to 0
AddPerspective (string Perspective, long Options)	<p>Boolean</p> <p>Notes: Deprecated - no longer in use.</p>
AddTab (string TabName, string ControlID)	<p>activeX custom control</p> <p>Notes: Adds an ActiveX custom control as a tabbed window. Enterprise Architect creates a control and, if successful, returns its Unknown pointer, which can be used by the caller to manipulate the control.</p> <p>Parameters:</p> <ul style="list-style-type: none"> TabName: String - used as the tab caption ControlID: String - the ProgID of the control; for example, "CS_AddinFramework.UserControl1"
AddWindow (string WindowName, string ControlID)	<p>activeX custom control</p> <p>Notes: Adds an ActiveX custom control as a window to the Add-Ins docked window. Enterprise Architect creates a control and, if successful, returns its Unknown pointer, which can be used by the caller to manipulate the control.</p> <p>Parameters:</p> <ul style="list-style-type: none"> WindowName: String - used as the window title ControlID: String - the ProgID of the control; for example, "CS_AddinFramework.UserControl1"
AdviseConnectorChange (long ConnectorID)	<p>Notes: Provides an Add-In or automation client with the ability to advise the Enterprise Architect user interface that a particular connector has changed and, if it is visible in any open diagram, to reload and refresh that connector for the user.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ConnectorID: Long - the ID of the connector
AdviseElementChange (long ObjectID)	<p>Notes: Provides an Add-In or automation client with the ability to advise the Enterprise Architect user interface that a particular element has changed and, if it is visible in any open diagram, to reload and refresh that element for the user.</p> <p>Parameters:</p>

	<ul style="list-style-type: none"> ObjectID: Long - the ID of the element
ChangeLoginUser (string Name, string Password)	<p>Boolean</p> <p>Notes: Sets the currently logged on user to be the one specified by a name and password; this logs the user into the repository when security is enabled. If security is not enabled an exception (Security not enabled) is thrown.</p> <p>Parameters:</p> <ul style="list-style-type: none"> Name: String - the name of the user Password: String - the password of the user
ClearAuditLogs (Object StartDateTime, Object EndDateTime)	<p>Boolean</p> <p>Notes: Clears all Audit Logs from the model.</p> <p>If StartDateTime and EndDateTime are not null then only log items that fall into this period are cleared.</p> <p>Returns True for success, False for failure.</p> <ul style="list-style-type: none"> This method cannot be undone; it is strongly advised that you call 'SaveAuditLogs' first to backup the logs This method might fail if the user logged into the model does not have the correct access permission <p>Parameters:</p> <ul style="list-style-type: none"> StartDateTime: Variant (DateTime) - the earliest date and time of log entries to clear EndDateTime: Variant (DateTime) - the latest date and time of log entries to clear
ClearOutput (string Name)	<p>Notes: Removes all the text from a tab in the System Output window.</p> <p>Parameters:</p> <ul style="list-style-type: none"> Name: String - the name of the tab to remove text from
CloseAddins ()	<p>Notes: Called by automation controllers to ensure that Add-Ins created in .NET do not linger after all controller references to Enterprise Architect have been cleared.</p>
CloseDiagram (long DiagramID)	<p>Notes: Closes a diagram in the current list of diagrams that Enterprise Architect has open.</p> <p>Parameters:</p> <ul style="list-style-type: none"> DiagramID: Long - the ID of the diagram to close
CloseFile ()	<p>Notes: Closes any open file.</p>
CreateDocumentGenerator()	<p>Document Generator</p> <p>Notes: Returns a pointer to the EA.DocumentGenerator interface.</p>
CreateModel (CreateModelType CreateType,string FilePath, long ParentWnd)	<p>Boolean</p> <p>Notes: Creates a new .eap model file based on the standard Enterprise Architect Base model, or a shortcut .eap based on a provided SQL connection.</p> <p>Returns True when the new file is created, otherwise returns False.</p> <p>Parameters:</p> <ul style="list-style-type: none"> CreateType: CreateModelType - Specify whether to make a new copy of the EABase.eap model, or create a .eap file shortcut to a DBMS repository; the latter option requires a dialog to be opened for the user to provide SQL

	<p>connection details</p> <ul style="list-style-type: none"> • FilePath: String - Destination for new .eap file • ParentWnd: Long - Window handle to act as the parent for the 'SQL connection' dialog; only required when using cmEAPFromSQLRepository
CreateOutputTab (string Name)	<p>Notes: Creates a tab in the System Output window.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Name: String - the name of the tab to create
DeletePerspective (string Perspective, long Options)	<p>Boolean</p> <p>Notes: Deprecated - no longer in use.</p>
DeleteTechnology (string ID)	<p>Boolean</p> <p>Notes: Removes a specified MDG Technology resource from the repository. Returns True if the technology is successfully removed from the model. Returns False otherwise.</p> <ul style="list-style-type: none"> • This applies to technologies imported into pre-7.0 versions of Enterprise Architect (imported technologies), not to technologies referenced in version 7.0 and later (referenced technologies) <p>Parameters:</p> <ul style="list-style-type: none"> • ID: String - the ID of the technology
EnsureOutputVisible (string Name)	<p>Notes: Checks that a specified tab in the System Output window is visible to the user. The System Output window is made visible if it is hidden.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Name: String - the name of the tab to make visible
ExecutePackageBuildScript (long ScriptOptions, string PackageGuid)	<p>Notes: Helps you to run the active Package build script based on your current selection in the Project Browser. You can also run a script by passing in the Package GUID.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ScriptOptions: Long - the script type; can be any one of these numerical values: <ul style="list-style-type: none"> 1 = Build 2 = Test 3 = Run 4 = Create Workbench Instance 5 = Debug • PackageGuid: String - the ID of the Package for which to run the script
Exit	<p>Notes: Shuts down Enterprise Architect immediately. Used by .NET programmers where the garbage collector does not immediately release all referenced COM objects.</p>
ExtractImagesFromNote (string Notes, string WriteImagePath, string RelativeImagePath)	<p>String</p> <p>Notes: Writes any Image Manager links to the WriteImagePath directory. Returns a modified notes text, which contains links to the images using the RelativeImagePath parameter.</p> <p>Parameters:</p>

	<ul style="list-style-type: none"> • Notes: String - the notes of the selected Package, diagram or element • WriteImagePath: String - the path where the image file links will be stored; this path must exist • RelativeImagePath: String - the path to be inserted into the modified string indicating where the images can be found (for example, "..\images\")
GetActivePerspective ()	<p>String</p> <p>Notes: Deprecated - no longer in use.</p>
GetAttributeByGuid (string Guid)	<p>Attribute</p> <p>Notes: Returns a pointer to an attribute in the repository, located by its GUID. This is usually found using the AttributeGUID property of an attribute.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Guid: String - the GUID of the attribute to locate
GetAttributeByID (string Id)	<p>Attribute</p> <p>Notes: Returns a pointer to an attribute in the repository, located by its ID. This is usually found using the AttributeID property of an attribute.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Id: String - the ID of the attribute to locate
GetConnectorByGuid (string Guid)	<p>Connector</p> <p>Notes: Returns a pointer to a connector in the repository, located by its GUID. This is usually found using the ConnectorGUID property of a connector.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Guid: String - the GUID of the connector to locate
GetConnectorByID (long ConnectorID)	<p>Connector</p> <p>Notes: Searches the repository for a connector with a specific ID.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ConnectorID: Long - the ID of the connector to locate
GetContextItem (object Item)	<p>ObjectType</p> <p>Notes: Sets a pointer to an item in context within Enterprise Architect.</p> <p>Also returns the corresponding ObjectType.</p> <p>For additional information about ContextItems and the supported ObjectTypes see the 'GetContextItemType' method.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Item: Object - the item to point to
GetContextItemType ()	<p>ObjectType</p> <p>Notes: Returns the ObjectType of an item in context within Enterprise Architect. A ContextItem is defined as an item selected anywhere within the Enterprise Architect GUI including:</p> <ul style="list-style-type: none"> • An item selected in the Project Browser • An item selected in an open diagram • An item selected in certain dialogs, such as the attribute 'Properties' dialog <p>The supported ObjectTypes can be any one of these values:</p>

	<ul style="list-style-type: none"> • otElement • otPackage • otDiagram • otAttribute • otMethod • otConnector
GetContextObject ()	<p>Object</p> <p>Notes: Returns the current context Object.</p>
GetCounts ()	<p>String</p> <p>Notes: Returns a set of counts from a number of tables within the base Enterprise Architect repository. These can be used to determine whether records have been added or deleted from the tables for which information is retrieved.</p>
GetCurrentDiagram ()	<p>Diagram</p> <p>Notes: Returns a selected diagram.</p>
GetCurrentLoginUser (boolean GetGuid)	<p>String</p> <p>Notes: If security is not enabled in the repository, an error is generated.</p> <p>If 'GetGuid' is True, a GUID generated by Enterprise Architect representing the user is returned; otherwise the text as entered in System Users/User Details/Login is returned.</p>
GetDiagramByGuid (string Guid)	<p>Diagram</p> <p>Notes: Returns a pointer to a diagram using the global reference ID (global ID). This is usually found using the diagram GUID property of an element, and stored for later use to open an diagram without using the collection GetAt() function.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Guid: String - the GUID of the diagram to locate
GetDiagramByID (long DiagramID)	<p>Diagram</p> <p>Notes: Gets a pointer to a diagram using an absolute reference number (local ID). This is usually found using the DiagramID property of an element, and stored for later use to open a diagram without using the collection GetAt() function.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • DiagramID: Long - the ID of the diagram to locate
GetElementByGuid (string Guid)	<p>Element</p> <p>Notes: Returns a pointer to an element in the repository, using the element's GUID reference number (global ID). This is usually found using the ElementGUID property of an element, and stored for later use to open an element without using the collection 'GetAt ()' function.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Guid: String - the GUID of the element to locate
GetElementByID (long ElementID)	<p>Element</p> <p>Notes: Gets a pointer to an element using an absolute reference number (local ID). This is usually found using the ElementID property of an element, and stored for later use to open an element without using the collection GetAt () function.</p>

	<p>Parameters:</p> <ul style="list-style-type: none"> ElementID: Long - the ID of the element to locate
<p>GetElementsByQuery (string QueryName, string SearchTerm)</p>	<p>Collection (of type Element)</p> <p>Notes: Helps you to run a search in Enterprise Architect, returning the result as a collection.</p> <p>For example: GetElementsByQuery('Simple','Class1'), where the results list elements with 'Class1' in the 'Name' and 'Notes' fields.</p> <p>Parameters:</p> <ul style="list-style-type: none"> QueryName: String - the name of the search to run, for example 'Simple' SearchTerm: String - the term to search for
<p>GetElementSet (string IDList, long Options)</p>	<p>Collection (of type Element)</p> <p>Notes: Returns a set of elements as a collection based on a comma-separated list of ElementID values. By default, if no values are provided in the IDList parameter, all objects for the entire project are returned.</p> <p>Parameters</p> <ul style="list-style-type: none"> IDList: String - a comma-separated list of ElementID values Options: Long - modifies default behavior of this method <ul style="list-style-type: none"> 1 - Returns empty collection when empty IDList parameter is given 2 - Use IDList string as an SQL query to populate this collection
<p>GetFieldFromFormat (string Format, string Text)</p>	<p>String</p> <p>Notes: Converts a field from your preferred format to Enterprise Architect's internal format; returns the field in that format.</p> <p>Parameters:</p> <ul style="list-style-type: none"> Format: String - The format to convert the field from; valid formats are: <ul style="list-style-type: none"> - HTML - Full HTML - RTF - Rich Text Format - TXT - Plain text Text: String - The field to be converted
<p>GetFormatFromField (string Format, string Text)</p>	<p>String</p> <p>Notes: After accessing a field that contains formatting, use this method to convert it to your preferred format; returns the field in the format specified.</p> <p>Parameters:</p> <ul style="list-style-type: none"> Format: String - The format to convert the field to; valid formats are: <ul style="list-style-type: none"> - HTML - Full HTML - RTF - Rich Text Format - TXT - Plain text Text: String - The field to be converted
<p>GetGapAnalysisMatrix ()</p>	<p>String</p> <p>Notes: Read Only</p> <p>Returns all Gap Analyses as an XML document.</p>
<p>GetLastError ()</p>	<p>String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p>

GetLocalPath (string Type, string Path)	<p>String</p> <p>Notes: Returns the expanded local file path for code generated from an element, with reference to the Type and Path defined in the 'Local Paths' dialog.</p> <p>Parameters:</p> <ul style="list-style-type: none"> Type: String - the coding language for the element, such as Java, C or C++ Path: String - the local path to be expanded; for example: %Desk%\Javacode\Motor.java <p>For example:</p> <p>Repository.GetLocalPath (Java, %Desk%\Javacode\Motor.java)</p> <p>This could return:</p> <p>C:\Users\fbloggs\Desktop\Javacode\Motor.java.</p>
GetMailInterface ()	<p>MailInterface</p> <p>Notes: Returns an instance of the EA.MailInterface; use this interface to automate the process of creating and sending Model Mail messages.</p>
GetMethodByGuid (string Guid)	<p>Method</p> <p>Notes: Returns a pointer to a method in the repository; this is usually found using the MethodGUID property of a method.</p> <p>Parameters:</p> <ul style="list-style-type: none"> Guid: String - the GUID of the method to look for
GetMethodById (string Id)	<p>Method</p> <p>Notes: Returns a pointer to a method in the repository; this is usually found using the MethodID property of a method.</p> <p>Parameters:</p> <ul style="list-style-type: none"> Id: String - the ID of the method to look for
GetPackageByGuid (string Guid)	<p>Package</p> <p>Notes: Returns a pointer to a Package in the repository using the Package's GUID reference number (global ID). This is usually found using the PackageGUID property of the Package.</p> <p>Each Package in the model also has an associated element with the same GUID, so if you have an element with Type="Package" then you can load the Package by calling:</p> <p>GetPackageByGuid(Element.ElementGUID)</p> <p>Parameters:</p> <ul style="list-style-type: none"> Guid: String - the GUID of the Package to look for
GetPackageById (long PackageID)	<p>Package</p> <p>Notes: Get a pointer to a Package using an absolute reference number (local ID). This is usually found using the PackageID property of an Package, and stored for later use to open a Package without using the collection GetAt () function.</p> <p>Parameters:</p> <ul style="list-style-type: none"> PackageID: Long - the ID of the Package to locate
GetProjectInterface ()	<p>Project</p> <p>Notes: Returns a pointer to the EA.Project interface (the XML-based automation server for Enterprise Architect). Use this interface to work with Enterprise Architect using XML, and also to access utility functions for loading diagrams,</p>

	running reports and so on.
GetReferenceList (string Type)	<p>Reference</p> <p>Notes: Uses the list type to get a pointer to a Reference List object.</p> <p>Parameters:</p> <ul style="list-style-type: none"> Type: String - specifies the list type to get; valid list types are: Diagram Element Constraint Requirement Connector Status Cardinality Effort Metric Scenario Status Test List:DifficultyType List:PriorityType List:TestStatusType List:ConstStatusType
GetRelationshipMatrix ()	<p>String</p> <p>Notes: Returns an XML document (as a string), containing definitions of all Relationship Matrix profiles saved in the current model.</p>
GetTechnologyVersion (string ID)	<p>String</p> <p>Notes: Returns the version of a specified MDG Technology resource.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ID: String - the specified technology ID
GetTreeSelectedElements ()	<p>Collection</p> <p>Notes: Returns the set of elements currently selected in the Project Browser as a collection.</p>
GetTreeSelectedItem (object SelectedItem)	<p>ObjectType</p> <p>Notes: Gets an object variable and type corresponding to the currently selected item in the tree view.</p> <p>To use this function, create a generic object variable and pass this as the parameter. Depending on the return type, cast it to a more specific type.</p> <p>The object passed back through the parameter can be a Package, element, diagram, attribute or operation object.</p> <p>Parameters:</p> <ul style="list-style-type: none"> SelectedItem: Object - the object to get the variable and type for
GetTreeSelectedItemType ()	<p>ObjectType</p> <p>Notes: Returns the type of the object currently selected in the tree. One of:</p> <ul style="list-style-type: none"> otDiagram otElement otPackage

	<ul style="list-style-type: none"> • otAttribute • otMethod
GetTreeSelectedObject ()	<p>Object</p> <p>Notes: The related method GetTreeSelectedItem () has an output parameter that is inaccessible by some scripting languages. As an alternative, this method provides the selected item through the return value.</p>
GetTreeSelectedPackage ()	<p>Package</p> <p>Notes: Returns the Package in which the currently selected tree view object is contained.</p>
HasPerspective (string Perspective)	<p>String</p> <p>Notes: Deprecated - no longer in use.</p>
HideAddinWindow ()	<p>Notes: Hides the docked Add-In window.</p>
ImportPackageBuildScripts (string PackageGuid, string BuildScriptXML)	<p>Notes: Imports build scripts into a Package in Enterprise Architect.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • PackageGuid: String - the GUID of the Package into which to import the build scripts • BuildScriptXML: String - the build script XML data, which you can export from within Enterprise Architect
ImportTechnology (string Technology)	<p>Boolean</p> <p>Notes: Installs a given MDG Technology resource into the repository.</p> <p>Returns True if the technology is successfully loaded into the model. Otherwise returns False.</p> <p>This applies to technologies imported into pre-7.0 versions of Enterprise Architect (imported technologies), not to technologies referenced in version 7.0 and later (referenced technologies).</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Technology: String - the contents of the technology resource file
InvokeConstructPicker (string ElementFilter)	<p>String</p> <p>Notes: Invokes the 'Select <Item>' dialog with filters on the object type and, optionally, stereotype. Returns the ElementID of the selected object, or 0 if no object was selected when the dialog was closed.</p> <p>For example:</p> <pre>elementid=Repository.InvokeConstructPicker("IncludedTypes=Class,Component;Stereotype=foo,bar")</pre> <p>In this example, the 'Select <item>' dialog will allow the user to select any Class or Component element in the model that has a stereotype of 'foo' or 'bar'. The 'IncludedTypes' and 'Stereotype' filters are separated by a semi-colon.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ElementFilter: String - specifies which elements or Packages are to be made available for selection, based on element types and stereotypes identified by the IncludedTypes and Stereotype filters <ul style="list-style-type: none"> - IncludedTypes - (mandatory) comma separated list of element types that can be selected in the dialog; for

	<p>example:</p> <p>Package,Class,Component</p> <ul style="list-style-type: none"> - MultiSelect - (optional) when set to True ("MultiSelect=True;") allows the Construct picker to select multiple elements - Selection (optional) - list of comma-separated element GUID's that will be selected by default - GetNext (optional) - returns the next ID in the list of selected elements, or 0 when no more are available; this option will not display a dialog and assumes the first call was made with MultiSelect=True; - Stereotype - (optional) comma separated list of stereotypes that can be selected in this dialog <p>Do not use leading or trailing spaces between element type or stereotype values. Parameter values must be written with the correct case; element type names are also case sensitive.</p> <p>Example:</p> <pre>val = Repository.InvokeConstructPicker("IncludedTypes=Class;MultiSelect=True;"); while(val != 0) { val = Repository.InvokeConstructPicker("GetNext=True;"); }</pre>
InvokeFileDialog (string FilterString, long Filterindex, long Flags)	<p>String</p> <p>Notes: Opens a standard 'Open File' dialog and returns a string containing the full path to the selected file on success. Returns an empty string if the dialog was canceled.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • FilterString: String - list of file type filters. • Filterindex: Long - one-based index of the filter to be used by default • Flags: Long - additional bit flags used to initialize the file dialog; see OPENFILENAME structure in MSDN documentation for accepted values
IsTabOpen (string TabName)	<p>String</p> <p>Notes: Checks whether a named Enterprise Architect tabbed view is open and active. This includes open diagram windows or custom controls added using 'Repository.AddTab ()'.</p> <p>Returns:</p> <ul style="list-style-type: none"> • 2 to indicate that a tab is open and active (top-most) • 1 to indicate that it is open but not top-most, or • 0 to indicate that it is not visible at all <p>Parameters:</p> <ul style="list-style-type: none"> • TabName: String - the name of the tab to check for; TabName is case sensitive
IsTechnologyEnabled (string ID)	<p>Boolean</p> <p>Notes: Checks whether a specified technology is enabled in Enterprise Architect. Returns True if the MDG Technology resource is enabled. Otherwise returns False.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ID: String - the technology ID to check for, from:

	id="UML2" name="Basic UML 2 Technology" id="EAExtended" name="Core Extensions" id="262139" name="MDG Technology Builder" id="DatabaseEngineering" name="Database Engineering" id="ArcGIS" name="ArcGIS" id="BRM" name="Business Rule Model" id="CODEENG" name="CodeEngineering" id="ERD" name="Entity Relationship Diagram" id="GML" name="GML" id="SoaML" name="SoaML" id="SysML1.1" name="SysML 1.1" id="SysML1.2" name="SysML 1.2" id="SysML1.3" name="SysML 1.3" id="SYSENG" name="System Engineering" id="Win32UI" name="Win32 User Interface Modeling"
IsTechnologyLoaded (string ID)	Boolean Notes: Checks whether a specified technology is loaded into the repository. Returns True if the MDG Technology resource is loaded into the repository. Otherwise returns False. Parameters: <ul style="list-style-type: none"> ID: String - the technology ID to check for
LoadAddins ()	Notes: Loads all Add-Ins from a repository when Enterprise Architect is opened from automation.
OpenDiagram (long DiagramID)	Notes: Provides a method for an automation client or Add-In to open a diagram. The diagram is added to the tabbed list of open diagrams in the main Enterprise Architect view. Parameters: <ul style="list-style-type: none"> DiagramID: Long - the ID of the diagram to open
OpenFile (string Filename)	Boolean Notes: This is the main point for opening an Enterprise Architect project file from an automation client, and working with the contained objects. If the required project is a DBMS repository, and you have created a shortcut .eap file containing the database connection string, you can call this shortcut file to access the DBMS repository. You can also connect to a SQL database by passing in the connection string itself instead of a filename. A valid connection string can be obtained from the 'Open Project' dialog by selecting a recently opened SQL repository. Parameters: <ul style="list-style-type: none"> Filename: String - the filename of the Enterprise Architect project to open
OpenFile2 (string FilePath, string Username, string Password)	Boolean Notes: As for 'OpenFile ()' except this provides for the specification of a password. Parameters: <ul style="list-style-type: none"> Filepath: String - the file path of the Enterprise Architect project to open Username: String - the user login ID Password: String - the user password
RefreshModelView (long	Notes: Reloads a Package or the entire model, updating the user interface.

PackageID)	<p>Parameters:</p> <ul style="list-style-type: none"> PackageID: Long - the ID of the Package to reload: if 0, the entire model is reloaded; if a valid Package ID, only that Package is reloaded
RefreshOpenDiagrams (boolean FullReload)	<p>Notes: Reloads the diagram contents for all open diagrams from the repository.</p> <p>Parameters:</p> <ul style="list-style-type: none"> FullReload: Boolean - if False only the contents of element compartments are reloaded; if True the full content of each diagram is reloaded
ReloadDiagram (long DiagramID)	<p>Notes: Reloads a specified diagram. This would commonly be used to refresh a visible diagram after code import/export or other batch process where the diagram requires complete refreshing.</p> <p>Calling this method within a call to <i>EA_OnNotifyContextItemModified</i> is not supported</p> <p>Parameters:</p> <ul style="list-style-type: none"> DiagramID: Long - the ID of the diagram to be reloaded
ReloadPackage (long PackageID)	<p>Notes: Reloads a Package and its open child diagrams.</p> <p>Parameters:</p> <p>PackageID: Long - The ID of the Package to reload; if a valid Package ID, only that Package is reloaded.</p>
RemoveOutputTab (string Name)	<p>Notes: Removes a specified tab from the System Output window.</p> <p>Parameters:</p> <ul style="list-style-type: none"> Name: String - the name of the tab to be removed
RemoveWindow (string WindowName)	<p>Boolean</p> <p>Notes: Removes an Add-In window that matches the specified WindowName.</p> <p>Parameters:</p> <ul style="list-style-type: none"> WindowName: String - the name of the window to remove
RepositoryType ()	<p>String</p> <p>Notes: Returns the currently open database/repository type.</p> <p>Can return one of these values:</p> <ul style="list-style-type: none"> JET (.EAP file, MS Access 97 to 2013 format) FIREBIRD ACCESS2007 (.accdb file, MS Access 2007+ format) ASA (Sybase SQL Anywhere) SQLSVR (Microsoft SQL Server) MYSQL (MySQL) ORACLE (Oracle) POSTGRES (PostgreSQL)
RunModelSearch (string sQueryName, string sSearchTerm, string sSearchOptions, string sSearchData)	<p>Notes: Runs a search, displaying the results in Enterprise Architect's Model Search window.</p> <p>Parameters:</p> <ul style="list-style-type: none"> sQueryName: String - the name of the search to run, for example Simple sSearchTerm: String - the term to search for

	<ul style="list-style-type: none"> • sSearchOptions: String - currently not being used • sSearchData: String - a list of results in the form of XML, which is appended onto the result list in Enterprise Architect - see the <i>XML Format</i> topic; this parameter is not mandatory so pass in an empty string to run the search as per normal
SaveAllDiagrams ()	Notes: Saves all open diagrams.
SaveAuditLogs (string FilePath, object StartDateTime, object EndDateTime)	<p>Boolean</p> <p>Notes: Saves the Audit Logs contained within a model to a specified file.</p> <p>If 'StartDateTime' and 'EndDateTime' are not null then only log items that fall into this period are saved.</p> <p>Returns True for success, False for failure.</p> <ul style="list-style-type: none"> • This might fail if the user logged into the model does not have the correct access permission <p>Parameters:</p> <ul style="list-style-type: none"> • FilePath: String - the file to save the Audit Logs to • StartDateTime: Variant (DateTime) - the earliest date and time of log entries to save • EndDateTime: Variant (DateTime) - the latest date and time of log entries to save
SaveDiagram (long DiagramID)	<p>Notes: Saves an open diagram; assumes the diagram is open in the main user interface Tab list.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • DiagramID: Long - the ID of the diagram to save
ScanXMIAndReconcile ()	<p>Notes: Scans the Package XMI files associated with each of the project's controlled Packages and restores any diagram objects or cross-references that are detected as missing from the project.</p> <p>This function is useful in team environments where each user maintains their own private copy of the model database (that is, multiple private EAP files) and model updates are propagated through the use of controlled Packages; it provides no benefit when the model is hosted in a single shared database that is accessed by all team members.</p> <p>Each controlled Package is compared with its associated XMI file and, if the cross-reference information in the model does not match the XMI, Enterprise Architect updates the model with the information from the XMI and records the update in the System Output window.</p> <p>You can roll back such updates by right-clicking on the entry in the System Output window and selecting the 'Rollback Update' option (or 'Rollback Selected Updates' if multiple entries are selected).</p> <p>Closing the model clears the entries in the System Output window; an entry in this window is also cleared as and when you roll-back the update for it.</p> <p>This functionality is invoked automatically as part of the 'Get All Latest' operation.</p> <p>When working in an environment that uses a Private Model deployment and your model contains a significant number of cross-Package references, it is recommended that you invoke this function from time to time, following the re-importation of controlled Packages - for example, after using 'Get Latest' to update a number of Packages, or after performing a number of Package check-outs.</p> <p>As a general rule, avoid running this function while you have uncommitted changes in your model. Generally, you:</p>

	<ul style="list-style-type: none"> • Check-out a number of Packages • Invoke 'ScanXMIAndReconcile' • Make your modifications • Commit any outstanding changes before you check-out more Packages and run 'ScanXMIAndReconcile' again
ShowAddinWindow (string TabName)	<p>Boolean</p> <p>Notes: Shows the docked Add-In window on the specified page. Returns True if a tab of the specified name is now displayed.</p> <p>Parameters</p> <ul style="list-style-type: none"> • TabName: String - specifies the tab
ShowDynamicHelp (string Topic)	<p>Notes: Shows a Help topic as a view.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Topic: String - specifies the Help topic
ShowInProjectView (object Item)	<p>Notes: Selects a specified object in the Project Browser.</p> <p>Accepted object types are Package, Element, Diagram, Attribute, and Method; an exception is thrown if the object is of an invalid type.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Item: Object - the object to highlight
ShowWindow (long Show)	<p>Notes: Shows or hides the Enterprise Architect User Interface.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Show: Long
SQLQuery (string SQL)	<p>String</p> <p>Notes: Enables execution of a SQL select statement against the current repository. Returns an XML formatted string value of the resulting record set.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • SQL: String - contains the SQL Select statement
SynchProfile (string Profile, string Stereotype)	<p>Boolean</p> <p>Notes: Synchronizes Tagged Values and constraints of a UML Profile item using the 'Synch Profiled Elements' dialog.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Profile: String - the name of the profile that contains the stereotype • Stereotype: String - the name of the profile stereotype for which the default tags and constraints are to be synchronized
VersionControlResynchPkgStatuses (boolean ClearSettings)	<p>Notes: Synchronizes the version control status of each version controlled Package within the current model with the status reported by your version control provider.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ClearSettings: Boolean <ul style="list-style-type: none"> - if True, clear the version control settings from Packages that are reported by the version control provider as uncontrolled - if False, leave the version control settings unchanged for Packages reported as uncontrolled

WriteOutput (string Name, string Output, long ID)	<p>Notes: Writes text to a specified tab in the System Output window, and associates the text with an ID.</p> <p>Parameters:</p> <ul style="list-style-type: none">• Name: String - specifies the tab on which to display the text• Output: String - specifies the text to display• ID: Long - specifies a numeric ID value to associate with this output item for further handling by Add-Ins; can be set to 0 if no handling is required
---	--

Stereotype Class

The Stereotype element corresponds to a UML stereotype, which is an extension mechanism for varying the behavior and type of a model element. Use the Repository Stereotypes collection to add new elements and delete existing ones.

Associated table in .EAP file

t_stereotypes

Stereotype Attributes

Attribute	Description
AppliesTo	String Notes: Read/Write A reference to the stereotype Base Class; that is, which element it applies to.
MetafileLoadPath	String Notes: Read/Write The path to an associated metafile. The Automation Interface does not yet support loading metafiles. To do this you must use the 'Stereotype' tab of the 'UML Types' dialog in Enterprise Architect.
Notes	String Notes: Read/Write. Notes about the stereotype.
Name	String Notes: Read/Write The stereotype name, which appears in the Stereotype drop list for elements that match the AppliesTo attribute.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
StereotypeGUID	String Notes: Read/Write A unique identifier for stereotype, generally set and maintained by Enterprise Architect.
Style	String Notes: Read/Write An additional style specifier for the stereotype.
VisualType	String

	<p>Notes: Read/Write</p> <p>Indicates an inbuilt visual style associated with a stereotype.</p> <p>Not currently implemented.</p>
--	---

Stereotype Methods

Method	Description
GetLastError()	<p>String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p>
Update()	<p>Boolean</p> <p>Notes: Updates the current stereotype object after modification or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>

Task Class

A Task is an entry in the System ToDo list. Tasks can be accessed using the Repository Tasks collection.

Associated table in .EAP file

t_tasks

Task Attributes

Attribute	Remarks
ActualTime	Long Notes: Read/Write The time already expended on the task, in hours, days or other units.
AssignedTo	String Notes: Read/Write The person this task is assigned to; that is, the responsible resource.
EndDate	Date Notes: Read/Write The date the task is scheduled to finish.
History	String Notes: Read/Write A memo field to hold, for example, task history or notes.
Name	Variant Notes: Read/Write The task name.
Notes	Variant Notes: Read/Write A description of the task.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Owner	String Notes: Read/Write The task owner.
Percent	Long

	Notes: Read/Write The percentage completion of the task.
Phase	String Notes: Read/Write The phase of the project the task relates to.
Priority	String Notes: Read/Write The priority of this task.
StartDate	Date Notes: Read/Write The date the task is to start.
Status	Variant Notes: Read/Write The current status of the task.
TaskID	Long Notes: Read only The local ID of the task.
TotalTime	Long Notes: Read/Write The total expected time the task might run, in hours, days or some other unit.
Type	String Notes: Read/Write Sets or returns a string representing the type.

Task Methods

Method	Type
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Updates the current Task object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Term Class

A Term object represents one entry in the system glossary. Terms can be accessed using the Repository Terms collection.

Associated table in .EAP file

t_glossary

Term Attributes

Attribute	Remarks
Meaning	String Notes: Read/Write The description of the term; its meaning.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Term	String Notes: Read/Write The glossary item name.
TermID	Long Notes: Read only A local ID number to identify the term in the model.
Type	String Notes: Read/Write The type this term applies to (for example, business or technical).

Term Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Refresh	Void Notes: Forces Enterprise Architect to reload the Glossary terms from the database. If an element is selected, it will have to be re-selected before the 'Note' fields and

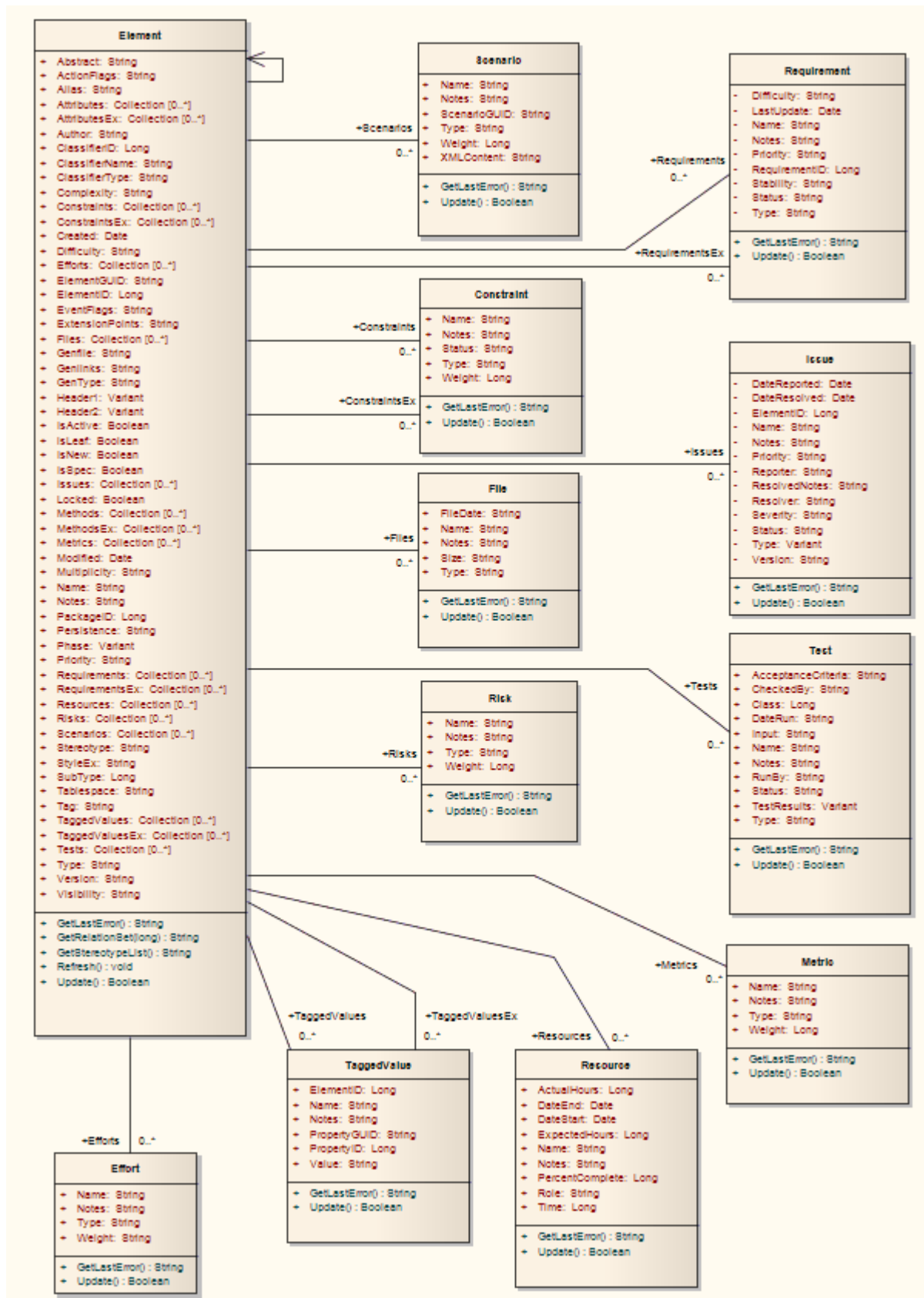
	windows reflect the updated Glossary terms.
Update()	<p>Boolean</p> <p>Notes: Updates the current Term object after modification or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>

Element Package

The Element Package contains information about an element and its associated extended properties such as testing and project management information. An element is the basic item in an Enterprise Architect model. Classes, Use Cases and Components are all different types of UML element.

This diagram illustrates the relationships between an element and its associated extended information. The related information is accessed through the collections owned by the element (for example, Scenarios and Tests). It also includes a full description of the element object (the basic model structural unit).

Example



Constraint Class

A Constraint is a condition imposed on an element. Constraints are accessed through the Element Constraints collection.

Associated table in .EAP file

t_objectconstraints

Constraint Attributes

Attribute	Remarks
Name	String Notes: Read/Write The name of the constraint (that is, the constraint).
Notes	String Notes: Read/Write Notes about the constraint.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
ParentID	Long Notes: Read only The ElementID of the element to which this constraint applies.
Status	String Notes: Read/Write The current status of the constraint.
Type	String Notes: Read/Write The constraint type.
Weight	Long Notes: Read/Write A weighting factor.

Constraint Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Update the current Constraint object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Effort Class

An Effort is a named item with a weighting that can be associated with an element for purposes of building metrics about the model. Efforts are accessed through the Element Efforts collection.

Associated table in .EAP file

t_objecteffort

Effort Attributes

Attribute	Remarks
Name	String Notes: Read/Write The name of the effort.
Notes	String Notes: Read/Write Notes about the effort.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Type	String Notes: Read/Write The effort type.
Weight	Long Notes: Read/Write A weighting factor.
Weight2	Float Notes: Read/Write A weighting factor.

Effort Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in

	relation to this object.
Update()	<p>Boolean</p> <p>Notes: Update the current Effort object after modification or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>

Element Class

An Element is the main modeling unit, corresponding to (for example) a Class, Use Case, Node or Component. You create new elements by adding to the Package Elements collection. Once you have created an element, you can add it to the DiagramObject Class of a diagram to include it in the diagram.

Elements also have a collection of connectors. Each entry in this collection indicates a relationship to another element.

There are also some extended collections for managing additional information about the element, including properties such as Tagged Values, Issues, Constraints and Requirements.

Associated table in .EAP file

t_object

Element Attributes

Attribute	Remarks
Abstract	String Notes: Read/Write Indicates if the element is Abstract (1) or Concrete (0).
ActionFlags	String Notes: Read/Write A structure to hold flags concerned with Action semantics.
Alias	String Notes: Read/Write An optional alias for this element.
AssociationClassConnector ID	Long Notes: Read only If the element is an AssociationClass, AssociationClassConnectorID contains the Connector ID of the respective Association connector.
Attributes	Collection Notes: Read only A collection of Attribute objects for the current element; use the AddNew and Delete functions to manage attributes.
AttributesEx	Collection Notes: Read only A collection of Attribute objects belonging to the current element and its parent elements.
Author	String Notes: Read/Write

	The element author.
BaseClasses	Collection Notes: Read only A list of Base Classes for this element, presented as a collection for convenience.
ClassifierID	Long Notes: Deprecated See 'ClassifierID'.
ClassifierID	Long Notes: Read/Write The ElementID of a Classifier associated with this element; that is, the base type. Only valid for instance type elements (such as Object or Sequence).
ClassifierName	String Notes: Read/Write Name of associated Classifier (if any).
ClassifierType	String Notes: Read only Type of associated Classifier.
Complexity	String Notes: Read/Write A complexity value indicating how complex the element is; used for metric reporting and estimation. Valid values are: 1 for Easy, 2 for Medium, 3 for Difficult.
CompositeDiagram	Diagram Notes: Read only If the element is Composite, returns its associated diagram; otherwise returns null.
Connectors	Collection Notes: Read only Returns a collection containing the connectors to other elements.
Constraints	Collection Notes: Read only A collection of Constraint objects.
ConstraintsEx	Collection Notes: Read only Collection of Constraint objects belonging to the current element and its parent elements.
Created	Date Notes: Read/Write

	The date the element was created.
CustomProperties	<p>Collection</p> <p>Notes: Read only</p> <p>List of advanced properties for an element.</p> <p>The collection of advanced properties differs depending on element type; for example, an Action and an Activity have different advanced properties.</p> <p>Currently only editable from the user interface.</p>
Diagrams	<p>Collection</p> <p>Notes: Read only</p> <p>Returns a collection of sub-diagrams (child diagrams) attached to this element as seen in the tree view.</p>
Difficulty	<p>String</p> <p>Notes: Read/Write</p> <p>A difficulty level associated with this element for estimation/metrics; only useable for Requirement, Change and Issue element types, otherwise ignored.</p> <p>Valid values are: Low, Medium, High.</p>
Efforts	<p>Collection</p> <p>Notes: Read only</p> <p>A collection of Effort objects.</p>
ElementGUID	<p>String</p> <p>Notes: Read only</p> <p>A globally unique ID for this element; that is, unique across all model files.</p>
ElementID	<p>Long</p> <p>Notes: Read only</p> <p>The local ID of the Element; valid for this file only.</p>
Elements	<p>Collection</p> <p>Notes: Read only</p> <p>Returns a collection of child elements (sub-elements) attached to this element as seen in the tree view.</p>
EmbeddedElements	<p>Collection</p> <p>Notes: Read only</p> <p>A list of elements that are embedded into this element, such as Ports, Parts, Pins and Parameter Sets.</p>
EventFlags	<p>String</p> <p>Notes: Read/Write</p> <p>A structure to hold a variety of flags to do with signals or events.</p>
ExtensionPoints	<p>String</p> <p>Notes: Read/Write</p>

	Optional extension points for a Use Case as a comma-separated list.
Files	Collection Notes: Read only A collection of File objects.
FQName	String Notes: Read only The fully-qualified name of the element, consisting of a dot-separated list of names including all parent elements and Packages up to the first namespace root that is encountered.
FQStereotype	String Notes: Read only The fully-qualified stereotype name in the format "Profile::Stereotype". One or more fully-qualified stereotype names can be assigned to StereotypeEx.
GenFile	String Notes: Read/Write The file associated with this element for code generation and synchronization purposes; can include macro expansion tags for local conversion to full path.
Genlinks	String Notes: Read/Write Links to other Classes discovered at code reversing time; Parents and Implements connectors only.
GenType	String Notes: Read/Write The code generation type; for example, Java, C++, C#, VBNet, Visual Basic, Delphi.
Header1	Variant Notes: Read/Write A user defined string for inclusion as header in the source files generated.
Header2	Variant Notes: Read/Write Same as for Header1, but used in the CPP source file.
IsActive	Boolean Notes: Read/Write Boolean value indicating whether the element is active or not. 1 = True, 0 = False.
IsComposite	Boolean Notes: Read/Write Indicates whether the element is composite or not. 1 = True, 0 = False.

IsLeaf	<p>Boolean</p> <p>Notes: Read/Write</p> <p>Indicates whether or not the element is a leaf node (and therefore cannot be a parent for any other elements).</p> <p>1 = True, 0 = False.</p>
IsNew	<p>Boolean</p> <p>Notes: Read/Write</p> <p>Boolean value indicating whether the element is new or not.</p> <p>1 = True, 0 = False.</p>
IsRoot	<p>Boolean</p> <p>Notes: Read/Write</p> <p>Indicates whether or not the element is a root node (and therefore cannot be descended from another element).</p> <p>1 = True, 0 = False.</p>
IsSpec	<p>Boolean</p> <p>Notes: Read/Write</p> <p>Boolean value indicating whether the element is a specification or not.</p> <p>1 = True, 0 = False.</p>
Issues	<p>Collection</p> <p>Notes: Read only</p> <p>Collection of Issue objects.</p>
Locked	<p>Boolean</p> <p>Notes: Read/Write</p> <p>Indicates if the element has been locked against further change.</p>
MetaType	<p>String</p> <p>Notes: Read only</p> <p>The element's domain-specific meta type, as defined by an applied stereotype from an MDG Technology.</p>
Methods	<p>Collection</p> <p>Notes: Read only</p> <p>Collection of Method objects for current element.</p>
MethodsEx	<p>Collection</p> <p>Notes: Read only</p> <p>Collection of Method objects belonging to the current element and its parent elements.</p>
Metrics	<p>Collection</p> <p>Notes: Read only</p> <p>Collection of Metric elements for current element.</p>

MiscData	<p>String</p> <p>Notes: Read only</p> <p>This low-level property provides information about the contents of the PData x fields.</p> <p>These database fields are not documented, and developers must gain understanding of these fields through their own endeavors to use this property.</p> <p>MiscData is zero based, therefore:</p> <ul style="list-style-type: none"> • MiscData(0) corresponds to PData1 • MiscData(1) to PData2, and so on
Modified	<p>Date</p> <p>Notes: Read/Write</p> <p>The date the element was last modified.</p>
Multiplicity	<p>String</p> <p>Notes: Read/Write</p> <p>Multiplicity value for this element.</p>
Name	<p>String</p> <p>Notes: Read/Write</p> <p>The element name; should be unique within the current Package.</p>
Notes	<p>String</p> <p>Notes: Read/Write</p> <p>Further descriptive text about the element.</p>
ObjectType	<p>ObjectType</p> <p>Notes: Read only</p> <p>Distinguishes objects referenced through a Dispatch interface.</p>
PackageID	<p>Long</p> <p>Notes: Read/Write</p> <p>A local ID for the Package containing this element.</p>
ParentID	<p>Long</p> <p>Notes: Read/Write</p> <p>If this element is a child of another, used to set or retrieve the ElementID of the other element; if not, returns 0.</p>
Partitions	<p>Collection</p> <p>Notes: Read only</p> <p>List of logical partitions into which an element can be divided.</p> <p>Only valid for elements that support partitions, such as Activities and States.</p>
Persistence	<p>String</p> <p>Notes: Read/Write</p> <p>The persistence associated with this element; can be Persistent or Transient.</p>

Phase	String Notes: Read/Write The phase this element is scheduled to be constructed in; any string value.
Priority	String Notes: Read/Write The priority of this element as compared to other project elements; only applies to Requirement, Change and Issue types, otherwise ignored. Valid values are: Low, Medium and High.
Properties	Properties Notes: Returns a list of specialized properties that apply to the element that might not be available using the automation model. The properties are purposely undocumented because of their obscure nature and because they are subject to change as progressive enhancements are made to them.
PropertyType	Long Notes: Read/Write The ElementID of a Type associated with this element; only valid for Port and Part elements.
PropertyTypeName	String Notes: Read The name of a Type associated with this element; only valid for Port and Part elements.
Realizes	Collection Notes: Read only List of Interfaces realized by this element for convenience.
Requirements	Collection Notes: Read only Collection of Requirement objects.
RequirementsEx	Collection Notes: Read only Collection of Requirement objects belonging to the current element and its parent elements.
Resources	Collection Notes: Read only Collection of Resource objects for current element.
Risks	Collection Notes: Read only Collection of Risk objects.
RunState	String Notes: Read/Write

	The object's runstate list as a string.
Scenarios	Collection Notes: Read only Collection of Scenario objects for current element.
StateTransitions	Collection Notes: Read only List of State Transitions that an element can support; applies in particular to Timing elements.
Status	String Notes: Read/Write Sets or gets the status, such as Proposed or Approved.
Stereotype	String Notes: Read/Write The primary element stereotype; the first of the list of stereotypes you can access using the 'StereotypeEx' attribute.
StereotypeEx	String Notes: Read/Write All the applied stereotypes of the element in a comma-separated list. Reading the value will provide the stereotype name only; assigning the value accepts either fully-qualified or simple names.
StyleEx	String Notes: Read/Write Advanced style settings; reserved for the use of Sparx Systems.
Subtype	Long Notes: Read/Write A numeric subtype that qualifies the Type of the main element <ul style="list-style-type: none"> • For Event: 0 = Receiver, 1 = Sender • For Class: 1 = Parameterised, 2 = Instantiated, 3 = Both, 0 = Neither, 17 = Association Class <p>If 17, because an Association Class has been created through the user interface, MiscData(3) contains the ID of the related Association; as MiscData is read-only, you cannot create an Association Class through the Automation Interface:</p> <ul style="list-style-type: none"> • For Note: 1 = Note linked to connector, 2 = Constraint linked to connector • For StateNode: 100 = ActivityInitial, 101 = ActivityFinal • For Activity: 0 = Activity, 8 = composite Activity (also set to 8 for other composite elements such as Use Cases) • For Synchronization: 0 = Horizontal, 1 = Vertical <p>Note that there are many more Types than indicated in the above examples.</p>
Tablespace	String

	<p>Notes: Read/Write</p> <p>Associated tablespace for a Table element.</p>
Tag	<p>String</p> <p>Notes: Read/Write</p> <p>Corresponds to the 'Keywords' field in the Enterprise Architect user interface.</p>
TaggedValues	<p>Collection</p> <p>Notes: Read only</p> <p>Returns a collection of TaggedValue objects.</p>
TaggedValuesEx	<p>Collection</p> <p>Notes: Read only</p> <p>Returns a collection of TaggedValue objects belonging to the current element and the elements specialized or realized by the current element.</p>
TemplateParameters	<p>Collection</p> <p>Notes: Read Only</p> <p>A collection of TemplateParameter objects.</p>
Tests	<p>Collection</p> <p>Notes: Read only</p> <p>A collection of Test objects for the current element.</p>
TreePos	<p>Long</p> <p>Notes: Read/Write</p> <p>Sets or gets the tree position.</p>
Type	<p>String</p> <p>Notes: Read/Write</p> <p>The element type (such as Class, Component).</p> <p>Note that Type is case sensitive inside Enterprise Architect and should be provided with an initial capital (proper case); valid types are:</p> <ul style="list-style-type: none"> • Action • Activity • ActivityPartition • ActivityRegion • Actor • Artifact • Association • Boundary • Change • Class • Collaboration • Component • Constraint • Decision

	<ul style="list-style-type: none"> • DeploymentSpecification • DiagramFrame • EmbeddedElement • Entity • EntryPoint • Event • ExceptionHandler • ExitPoint • ExpansionNode • ExpansionRegion • Feature • GUIElement • InteractionFragment • InteractionOccurrence • InteractionState • Interface • InterruptibleActivityRegion • Issue • Node • Note • Object • Package • Parameter • Part • Port • ProvidedInterface • Report • RequiredInterface • Requirement • Screen • Sequence • State • StateNode • Synchronization • Text • TimeLine • UMLDiagram • UseCase
Version	String Notes: Read/Write The version of the element.
Visibility	String Notes: Read/Write

	<p>The Scope of this element within the current Package.</p> <p>Valid values are: Public, Private, Protected or Package.</p>
--	--

Element Methods

Method	Remarks
ApplyGroupLock(string aGroupName)	<p>Boolean</p> <p>Notes: Applies a group lock to the element object, for the specified group, on behalf of the current user.</p> <p>Returns True if the operation is successful; returns False if the operation is unsuccessful. Use 'GetLastError()' to retrieve error information.</p> <p>Parameters:</p> <ul style="list-style-type: none"> aGroupName: String - the name of the user group for which to set the group lock
ApplyUserLock()	<p>Boolean</p> <p>Notes: Applies a user lock to the element object for the current user.</p> <p>Returns True if the operation is successful; returns False if the operation is unsuccessful. Use 'GetLastError()' to retrieve error information.</p>
Clone ()	<p>LDISPATCH</p> <p>Notes: Inserts a copy of the selected element under the same parent as the selected element.</p> <p>Returns the newly-created element.</p>
CreateAssociationClass(long ConnectorID)	<p>Boolean</p> <p>Notes: Makes this element an AssociationClass of the Association with the provided Connector ID; the return value indicates whether the function succeeded in converting the element to an AssociationClass.</p> <p>AssociationClasses are created only where:</p> <ul style="list-style-type: none"> The current element is valid The current element is a Class The current element is not already an AssociationClass The specified connector exists The specified connector is an Association The specified connector is not already in an AssociationClass pair The current element is not at either end of the specified connector <p>Parameters:</p> <ul style="list-style-type: none"> ConnectorID: Long - the Connector ID of an Association connector
DeleteLinkedDocument()	<p>Boolean</p> <p>Notes: Removes the Linked Document for the element. This method does not display a confirmatory prompt.</p> <p>Returns True if a document was deleted.</p>

GetBusinessRules()	String Notes: Read Only. Returns all the Business Rules for the element.
GetDecisionTable()	String Notes: Provides read-only access to a decision table XML string. Returns the XML data for the decion table as a string.
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
GetLinkedDocument()	String Notes: Returns a string value containing the element's linked document contents, in Rich Text Format. If the element contains no linked document, an empty string is returned.
GetRelationSet(EnumRelationSetType Type)	String Notes: Returns a string containing a comma-separated list of ElementIDs of directly- and indirectly -related elements based on the given type. Recurses using the same relation type on all elements it finds, retrieving all dependencies and sub-dependencies of the current element; for example, Object1 depends on Object2, which depends on Object3, therefore this method returns Object2 and Object3. To obtain only the direct relationships of the element, use the Connector collection instead.
GetStereotypeList()	String Notes: Returns a comma-separated list of stereotypes allied to this element.
HasStereotype(string Stereotype)	Boolean Notes: Returns true if the current element has the specified stereotype applied to it. Accepts either qualified or unqualified stereotype names; for example, 'block' or 'SysML1.3::block'. Parameters: <ul style="list-style-type: none"> • Stereotype: String - the name of the stereotype to search for
IsAssociationClass	Boolean Notes: Returns whether or not the current element is an AssociationClass.
LoadLinkedDocument(string Filename)	Boolean Notes: Loads the document from the specified file into the element's linked document. Parameters: <ul style="list-style-type: none"> • FileName: String - the name of the file from which to load the document; both RTF and DOCX input formats are supported
Refresh()	Void Notes: Refreshes the element features in the Project Browser. Usually called after adding or deleting attributes or methods, when the user

	interface is required to be updated as well.
ReleaseUserLock()	<p>Boolean</p> <p>Notes: Releases a user lock or group lock on the element object.</p> <p>Returns True if the operation is successful; returns False if the operation is unsuccessful. Use GetLastError() to retrieve error information.</p>
SaveLinkedDocument(string Filename)	<p>Boolean</p> <p>Notes: Saves the linked document for this element to the specified file. Returns False if the element does not have a Linked document or fails to save the file.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • FileName: String - the name of the file to save to disk The output format will be determined by the file's extension - currently rtf, docx and pdf are supported; if an invalid extension is used, it will write the file in RTF format regardless of the extension
SetAppearance(long Scope, long Item, long Value)	<p>Void</p> <p>Notes: Sets the visual appearance of the element.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Scope: Long - Scope of appearance set to modify 1 - Base (Default appearance across entire model) To set appearance for the element (diagram object) in a selected diagram only, see <i>Setting The Style</i> in the <i>DiagramObject Class</i> topic • Item: Long - Appearance feature to modify 0 - Background color 1 - Font Color 2 - Border Color 3 - Border Width • Value: Long - Value to set appearance to
SetCompositeDiagram()	<p>Boolean</p> <p>Notes: Sets the composite diagram of the element.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • GUID: String - the GUID of the composite diagram; a blank GUID will remove the link to the composite diagram
SetCreated(Date NewVal)	<p>Void</p> <p>Notes: Deprecated</p> <p>This method is no longer supported.</p>
SetModified(Date NewVal)	<p>Void</p> <p>Notes: Deprecated</p> <p>This method is no longer supported.</p>
SynchConstraints(string Profile, string Stereotype)	<p>Boolean</p> <p>Notes: Synchronizes the constraints of a UML Profile item for this element, only if the specified stereotype has been applied.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Profile: String - Name of the profile that contains the stereotype • Stereotype: String - Name of the profile stereotype for which the default

	constraints are to be synchronized
SynchTaggedValues(string Profile, string Stereotype)	<p>Boolean</p> <p>Notes: Synchronizes the Tagged Values of a UML Profile item for this element, only if the specified stereotype has been applied.</p> <p>Parameters:</p> <ul style="list-style-type: none">• Profile: String - Name of the profile that contains the stereotype• Stereotype: String - Name of the profile stereotype for which the default tags are to be synchronized
UnlinkFromAssociation	<p>Boolean</p> <p>Notes: Performs the opposite of CreateAssociationClass().</p>
Update()	<p>Boolean</p> <p>Notes: Updates the current element object after modification or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>

File Class

A File represents an associated file for an element. Files are accessed through the Element Files collection.

Associated table in .EAP file

t_objectfiles

File Attributes

Attribute	Remarks
FileDate	String Notes: Read/Write The file date when the entry was created.
Name	String Notes: Read/Write The file name can be a logical file or a reference to a web address (using http://).
Notes	String Notes: Read/Write Notes about the file.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Size	String Notes: Read/Write The file size.
Type	String Notes: Read/Write The file type.

File Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in

	relation to this object.
Update()	<p>Boolean</p> <p>Notes: Updates the current File object after modification or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>

Issue (Maintenance) Class

An Issue is either a Change or a Defect, is associated with the containing element, and is accessed through the Issues collection of an element.

Associated table in .EAP file

t_objectproblems

Issue Attributes

Attribute	Remarks
DateReported	Date Notes: Read/Write The date the issue was reported.
DateResolved	Date Notes: Read/Write The date the issue was resolved.
ElementID	Long Notes: Read/Write The ID of the element associated with this issue.
Name	String Notes: Read/Write The Issue name; that is, the Issue itself.
Notes	String Notes: Read/Write The Issue description.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Priority	String Notes: Read/Write The priority of the Issue - Low, Medium or High.
Reporter	String Notes: Read/Write The user ID of the person reporting the issue.

Resolver	String Notes: Read/Write The user ID of the person resolving the issue.
ResolverNotes	String Notes: Read/Write Notes entered by the resolver about resolution of the Issue.
Severity	String Notes: Read/Write The Issue severity - Low, Medium or High.
Status	String Notes: Read/Write The current status of the issue.
Type	Variant Notes: Read/Write The Issue type - Defect, Change, Issue or ToDo.
Version	String Notes: Read/Write The version associated with the issue. Note that this method is only available through a Dispatch interface. Object ob = Issue; Print ob.Version;

Issue Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Updates the current Issue object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Metric Class

A Metric is a named item with a weighting that can be associated with an element for purposes of building metrics about the model. Metrics are accessed through the Element Metrics collection.

Associated table in .EAP file

t_objectmetrics

Metric Attributes

Attribute	Remarks
Name	String Notes: Read/Write The name of the metric.
Notes	String Notes: Read/Write Notes about this metric.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Type	String Notes: Read/Write The metric type.
Weight	Long Notes: Read/Write A user-defined weighting for estimation or metric purposes.

Metric Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Updates the current Metric object after modification or appending a new

	<p>item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>
--	---

Requirement Class

An Element Requirement object holds information about the responsibilities of an element in the context of the model. Requirements can be accessed using the Element Requirements collection.

Associated table in .EAP file

t_objectrequires

Requirement Attributes

Attribute	Remarks
Difficulty	String Notes: Read/Write The estimated difficulty of implementing the requirement.
LastUpdate	Date Notes: Read/Write The date the requirement was last updated.
Name	String Notes: Read/Write The requirement itself.
Notes	String Notes: Read/Write Further notes on the requirement.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
ParentID	Long Notes: Read only The ElementID of the element to which this requirement applies.
Priority	String Notes: Read/Write The assigned priority of the requirement.
RequirementID	Long Notes: Read only A local ID for this requirement.

Stability	String Notes: Read/Write The estimated stability of the requirement. This is an indication of the probability of the requirement - or understanding of the requirement - changing. High stability indicates a low probability of the requirement changing.
Status	String Notes: Read/Write The current status of the requirement.
Type	String Notes: Read/Write The requirement type.

Requirement Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Updates the current Requirement object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Resource Class

An Element Resource is a named person/task pair with timing constraints and percent complete indicators. Use this to manage the work associated with delivering an Element.

Associated table in .EAP file

t_objectresources

Resource Attributes

Attribute	Description
ActualHours	Long Notes: Read/Write The time already expended on the task, in hours, days or other units.
DateEnd	Date Notes: Read/Write The expected end date.
DateStart	Date Notes: Read/Write The date to start work.
ExpectedHours	Long Notes: Read/Write The total expected time the task might run, in hours, days or other units.
History	String Notes: Read/Write Gets or sets history text.
Name	String Notes: Read/Write The name of the resource (for example, a person's name).
Notes	String Notes: Read/Write Descriptive notes.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

PercentComplete	Long Notes: Read/Write The current percent complete figure.
Role	String Notes: Read/Write The role the resource plays in implementing the element.
Time	Long Notes: Read/Write The time expected to complete the task; a numeric indicating the number of days.

Resource Methods

Method	Description
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.
Update()	Boolean Notes: Update the current Resource object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Risk Class

A Risk object represents a named risk associated with an element, it is used for project management purposes. Risks can be accessed through the Element Risks collection.

Associated table in .EAP file

t_objectrisks

Risk Attributes

Attribute	Description
Name	String Notes: Read/Write The name of the risk.
Notes	String Notes: Read/Write Further notes describing the risk.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Type	String Notes: Read/Write The risk type associated with this element.
Weight	Long Notes: Read/Write A weighting for estimation or metric purposes.

Risk Methods

Method	Description
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Update the current Risk object after modification or appending a new item.

	If False is returned, check the 'GetLastError()' function for more information.
--	---

Scenario Class

A Scenario corresponds to a Collaboration or Use Case instance. Each Scenario is a path of execution through the logic of a Use Case. Scenarios can be added to using the Element Scenarios collection.

Associated table in .EAP file

t_objectscenarios

Scenario Attributes

Attribute	Description
Name	String Notes: Read/Write The Scenario name.
Notes	String Notes: Read/Write A description of the Scenario, usually containing the steps to execute the scenario.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
ScenarioGUID	String Notes: Read/Write A unique ID for the Scenario, used to identify the Scenario unambiguously within a model.
Steps	Collection of ScenarioStep Class Notes: Read only A collection of step objects for this Scenario. Use the 'AddNew' and 'Delete' functions to manage steps. 'AddNew' passes the step name and 'I' as the type for an actor step.
Type	String Notes: Read/Write The scenario type (for example, Basic Path).
Weight	Long Notes: Read/Write Currently used to position scenarios in the scenario list (that is, List Position).
XMLContent	String

	<p>Notes: Read/Write</p> <p>A structured field that can contain scenario details in XML format. It is recommended that you use the 'Steps' collection to read or modify this field.</p>
--	---

Scenario Methods

Method	Description
GetLastError()	<p>String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p>
Update()	<p>Boolean</p> <p>Notes: Update the current Scenario object after modification or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>

ScenarioExtension Class

ScenarioExtension Attributes

Attribute	Description
ExtensionGUID	String Notes: Read/Write A unique GUID for this Extension.
Join	String Notes: Read/Write The GUID of the step where this Extension rejoins the Scenario.
JoiningStep	ScenarioStep Notes: Read only The actual step where this Extension rejoins the Scenario, if any.
Level	String Notes: Read only The number of this Extension as shown in the scenario editor. This is derived from the value of Pos for this object and the owning step.
Name	String Notes: Read/Write The Extension name. This should match the name of the linked scenario.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Pos	Long Notes: Read/Write The position of the Extension in the Extensions list.
Scenario	Scenario Notes: Read only The scenario that is executed as an alternative path for this Extension.

ScenarioExtension Methods

Method	Description

GetLastError()	<p>String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p>
Update()	<p>Boolean</p> <p>Notes: Updates the current ScenarioExtension object after modification or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>

ScenarioStep Class

ScenarioStep Attributes

Attribute	Description
Extensions	Collection of ScenarioExtension Notes: Read only A collection of ScenarioExtension objects that specify how the scenario is extended from this step. The arguments to 'AddNew' should match the name and GUID of the alternative scenario being linked to.
Level	String Notes: Read only The number of this Step as shown in the scenario editor. This is derived from the value of Pos.
Link	String Notes: Read/Write The GUID of a Use Case that is relevant to this step.
LinkedElement	Element Notes: Read only The actual element specified by Link, if any.
Name	String Notes: Read/Write The step name.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Pos	Long Notes: Read/Write The position of the 'Step' in the 'Scenario Step' list.
Results	String Notes: Read/Write Any results that are given from this step.
State	String Notes: Read/Write A description of the state the system enters when this Step is executed.
StepGUID	String

	Notes: Read/Write A unique GUID for this Step.
StepType	ScenarioStepType Notes: Read/Write Identifies whether this step is being performed by a user or the system.
Uses	String Notes: Read/Write The input and requirements that are relevant to this step.
UsesElementList	Collection of Element Notes: Read only Indicates that the 'Structured Specification' tab 'Uses' field is a linked element list.

ScenarioStep Methods

Method	Description
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Updates the current ScenarioStep object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

TaggedValue Class

A TaggedValue is a named property and value associated with an element. Tagged Values can be accessed through the TaggedValues collection.

Associated table in .EAP file

t_objectproperties

TaggedValue Attributes

Attribute	Description
ElementID	Long Notes: Read/Write The local ID of the associated element.
FQName	String Notes: Read only The fully-qualified name of the tag.
Name	String Notes: Read/Write The name of the tag.
Notes	String Notes: Read/Write Further descriptive notes about this tag. If 'Value' is set to '<memo>', then 'Notes' should contain the actual Tagged Value content.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
PropertyGUID	String Notes: Read/Write The global ID of the tag.
PropertyID	Long Notes: Read only The local ID of the tag.
Value	String Notes: Read/Write

	<p>The value assigned to this tag.</p> <p>This field has a 255 character limit. If the value is greater than 255 characters long, set the value to "<memo>" and insert the body of text in the 'Notes' attribute.</p> <p>When reading existing Tagged Values, if 'Value' = "<memo>" then the developer should read the actual body of text from the 'Notes' attribute.</p>
--	--

TaggedValue Methods

Method	Description
GetAttribute(string propName)	<p>String</p> <p>Notes: Returns the text of a single named property within a structured Tagged Value.</p> <p>Parameters:</p> <ul style="list-style-type: none">propName: String - the name of the property for which the text is being returned
GetLastError()	<p>String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p>
HasAttributes()	<p>Boolean</p> <p>Notes: Returns True if the Tagged Value is a structured Tagged Value with one or more properties.</p>
SetAttribute(string propName, string propValue)	<p>Boolean</p> <p>Notes: Sets the text of a single named property within a structured Tagged Value.</p> <p>Parameters:</p> <ul style="list-style-type: none">propName: String - the name of the property for which the text is being setpropValue: the value of the property
Update()	<p>Boolean</p> <p>Notes: Updates the current TaggedValue object after modification or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>

Test Class

A Test is a single Test Case applied to an element. Tests are added and accessed through the Element Tests collection.

Associated table in .EAP file

t_objecttests

Test Attributes

Attribute	Description
AcceptanceCriteria	String Notes: Read/Write The acceptance criteria for successful execution.
CheckedBy	String Notes: Read/Write User ID of the person confirming the results.
Class	Long Notes: Read/Write The test Class: 1 = Unit Test 2 = Integration Test 3 = System Test 4 = Acceptance Test 5 = Scenario Test 6 = Inspection Test
DateRun	Date Notes: Read/Write The date the test was last run.
Input	String Notes: Read/Write Input data for the test.
Name	String Notes: Read/Write The test name.
Notes	String Notes: Read/Write

	Detailed notes about test to be carried out.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
RunBy	String Notes: Read/Write The user ID of the person conducting the test.
Status	String Notes: Read/Write The current status of the test.
TestResults	Variant Notes: Read/Write Results of test.
Type	String Notes: Read/Write The test type, such as Load or Regression.

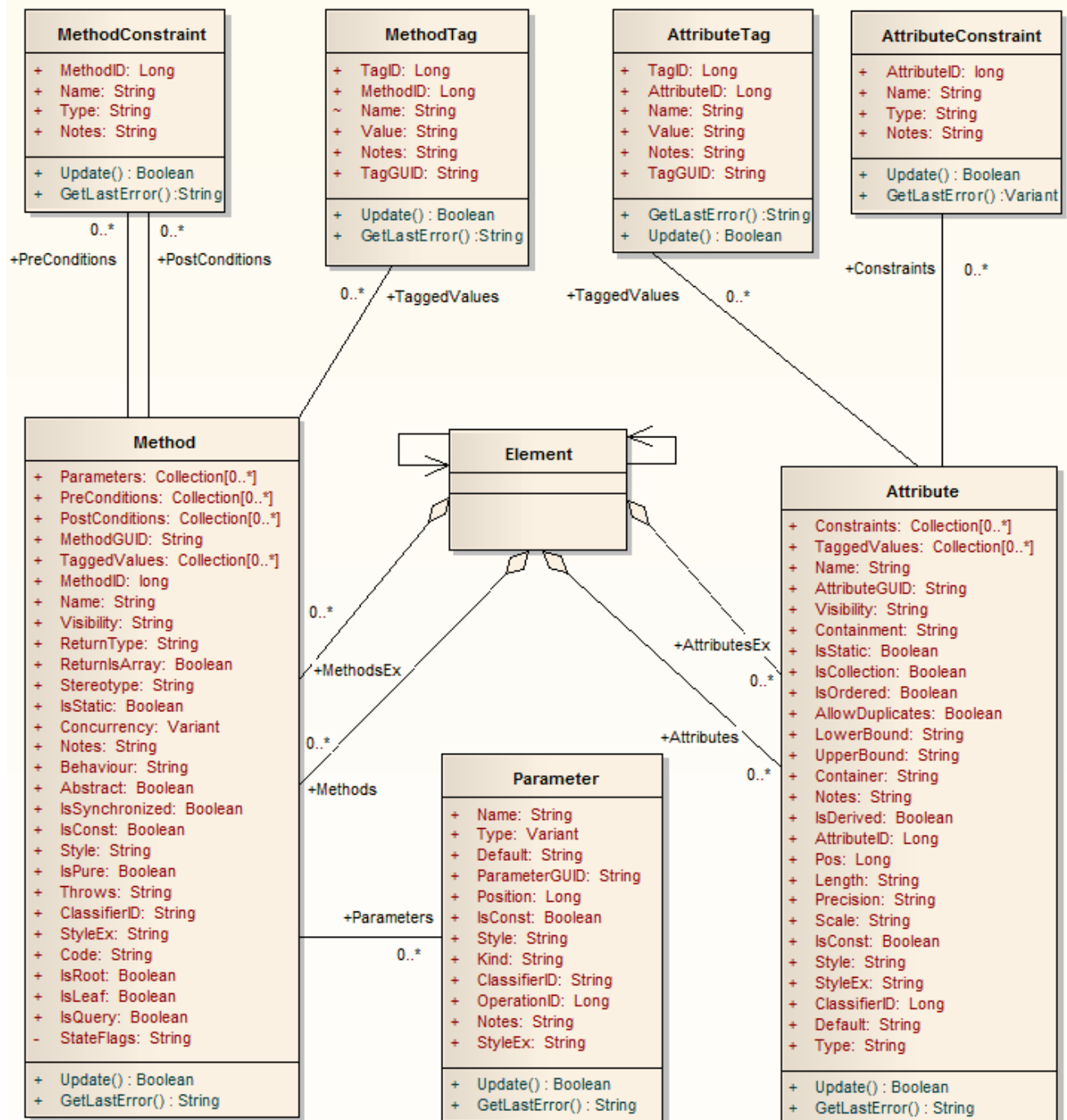
Test Methods

Method	Description
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Update the current Test object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Element Features Package

The ElementFeatures Package contains descriptions of the model interfaces that enable access to operations and attributes, and their associated Tagged Values and constraints.

This diagram illustrates the components associated with element features. These include Attributes and Methods, and the associated constraints and Tagged Values related to them. It also includes the Parameter object that defines the arguments associated with an operation (method).



Attribute Class

An attribute corresponds to a UML Attribute. It contains further collections for constraints and Tagged Values. Attributes are accessed from the element Attributes collection.

Associated table in .EAP file

t_attribute

Attribute Attributes

Attribute	Remarks
Alias	String Notes: Read/Write Contains the (optional) 'Alias' property for this attribute. This can be used interchangeably with the Style attribute.
AllowDuplicates	Boolean Notes: Read/Write Indicates if duplicates are allowed in the collection. If the attribute represents a database column this, when set, represents the 'Not Null' option.
AttributeGUID	String Notes: Read only A globally unique ID for the current attribute. This attribute is system generated.
AttributeID	Long Notes: Read only The local ID number of the attribute.
ClassifierID	Long Notes: Read/Write The classifier ID, if appropriate, indicating the base type associated with the attribute, if not a primitive type.
Container	String Notes: Read/Write The container type.
Containment	String Notes: Read/Write The type of containment - Not Specified, By Reference or By Value.
Constraints	Collection

	<p>Notes: Read only</p> <p>A collection of AttributeConstraint objects, used to access and manage constraints associated with this attribute.</p>
Default	<p>String</p> <p>Notes: Read/Write</p> <p>The initial value assigned to this attribute.</p>
FQStereotype	<p>String</p> <p>Notes: Read Only</p> <p>The fully-qualified stereotype name in the format "Profile::Stereotype". One or more fully-qualified stereotype names can be assigned to StereotypeEx.</p>
IsCollection	<p>Boolean</p> <p>Notes: Read/Write</p> <p>Indicates if the current feature is a collection or not. If the attribute represents a database column this, when set, represents a Foreign Key.</p>
IsConst	<p>Boolean</p> <p>Notes: Read/Write</p> <p>A flag indicating if the attribute is Const or not.</p>
IsDerived	<p>Boolean</p> <p>Notes: Read/Write</p> <p>Indicates if the attribute is derived (that is, a calculated value).</p>
IsID	<p>Boolean</p> <p>Notes: Read/Write</p> <p>Indicates if the attribute uniquely identifies an instance of the containing Class, or not.</p>
IsOrdered	<p>Boolean</p> <p>Notes: Read/Write</p> <p>Indicates if a collection is ordered or not. If the attribute represents a database column this, when set, represents a Primary Key.</p>
IsStatic	<p>Boolean</p> <p>Notes: Read/Write</p> <p>Indicates if the current attribute is a static feature or not. If the attribute represents a database column this, when set, represents the 'Unique' option.</p>
Length	<p>String</p> <p>Notes: Read/Write</p> <p>The attribute length, where applicable.</p>
LowerBound	<p>String</p> <p>Notes: Read/Write</p> <p>A value for the collection lower boundary.</p>

Name	String Notes: Read/Write The attribute name.
Notes	String Notes: Read/Write Further notes on this attribute.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
ParentID	Long Notes: Read only Returns the ElementID of the element that this attribute is a part of.
Pos	Long Notes: Read/Write The position of the attribute in the Class attribute list.
Precision	String Notes: Read/Write The precision value.
RedefinedProperty	String Notes: Read/Write Corresponds to the 'Redefined Property' field on the 'Detail' page of the attribute 'Properties' dialog, or the UML <i>redefinedProperty</i> attribute. Contains a comma separated list of GUIDs.
Scale	String Notes: Read/Write The scale value.
Stereotype	String Notes: Read/Write Sets or gets the stereotype for this attribute.
StereotypeEx	String Notes: Read/Write Provides all the applied stereotypes of the attribute, in a comma-separated list. Reading the value will provide the stereotype name only; assigning the value accepts either fully-qualified or simple names.
Style	String Notes: Read/Write Contains the (optional) Alias property for this attribute. This can be used interchangeably with the Alias attribute.

StyleEx	String Notes: Read/Write Advanced style settings, reserved for the use of Sparx Systems.
SubsettedProperty	String Notes: Read/Write Corresponds to the 'Subsetted Property' field on the 'Detail' page of the attribute 'Properties' dialog, or the UML <i>subsettedProperty</i> attribute. Contains a comma separated list of GUIDs.
TaggedValues	Collection of type AttributeTag Notes: Read only A collection of AttributeTag objects, used to access and manage Tagged Values associated with this attribute.
TaggedValuesEx	Collection of type TaggedValue Notes: Read only A collection of TaggedValue objects belonging to the current attribute and the TaggedValuesEx property of its classifier.
Type	String Notes: Read/Write The attribute type (by name; also see <i>ClassifierID</i>).
UpperBound	String Notes: Read/Write A value for the collection upper boundary.
Visibility	String Notes: Read/Write Identifies the scope of the attribute - Private, Protected, Public or Package.

Attribute Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Updates the current attribute object after modifying or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

AttributeConstraint Class

An AttributeConstraint is a constraint associated with the current Attribute.

Associated table in .EAP file

t_attributeconstraints

AttributeConstraint Attributes

Attribute	Remarks
AttributeID	Long Notes: Read/Write The ID of the attribute this constraint applies to.
Name	String Notes: Read/Write The name of the constraint.
Notes	String Notes: Read/Write Descriptive notes about the constraint.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Type	String Notes: Read/Write The type of constraint.

AttributeConstraint Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Update the current AttributeConstraint object after modification or appending a new item.

	If False is returned, check the 'GetLastError()' function for more information.
--	---

AttributeTag Class

An AttributeTag represents a Tagged Value associated with an attribute.

Associated table in .EAP file:

t_attributetag

AttributeTag Attributes:

Attribute	Remarks
AttributeID	Long Notes: Read/Write The local ID of the attribute associated with this Tagged Value.
FQName	String Notes: Read only The fully-qualified name of the tag.
Name	String Notes: Read/Write The name of the tag.
Notes	String Notes: Read/Write Further descriptive notes about this tag. If 'Value' is set to '<memo>', then 'Notes' should contain the actual Tagged Value content.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
TagGUID	String Notes: Read/Write A globally unique ID for this Tagged Value.
TagID	Long Notes: Read only The local ID to identify the Tagged Value.
Value	String Notes: Read/Write The value assigned to this tag.

	<p>This field has a 255 character limit. If the value is greater than 255 characters long, set the value to "<memo>" and insert the body of text in the 'Notes' attribute.</p> <p>When reading existing Tagged Values, if 'Value' = "<memo>" then the developer should read the actual body of text from the 'Notes' attribute.</p>
--	---

AttributeTag Methods:

Method	Remarks
GetAttribute(string propName)	<p>String</p> <p>Notes: Returns the text of a single named property within a structured Tagged Value.</p>
GetLastError()	<p>String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p> <p>This function is rarely used as an exception is thrown when an error occurs.</p>
HasAttributes()	<p>Boolean</p> <p>Notes: Returns True if the Tagged Value is a structured Tagged Value with one or more properties.</p>
SetAttribute(string propName, string propValue)	<p>Boolean</p> <p>Notes: Sets the text of a single named property within a structured Tagged Value.</p>
Update()	<p>Boolean</p> <p>Notes: Updates the current AttributeTag object after modification or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>

CustomProperties Collection

The CustomProperties collection contains 0 or more CustomProperties associated with the current element. These properties provide advanced UML configuration options, and must not be added to or deleted. The value of each property can be set.

CustomProperty

Attribute	Remarks
Name	String Notes: Read only The CustomProperty name.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Value	String Notes: Read/Write The value associated with this CustomProperty. This can be: <ul style="list-style-type: none">• A string• The boolean values True or False, or• An enumeration value from a defined list The UML 2.5 specification in general provides information on the kinds of enumeration relevant here.

Notes

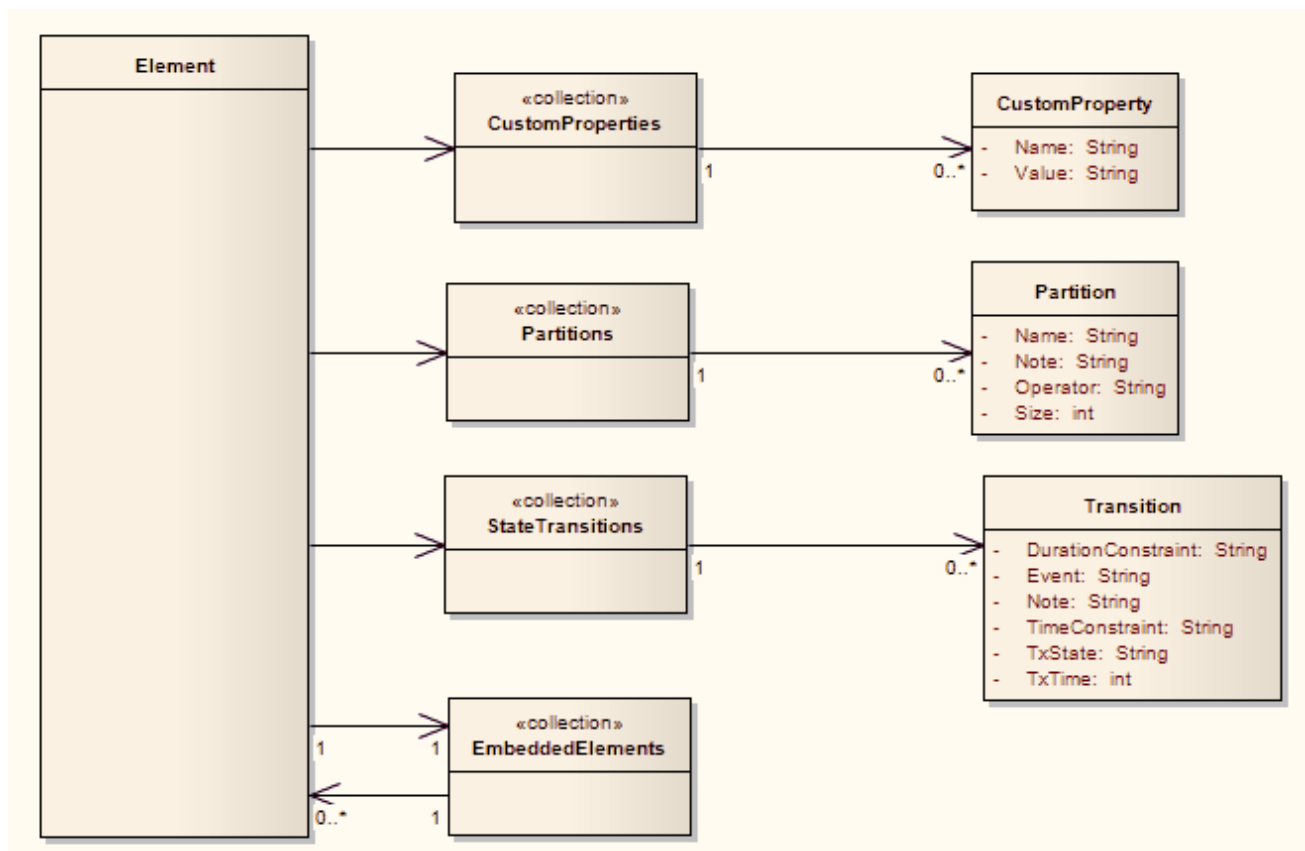
- The number and type of properties vary depending on the actual element

EmbeddedElements Collection

In UML 2.5 an element can have one or more embedded elements such as Ports, Pins, Parameters or ObjectNodes. These are attached to the boundary of the host element and cannot be moved off the element. They are owned by their host element. This collection gives easy access to the set of elements embedded on the surface of an element. Note that some embedded elements can have their own embedded element collection (for example, Ports can have Interfaces embedded on them).

The EmbeddedElements collection contains Element objects.

Example



Method Class

A method represents a UML operation. It is accessed from the Element Methods collection and includes collections for parameters, constraints and Tagged Values.

Associated table in .EAP file

t_operation

Method Attributes

Attribute	Remarks
Abstract	Boolean Notes: Read/Write A flag indicating if the method is abstract (1) or not (0).
Behavior	String Notes: Read/Write Some further explanatory behavior notes (for example, pseudocode). In earlier releases of Enterprise Architect this attribute had the UK/Australian spelling 'Behaviour'; this is still present for backwards compatibility, but please now use the 'Behavior' attribute for consistency.
ClassifierID	String Notes: Read/Write The Classifier ID that applies to the ReturnType.
Code	String Notes: Read/Write An optional field to hold the method code (used for the 'Initial Code' field).
Concurrency	Variant Notes: Read/Write Indicates the concurrency type of the method.
FQStereotype	String Notes: Read Only The fully-qualified stereotype name in the format "Profile::Stereotype". One or more fully-qualified stereotype names can be assigned to StereotypeEx.
IsConst	Boolean Notes: Read/Write A flag indicating that the method is Const.
IsLeaf	Boolean

	<p>Notes: Read/Write</p> <p>A flag to indicate if the method is a Leaf (cannot be overridden).</p>
IsPure	<p>Boolean</p> <p>Notes: Read/Write</p> <p>A flag indicating that the method is defined as 'Pure' in C++.</p>
IsQuery	<p>Boolean</p> <p>Notes: Read/Write</p> <p>A flag to indicate if the method is a query (that is, does not alter Class variables).</p>
IsRoot	<p>Boolean</p> <p>Notes: Read/Write</p> <p>A flag to indicate if the method is Root.</p>
IsStatic	<p>Boolean</p> <p>Notes: Read/Write</p> <p>A flag to indicate a static method.</p>
IsSynchronized	<p>Boolean</p> <p>Notes: Read/Write</p> <p>A flag indicating a Synchronized method call.</p>
MethodGUID	<p>String</p> <p>Notes: Read/Write</p> <p>A globally unique ID for the current method. This is system generated.</p>
MethodID	<p>Long</p> <p>Notes: Read only</p> <p>A local ID for the current method, only valid within this .eap file.</p>
Name	<p>String</p> <p>Notes: Read/Write</p> <p>The method name.</p>
Notes	<p>String</p> <p>Notes: Read/Write</p> <p>Descriptive notes on the method.</p>
ObjectType	<p>ObjectType</p> <p>Notes: Read only</p> <p>Distinguishes objects referenced through a Dispatch interface.</p>
Parameters	<p>Collection Class</p> <p>Notes: Read only</p> <p>The Parameters collection for the current method, used to add and access parameter objects for the current method.</p>

ParentID	<p>Long</p> <p>Notes: Read only</p> <p>Returns the ElementID of the element that this method belongs to.</p>
Pos	<p>Long</p> <p>Notes: Read/Write</p> <p>Specifies the position of the method within the set of operations defined for a Class.</p>
PostConditions	<p>Collection Class</p> <p>Notes: Read only</p> <p>The PostConditions (constraints) as they apply to this method. This returns a MethodConstraint object of type 'post'.</p>
PreConditions	<p>Collection Class</p> <p>Notes: Read only</p> <p>The PreConditions (constraints) as they apply to this method. This returns a MethodConstraint object of type 'pre'.</p>
ReturnIsArray	<p>Boolean</p> <p>Notes: Read/Write</p> <p>A flag to indicate that the return value is an array.</p>
ReturnType	<p>String</p> <p>Notes: Read/Write</p> <p>The return type for the method; this can be a primitive data type or a Class or Interface type.</p>
StateFlags	<p>String</p> <p>Notes: Read/Write</p> <p>Some flags as applied to methods in State elements.</p>
Stereotype	<p>String</p> <p>Notes: Read/Write</p> <p>The method stereotype (optional).</p>
StereotypeEx	<p>String</p> <p>Notes: Read/Write</p> <p>All the applied stereotypes of the method in a comma-separated list. Reading the value will provide the stereotype name only; assigning the value accepts either fully-qualified or simple names.</p>
Style	<p>String</p> <p>Notes: Read/Write</p> <p>Contains the Alias property for this method.</p>
StyleEx	<p>String</p> <p>Notes: Read/Write</p> <p>Advanced style settings, reserved for the use of Sparx Systems.</p>

TaggedValues	Collection Class of type MethodTag Class Notes: Read only The TaggedValues collection for the current method. This accesses a list of MethodTag objects.
Throws	String Notes: Read/Write Exception information. Valid input for setting the Throws is: <ul style="list-style-type: none">• GUID String - the GUID of an element in the model or a comma-separated list of element GUIDS• <none> - removes the existing Throws set
Visibility	String Notes: Read/Write The method scope - Public, Protected, Private or Package.

Method Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Update the current method object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

MethodConstraint Class

A MethodConstraint is a condition imposed on a method. It is accessed through either the Method PreConditions or Method PostConditions collection.

Associated table in .EAP file

t_operationpres and t_operationposts

MethodConstraint Attributes

Attribute	Remarks
MethodID	Long Notes: Read/Write The local ID of the associated method.
Name	String Notes: Read/Write The name of the constraint.
Notes	String Notes: Read/Write Descriptive notes about this constraint.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Type	String Notes: Read/Write The constraint type.

MethodConstraint Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.
Update()	Boolean

	<p>Notes: Update the current MethodConstraint object after modification or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>
--	---

MethodTag Class

A MethodTag is a Tagged Value associated with a method.

Associated table in .EAP file:

t_operationtag

MethodTag Attributes:

Attribute	Remarks
FQName	String Notes: Read only The fully-qualified name of the tag.
MethodID	Long Notes: Read/Write The ID of the associated method.
Name	String Notes: Read/Write The tag or name of the property.
Notes	String Notes: Read/Write Further descriptive notes about this tag. If 'Value' is set to '<memo>', then 'Notes' should contain the actual Tagged Value content.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
TagGUID	String Notes: Read/Write A unique GUID for this Tagged Value.
TagID	Long Notes: Read only A unique ID for this Tagged Value.
Value	String Notes: Read/Write The value assigned to this tag.

	<p>This field has a 255 character limit. If the value is greater than 255 characters long, set the value to "<memo>" and insert the body of text in the 'Notes' attribute.</p> <p>When reading existing Tagged Values, if 'Value' = "<memo>" then the developer should read the actual body of text from the 'Notes' attribute.</p>
--	---

MethodTag Methods:

Method	Remarks
GetAttribute(string propName)	<p>String</p> <p>Notes: Returns the text of a single named property within a structured Tagged Value.</p>
GetLastError()	<p>String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p> <p>This function is rarely used as an exception is thrown when an error occurs.</p>
HasAttributes()	<p>Boolean</p> <p>Notes: Returns True if the Tagged Value is a structured Tagged Value with one or more properties.</p>
SetAttribute(string propName, string propValue)	<p>Boolean</p> <p>Notes: Sets the text of a single named property within a structured Tagged Value.</p>
Update()	<p>Boolean</p> <p>Notes: Updates the current MethodTag object after modification or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>

Parameter Class

A Parameter object represents a method argument and is accessed through the Method Parameters collection.

Associated table in .EAP file

t_operationparams

Parameter Attributes

Attribute	Remarks
Alias	String Notes: Read/Write An optional alias for this parameter.
ClassifierID	String Notes: Read/Write A ClassifierID for the parameter, if known.
Default	String Notes: Read/Write A default value for this parameter.
IsConst	Boolean Notes: Read/Write A flag indicating that the parameter is Const (cannot be altered).
Kind	String Notes: Read/Write The parameter kind - in, inout, out, or return.
Name	String Notes: Read/Write The parameter name; this must be unique for a single method.
Notes	String Notes: Read/Write Descriptive notes.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
OperationID	Long

	Notes: Read only The ID of the method associated with this parameter.
ParameterGUID	String Notes: Read/Write A system generated, globally unique ID for the current Parameter.
Position	Long Notes: Read/Write The position of the parameter in the argument list.
Stereotype	String Notes: Read/Write The first stereotype of the parameter.
StereotypeEx	String Notes: Read/Write All the applied stereotypes of the parameter in a comma-separated list. Reading the value will provide the stereotype name only; assigning the value accepts either fully-qualified or simple names.
Style	String Notes: Read/Write Some style information.
StyleEx	String Notes: Read/Write Advanced style settings, reserved for the use of Sparx Systems.
TaggedValues	Collection Class of type ParamTag Class Notes: Read/Write The GUID of the parameter with which this ParamTag is associated.
Type	Variant Notes: Read/Write The parameter type; can be a primitive type or a defined classifier.

Parameter Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean

	<p>Notes: Update the current Parameter object after modifying or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>
--	---

ParamTag Class

A ParamTag is a Tagged Value associated with a method parameter.

Associated table in .EAP file

t_taggedvalue

ParamTag Attributes

Attribute	Remarks
ElementGUID	String Notes: Read/Write The GUID of the parameter with which this ParamTag is associated.
FQName	String Notes: Read only The fully qualified name of the tag.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
PropertyGUID	String Notes: Read/Write A system generated GUID to identify the Tagged Value.
Tag	String Notes: Read/Write The actual tag name.
Value	String Notes: Read/Write The value associated with this tag.

ParamTag Methods

Method	Remarks
GetAttribute(string propName)	String Notes: Returns the text of a single named property within a structured Tagged

	Value.
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
HasAttributes()	Boolean Notes: Returns True if the Tagged Value is a structured Tagged Value with one or more properties.
SetAttribute(string propName, string propValue)	Boolean Notes: Sets the text of a single named property within a structured Tagged Value.
Update()	Boolean Notes: Updates the current ParamTag object after modifying or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Partitions Collection

A collection of internal element partitions (regions). This is commonly seen in Activity, State, Boundary, Diagram Frame and similar elements. Not all elements support partitions.

This collection contains a set of Partition elements. The set is read/write: information is not saved until the host element is saved, so ensure that you call the `Element.Save` method after making changes to a Partition.

Partition Attributes

Attribute	Remarks
Name	String Notes: Read/Write The partition name; this can represent a condition or constraint in some cases.
Note	String Notes: Read/Write A free text note associated with this partition.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Operator	String Notes: Read/Write An optional operator value that specifies the partition type.
Size	String Notes: Read/Write The vertical or horizontal width of the partition in pixels.

Properties Class

Properties

Properties Attributes

Attribute	Remarks
Count	Long Notes: The number of properties that are available for this object.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

Properties Methods

Property

Method	Remarks
Item(object Index)	Property Notes: Returns a property either by name or by a zero-based integer offset into the list of properties. Parameter: <ul style="list-style-type: none">Index: Variant - either a string representing the property name or an integer representing the zero-based offset into the property list

Property Attributes

Attribute	Remarks
Name	String Notes: Read only The name of the property. The object to which the properties list applies can have an automation property with the same name, in which case the data accessed through Value is identical to that obtained through the automation property.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

Type	PropType Notes: Read only Provides an indication of what sort of data is going to be stored by this property. This restriction can be further defined by the Validation attribute.
Validation	String Notes: Read only An optional string that is used to validate any data that is passed to the Value attribute. This string is used by the programmer at run time to provide an indication of what is expected, and by Enterprise Architect to ensure that the submitted data is appropriate.
Value	Variant Notes: Read/write The value of the property as defined in the other fields.

TemplateParameter Class

A TemplateParameter for a template signature specifies a formal parameter that will be substituted by an actual parameter (or the default) in a TemplateBinding relationship on a Class element.

Associated table in .EAP file

t_xref

TemplateParameter Attributes

Attribute	Remarks
Constraint	String Notes: Read/Write The name of the Classifier that acts as the constraint value.
Default	String Notes: Read/Write The name of the Classifier that acts as the default value.
Name	String Notes: Read/Write The name of the Template Parameter.
ObjectType	ObjectType Notes: Read Only Distinguishes objects referenced through a Dispatch interface.
TemplateParameterID	String Notes: Read Only The Enterprise Architect Globally Unique ID (GUID) of the current Template Parameter, in the XrefID column of t_xref.
Type	String Notes: Read/Write The Template Parameter type.

TemplateParameter Methods

Method	Remarks
GetLastError()	String

	Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	<p>Boolean</p> <p>Notes: Updates the current TemplateParameter object after modifying or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>

Transitions Collection

The Transitions collection applies only to Timeline elements.

A Timeline element displays 0 or more state transitions at set times on its extent. This collection enables you to access the transition set. You can also access additional information by referring to the connectors associated with the Timeline, and by referencing messages passed between timelines. Note that any changes made to elements in this collection are only saved when the main element is saved.

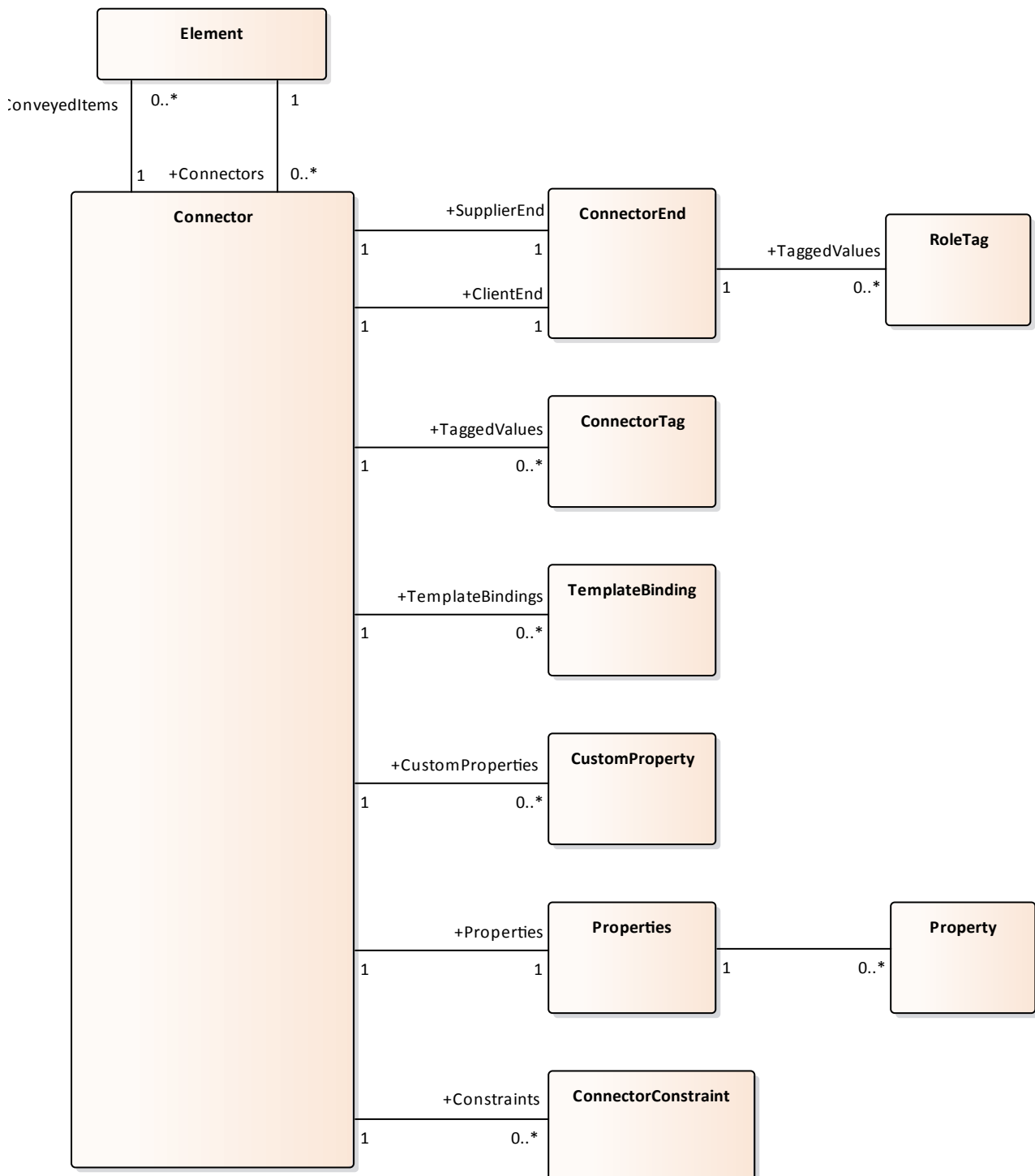
Transition Attributes

Attribute	Remarks
DurationConstraint	String Notes: Read/Write A constraint on the time duration of the transition.
Event	String Notes: Read/Write The event (optional) that initiated the transition.
Note	String Notes: Read/Write A free text note.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
TimeConstraint	String Notes: Read/Write A constraint on when the transition has to be completed.
TxState	String Notes: Read/Write The state to transition to, as defined in the 'Timeline Properties' dialog.
TxTime	String Notes: Read/Write. The time that the transition occurs. The value depends on a range set in the diagram.

Connector Package

The Connector Package details how connectors between elements are accessed and managed.

This diagram shows the Connector Class, its collections and its relationships to the Element Class. Association Target roles correspond to member variable names in the source interface. The associated Classes represent the object type used in each collection.



Connector Class

To represent the various kinds of connectors between UML elements, you use a Connector object. You can access this from either the Client or Supplier element, using the Connectors collection of that element. When creating a new connector you assign to it a valid type from this list:

- Aggregation
- Assembly
- Association
- Collaboration
- CommunicationPath
- Connector
- ControlFlow
- Delegate
- Dependency
- Deployment
- ERLink
- Generalization
- InformationFlow
- Instantiation
- InterruptFlow
- Manifest
- Nesting
- NoteLink
- ObjectFlow
- Package
- Realization
- Sequence
- StateFlow
- TemplateBinding
- UseCase

Associated table in .EAP file

t_connector

Connector Attributes

Attribute	Remarks
Alias	String Notes: Read/Write An optional alias for this connector.

AssociationClass	<p>Element</p> <p>Notes: Read Only</p> <p>Returns the Association Class element if the connector has one; otherwise NULL/.</p>
ClientEnd	<p>ConnectorEnd</p> <p>Notes: Read Only</p> <p>A pointer to the ConnectorEnd object representing the source end of the relationship.</p>
ClientID	<p>Long</p> <p>Notes: Read/Write</p> <p>The ElementID of the element at the source end of this connector.</p>
Color	<p>Long</p> <p>Notes: Read/Write</p> <p>Sets the color of the connector.</p>
ConnectorGUID	<p>String</p> <p>Notes: Read Only</p> <p>A system generated, globally unique ID for the current connector.</p>
ConnectorID	<p>Long</p> <p>Notes: Read Only</p> <p>A system generated local identifier for the current connector.</p>
Constraints	<p>Collection</p> <p>Notes: Read Only</p> <p>A collection of constraint objects.</p>
ConveyedItems	<p>Collection of type Element</p> <p>Notes: Read Only</p> <p>Returns a collection of elements that have been conveyed.</p> <p>To add another element to the conveyed Collection, use 'AddNew (ElementGUID,NULL)', where 'ElementGUID' is the GUID of the element to be added.</p>
CustomProperties	<p>Collection</p> <p>Notes: Read Only</p> <p>Returns a collection of advanced properties associated with an element in the form of CustomProperty objects.</p>
DiagramID	<p>Long</p> <p>Notes: Read/Write</p> <p>The DiagramID of the connector.</p>
Direction	<p>String</p> <p>Notes: Read/Write</p>

	<p>The connector direction, which can be set to one of:</p> <ul style="list-style-type: none"> • Unspecified • Bi-Directional • Source -> Destination or • Destination -> Source <p>If the connector is non-navigable, set the 'sourceNavigability' and/or 'targetNavigability' attributes.</p>
EndPointX	<p>Long</p> <p>Notes: Read/Write</p> <p>The x-coordinate of the connector's end point.</p> <p>Connector end points are specified in Cartesian coordinates with the origin to the top left of the screen.</p>
EndPointY	<p>Long</p> <p>Notes: Read/Write</p> <p>The y-coordinate of the connector's end point.</p> <p>Connector end points are specified in Cartesian coordinates with the origin to the top left of the screen.</p>
EventFlags	<p>String</p> <p>Notes: Read/Write</p> <p>A structure to hold a variety of flags concerned with event signaling on messages.</p>
FQStereotype	<p>String</p> <p>Notes: Read Only</p> <p>The fully-qualified stereotype name in the format "Profile::Stereotype". One or more fully-qualified stereotype names can be assigned to StereotypeEx.</p>
ForeignKeyInformation	<p>String</p> <p>Notes: Read Only</p> <p>Returns the Foreign Key information.</p>
IsLeaf	<p>Boolean</p> <p>Notes: Read/Write</p> <p>A flag indicating that the connector is a leaf.</p>
IsRoot	<p>Boolean</p> <p>Notes: Read/Write</p> <p>A flag indicating that the connector is a root.</p>
IsSpec	<p>Boolean</p> <p>Notes: Read/Write</p> <p>A flag indicating that the connector is a specification.</p>
MessageArguments	<p>String</p> <p>Notes: Read Only</p> <p>The connector Message arguments.</p>

MetaType	String Notes: Read Only The connector's domain-specific meta type, as defined by an applied stereotype from an MDG Technology.
MiscData	String Notes: Read Only This low-level property returns an array providing information about the contents of the PData x fields. These database fields are not documented and developers must gain understanding of these fields through their own endeavors to use this property. MiscData is zero based, therefore: <ul style="list-style-type: none"> • MiscData(0) corresponds to PData1 • MiscData(1) corresponds to PData2, and so on
Name	String Notes: Read/Write The connector name.
Notes	String Notes: Read/Write Descriptive notes about the connector.
ObjectType	ObjectType Notes: Read Only Distinguishes objects referenced through a Dispatch interface.
Properties	Properties Notes: Returns a list of specialized properties applicable to the connector that might not be available using the automation model. The properties are purposely undocumented because of their obscure nature and because they are subject to change as progressive enhancements are made to them.
ReturnValueAlias	String Notes: Shows the 'Return Value Alias' field of the operation.
RouteStyle	Long Notes: Read/Write The route style.
SequenceNo	Long Notes: Read/Write The SequenceNo of the connector.
StartPointX	Long Notes: Read/Write The x-coordinate of the connector's start point. Connector end points are specified in Cartesian coordinates with the origin to the

	top left of the screen.
StartPointY	<p>Long</p> <p>Notes: Read/Write</p> <p>The y-coordinate of the connector's start point.</p> <p>Connector end points are specified in Cartesian coordinates with the origin to the top left of the screen.</p>
StateFlags	<p>String</p> <p>Notes: Read/Write</p> <p>A structure to hold a variety of flags concerned with State signaling on messages; the list is delimited by semi-colons.</p>
Stereotype	<p>String</p> <p>Notes: Read/Write</p> <p>Sets or gets the stereotype for this connector end.</p>
StereotypeEx	<p>String</p> <p>Notes: Read/Write</p> <p>All the applied stereotypes of the connector in a comma-separated list. Reading the value will provide the stereotype name only; assigning the value accepts either fully-qualified or simple names.</p>
StyleEx	<p>String</p> <p>Notes: Read/Write</p> <p>Advanced style settings; reserved for the use of Sparx Systems.</p>
Subtype	<p>String</p> <p>Notes: Read/Write</p> <p>A possible subtype to refine the meaning of the connector.</p>
SupplierEnd	<p>ConnectorEnd</p> <p>Notes: Read Only</p> <p>A pointer to the ConnectorEnd object representing the target end of the relationship.</p>
SupplierID	<p>Long</p> <p>Notes: Read/Write</p> <p>The ElementID of the element at the target end of this connector.</p>
TaggedValues	<p>Collection of type ConnectorTag</p> <p>Notes: Read Only</p> <p>The collection of ConnectorTag objects.</p>
TemplateBindings	<p>Collection of type TemplateBinding</p> <p>Notes: Read Only</p> <p>A collection of TemplateBinding objects.</p>
TransitionAction	String

	Notes: Read/Write See the <i>Transition</i> topic for appropriate values.
TransitionEvent	String Notes: Read/Write See the <i>Transition</i> topic for appropriate values.
TransitionGuard	String Notes: Read/Write See the <i>Transition</i> topic for appropriate values.
Type	String Notes: Read/Write The connector type; valid types are held in the t_connectortypes table in the .eap file.
VirtualInheritance	String Notes: Read/Write For Generalization, indicates if the inheritance is virtual.
Width	Long Notes: Read/Write Specifies the width of the connector.

Connector Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
IsConnectorValid()	Boolean Notes: Queries Enterprise Architect's internal relationship validation schema on the current connector. If False is returned, check the 'GetLastError()' function for more information.
Update()	Boolean Notes: Updates the current ConnectorObject after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

ConnectorConstraint Class

A ConnectorConstraint holds information about special conditions that apply to a connector. It is accessed through the Connector Constraints collection.

Associated table in .EAP file

t_connectorconstraints

ConnectorConstraint Attributes

Attribute	Remarks
ConnectorID	Long Notes: Read/Write A local ID value (long) - system generated.
Name	String Notes: Read/Write The constraint name.
Notes	String Notes: Read/Write Notes about this constraint.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Type	String Notes: Read/Write The constraint type.

ConnectorConstraint Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Update the current ConnectorConstraint object after modification or

	appending a new item. If False is returned, check the 'GetLastError()' function for more information.
--	--

ConnectorEnd Class

A ConnectorEnd contains information about a single end of a connector. A ConnectorEnd is accessed from the connector as either the ClientEnd or SupplierEnd.

Associated table in .EAP file

derived from t_connector

ConnectorEnd Attributes

Attribute	Remarks
Aggregation	Long Notes: Read/Write The type of Aggregation as it applies to this end; valid values are: 0 = None 1 = Shared 2 = Composite
Alias	String Notes: Read/Write An optional alias for this connector end.
AllowDuplicates	Boolean Notes: Read/Write For multiplicities greater than 1, indicates that duplicate entries are possible.
Cardinality	String Notes: Read/Write The cardinality associated with this end.
Constraint	String Notes: Read/Write A constraint that can be applied to this connector end.
Containment	String Notes: Read/Write The containment type applied to this connector end.
Derived	Boolean Notes: Read/Write Indicates that the value of this end is derived.
DerivedUnion	Boolean

	<p>Notes: Read/Write</p> <p>Indicates the value of this role derived from the union of all roles that subset this.</p>
End	<p>String</p> <p>Notes: Read only</p> <p>The end this ConnectorEnd object applies to - Client or Supplier.</p>
IsChangeable	<p>String</p> <p>Notes: Read/Write</p> <p>Flag indicating whether this end is changeable or not - 'frozen', 'addOnly' or none.</p>
IsNavigable	<p>Note: This property is not used</p> <p>Boolean</p> <p>Notes: Read/Write</p> <p>A flag indicating this end is navigable from the other end.</p>
Navigable	<p>String</p> <p>Notes: Read/Write</p> <p>Indicates whether this role of an association is navigable from the opposite classifier - Navigable, Non-Navigable or Unspecified.</p>
ObjectType	<p>ObjectType</p> <p>Notes: Read only</p> <p>Distinguishes objects referenced through a Dispatch interface.</p>
Ordering	<p>Long</p> <p>Notes: Read/Write</p> <p>Ordering for this connector end.</p>
OwnedByClassifier	<p>Boolean</p> <p>Notes: Read/Write</p> <p>Indicates that this Association end corresponds to an attribute on the opposite end of the Association.</p>
Qualifier	<p>String</p> <p>Notes: Read/Write</p> <p>A qualifier that can apply to the connector end.</p>
Role	<p>String</p> <p>Notes: Read/Write</p> <p>The connector end role.</p>
RoleNote	<p>String</p> <p>Notes: Read/Write</p> <p>Notes associated with the role of this connector end.</p>
RoleType	<p>String</p> <p>Notes: Read/Write</p>

	The role type applied to this end of the connector.
Stereotype	String Notes: Read/Write Sets or gets the stereotype for this connector end.
StereotypeEx	String Notes: Read/Write All the applied stereotypes of the connector end in a comma-separated list. Reading the value will provide the stereotype name only; assigning the value accepts either fully qualified or simple names.
TaggedValues	Collection of type RoleTag Notes: Read only A collection of RoleTag objects.
Visibility	String Notes: Read/Write The Scope associated with this connector end - Public, Private, Protected or Package.

ConnectorEnd Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Update the current ConnectorEnd object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

ConnectorTag Class

A ConnectorTag is a Tagged Value for a connector and is accessed through the Connector TaggedValues collection.

Associated table in .EAP file

t_connectortag

ConnectorTag Attributes

Attribute	Remarks
ConnectorID	Long Notes: Read/Write The local ID of the associated connector.
FQName	String Notes: Read only The fully qualified name of the tag.
Name	String Notes: Read/Write The tag or name.
Notes	String Notes: Read/Write Further descriptive notes on this tag. If 'Value' is set to '<memo>', then 'Notes' should contain the actual Tagged Value content.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
TagGUID	String Notes: Read/Write A globally unique ID for this Tagged Value.
TagID	Long Notes: Read only A local ID to identify the Tagged Value.
Value	String Notes: Read/Write The value assigned to this tag.

	<p>This field has a 255 character limit. If the value is greater than 255 characters long, set the value to "<memo>" and insert the body of text in the 'Notes' attribute.</p> <p>When reading existing Tagged Values, if 'Value' = "<memo>" then the developer should read the actual body of text from the 'Notes' attribute.</p>
--	---

ConnectorTag Methods

Method	Remarks
GetAttribute(string propName)	String Notes: Returns the text of a single named property within a Structured Tagged Value.
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
HasAttributes()	Boolean Notes: Returns True if the Tagged Value is a Structured Tagged Value with one or more properties.
SetAttribute(string propName, string propValue)	Boolean Notes: Sets the text of a single named property within a Structured Tagged Value.
Update()	Boolean Notes: Update the current ConnectorTag object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

RoleTag Class

The RoleTag interface provides access to an Association's Role Tagged Values. Each connector end has a RoleTag collection that can be accessed to add, delete and access the RoleTags.

You might use this in creating code that resembles this fragment for accessing a RoleTag in VB.NET (where con is a Connector Object):

```
client = con.ClientEnd
client.Role = "m_client"
client.Update()
tag = client.TaggedValues.AddNew("tag", "value")
tag.Update()
tag = client.TaggedValues.AddNew("tag2", "value2")
tag.Update()
client.TaggedValues.Refresh()
For idx = 0 To client.TaggedValues.Count - 1
tag = client.TaggedValues.GetAt(idx)
Console.WriteLine(tag.Tag)
client.TaggedValues.DeleteAt(idx, False)
Next
tag = Nothing
```

Associated table in .EAP file

t_taggedvalue

RoleTag Attributes

Attribute	Description
BaseClass	String Notes: Read/Write Indicates the role end; set to ASSOCIATION_SOURCE or ASSOCIATION_TARGET.
ElementGUID	String Notes: Read/Write The GUID of the connector with which this role tag is associated.
FQName	String Notes: Read only The fully qualified name of the tag.
ObjectType	ObjectType

	Notes: Read only Distinguishes objects referenced through a Dispatch interface.
PropertyGUID	String Notes: Read/Write A system generated GUID to identify the Tagged Value.
Tag	String Notes: Read/Write The actual tag name.
Value	String Notes: Read/Write The value associated with this tag.

RoleTag Methods

Method	Description
GetAttribute(string propName)	String Notes: Returns the text of a single named property within a Structured Tagged Value.
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
HasAttributes()	Boolean Notes: Returns True if the Tagged Value is a Structured Tagged Value with one or more properties.
SetAttribute(string propName, string propValue)	Boolean Notes: Sets the text of a single named property within a Structured Tagged Value.
Update()	Boolean Notes: Update the RoleTag after changes or on initial creation. If False is returned, check the 'GetLastError()' function for more information.

TemplateBinding Class

A TemplateBinding defines the connector between a binding Class and a parameterized Class, and the binding expression on that connector.

TemplateBinding Attributes

Attribute	Remarks
ActualGUID	String Notes: Read/Write The GUID of the element classifier set as the Actual Template Binding parameter. If the Actual Template Binding parameter is set as a string expression only, this will be an empty string. Assigning a GUID value will automatically change the ActualName attribute after Update() has been called.
ActualName	String Notes: Read/Write The name of the Actual Template Binding parameter. Assigning a new value will clear any current ActualGUID value.
BindingExpression	String Notes: Read only The Binding Expression as shown in Enterprise Architect.
ConnectorGUID	String Notes: Read only The Globally Unique ID of the associated connector.
ConnectorType	String Notes: Read only The type of the associated connector.
FormalName	String Notes: Read/Write The name of the Formal Template Binding parameter.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch Interface.
Pos	String Notes: Read only The position of the Template Binding in the list (as on the 'Bindings' page of the connector 'Properties' dialog).

TemplateBindingID	String Notes: Read only The Globally Unique ID of the current Template Binding.
-------------------	---

TemplateBinding Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Update the current TemplateBinding object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Diagram Package

The Diagram Package has information on a diagram and on DiagramObject and DiagramLink, which are the instances of elements within a diagram.

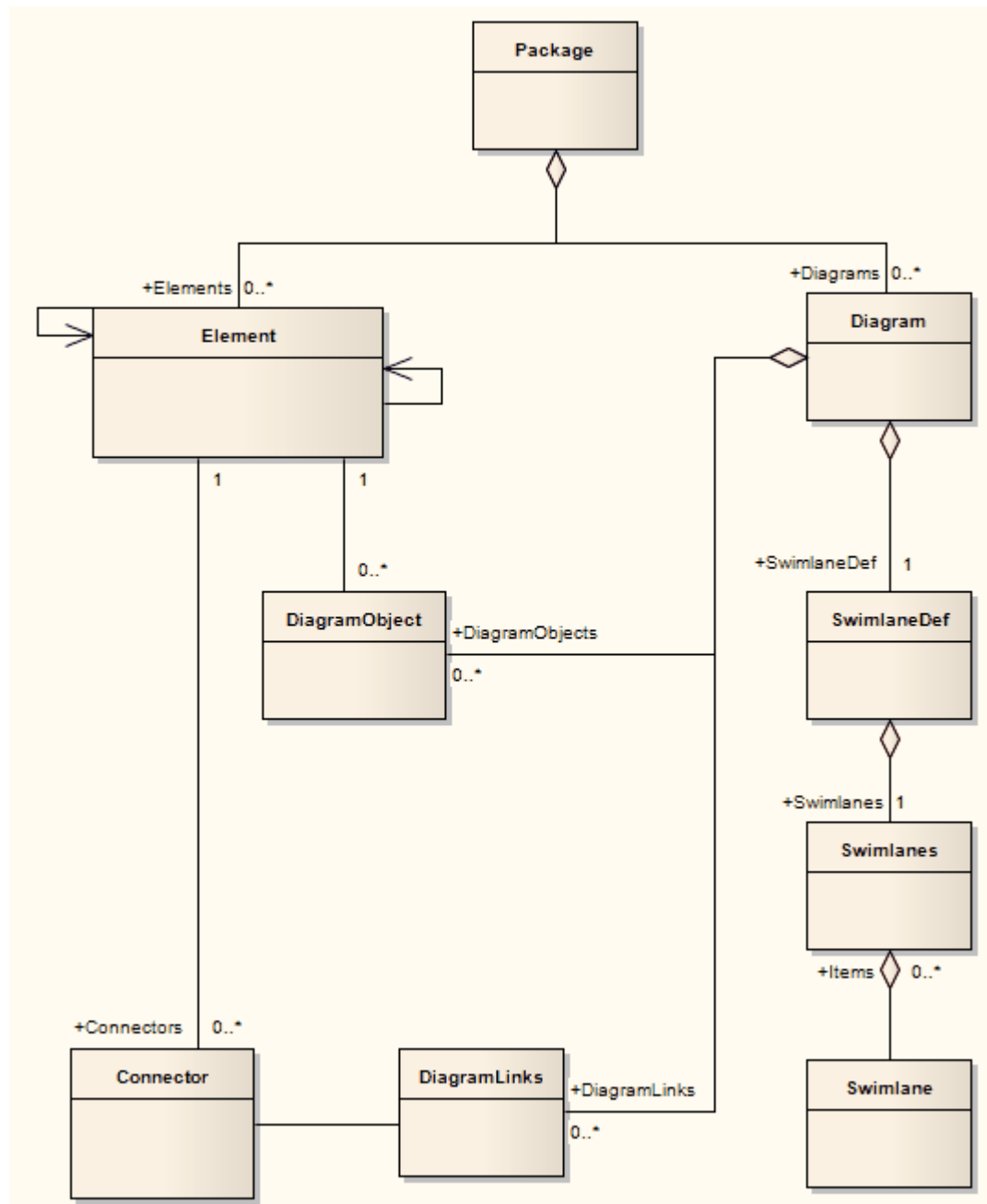


Diagram Class

A Diagram corresponds to a single UML diagram. It is accessed through the Package Diagrams collection and in turn contains a collection of diagram objects and diagram connectors. Adding to the DiagramObject Class adds an existing element to the diagram. When adding a new diagram, you must set the diagram type to one of the valid types:

- Activity
- Analysis
- Component
- Custom
- Deployment
- Logical
- Sequence
- Statechart
- Use Case

For a Collaboration (Communication) diagram, use the Analysis type.

Associated table in .EAP file

t_diagram

Diagram Attributes

Attribute	Remarks
Author	String Notes: Read/Write The name of the author.
CreatedDate	Date Notes: Read/Write The date the diagram was created.
cx	Long Notes: Read/Write The X dimension of the diagram (the default is 800).
cy	Long Notes: Read/Write The Y dimension of the diagram (the default is 1100).
DiagramGUID	Variant Notes: Read/Write A globally unique ID for this diagram.

DiagramID	Long Notes: Read only A local ID for the diagram.
DiagramLinks	Collection Notes: Read only A list of DiagramLink objects, each containing information about the display characteristics of a connector in a diagram.
DiagramObjects	Collection Notes: Read only A collection of references to DiagramObjects. A DiagramObject is an instance of an element in a diagram, and includes size and display characteristics.
ExtendedStyle	String Notes: Read/Write An extended style attribute.
FilterElements	String Notes: Read/Write Applies a comma-separated list of object ids (from SelectedObjects) to the currently-applied diagram filter, overriding the filter. The effect persists until another filter is applied, or the diagram is closed.
HighlightImports	Boolean Notes: Read/Write A flag to indicate that elements from other Packages should be highlighted. Corresponds with the 'Show Namespace' option in the diagram 'Properties' dialog.
IsLocked	Boolean Notes: Read/Write A flag indicating whether this diagram is locked or not.
MetaType	String Notes: Read/Write The diagram's domain-specific meta type, as defined by an MDG Technology. When writing, the meta type must be fully qualified and from an existing profile.
ModifiedDate	Variant Notes: Read/Write The date the diagram was last modified.
Name	String Notes: Read/Write The diagram name.
Notes	String Notes: Read/Write Set or retrieve notes for this diagram.

ObjectType	<p>ObjectType</p> <p>Notes: Read only</p> <p>Distinguishes objects referenced through a Dispatch interface.</p>
Orientation	<p>String</p> <p>Notes: Read/Write</p> <p>The page orientation: P for Portrait or L for Landscape.</p>
PackageID	<p>Long</p> <p>Notes: Read/Write</p> <p>The ID of the Package that this diagram belongs to.</p>
PageHeight	<p>Long</p> <p>Notes: Read</p> <p>The number of pages high the diagram is.</p>
PageWidth	<p>Long</p> <p>Notes: Read</p> <p>The number of pages wide the diagram is.</p>
ParentID	<p>Long</p> <p>Notes: Read/Write</p> <p>The optional ID of an element that 'owns' this diagram; for example, a Sequence diagram owned by a Use Case.</p>
Scale	<p>Long</p> <p>Notes: Read/Write</p> <p>The zoom scale (the default is 100).</p>
SelectedConnector	<p>Connector</p> <p>Notes: Read/Write</p> <p>The currently selected connector on this diagram. Null if there is no currently selected diagram.</p>
SelectedObjects	<p>Collection</p> <p>Notes: Read only</p> <p>Gets a collection representing the currently selected elements on the diagram.</p> <p>You can remove objects from this collection to deselect them, and add elements to the collection by passing the Object ID as a name to select them.</p>
ShowDetails	<p>Long</p> <p>Notes: Read/Write</p> <p>A flag to indicate that the Diagram Details text should be shown: 1 = Show, 0 = Hide.</p>
ShowPackageContents	<p>Boolean</p> <p>Notes: Read/Write</p> <p>A flag to indicate that the Package contents should be shown in the current</p>

	diagram.
ShowPrivate	Boolean Notes: Read/Write A flag to show or hide Private features.
ShowProtected	Boolean Notes: Read/Write A flag to show or hide Protected features.
ShowPublic	Boolean Notes: Read/Write A flag to show or hide Public features.
Stereotype	String Notes: Read/Write Sets or gets the stereotype for this diagram.
StyleEx	String Notes: Read/Write Advanced style settings, reserved for the use of Sparx Systems.
Swimlanes	String Notes: Read/Write Information on swimlanes contained in the diagram. Please note that this property is superseded by SwimlaneDef.
SwimlaneDef	SwimlaneDef Notes: Read/Write Information on swimlanes contained in the diagram.
Type	String Notes: Read only The diagram type; see the t_diagramtypes table in the .eap file for more information.
Version	String Notes: Read/Write The version of the diagram.

Diagram Methods

Method	Details
ApplyGroupLock (string	Boolean

aGroupName)	<p>Notes: Applies a group lock to this diagram object, for the specified group, on behalf of the current user.</p> <p>Returns True if the operation is successful; returns False if the operation is unsuccessful. Use GetLastError() to retrieve error information.</p> <p>Parameter:</p> <ul style="list-style-type: none"> aGroupName: String - the name of the user group for which to set the group lock
ApplyUserLock ()	<p>Boolean</p> <p>Notes: Applies a user lock to this diagram object, for the current user.</p> <p>Returns True if the operation is successful; returns False if the operation is unsuccessful. Use GetLastError() to retrieve error information.</p>
FindElementInDiagram (long ElementID)	<p>Boolean</p> <p>Notes: This function activates the Diagram View and displays the diagram with the diagram object selected. If the diagram is too large to display all of it on the screen, the portion of the diagram containing the object is displayed with the object shown in the center of the screen. Diagram objects flagged as non-selected are shown but are not selected</p> <p>Returns True if the diagram object was found, the diagram displayed and the object selected (or at least displayed) in the view. Returns False if the diagram object was not found in the diagram and the diagram not displayed.</p> <p>Parameter</p> <ul style="list-style-type: none"> ElementID: Long - the element ID of the diagram object to locate
GetDiagramObjectByID (long ID, string DUID)	<p>DiagramObject</p> <p>Notes: Returns the DiagramObject object, if it exists on the diagram.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ID: Long - the ElementID of the diagram object DUID: String - the optional Diagram Unique ID of the diagram object
GetLastError ()	<p>String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p>
ReadStyle (string StyleName)	<p>String</p> <p>Notes: Returns the current value of the named diagram style.</p> <p>Use GetLastError() to retrieve error information.</p> <p>Parameters:</p> <ul style="list-style-type: none"> StyleName: String - the name of the diagram style whose value is to be retrieved; valid StyleNames are: <ul style="list-style-type: none"> Show Element Property String Show Connector Property String Show Feature Property String
ReleaseUserLock ()	<p>Boolean</p> <p>Notes: Releases a group lock or user lock on this diagram object.</p> <p>Returns True if the operation is successful; returns False if the operation is unsuccessful. Use GetLastError() to retrieve error information.</p>

ReorderMessages ()	<p>Void</p> <p>Notes: Resets the display order of Sequence and Collaboration messages. This is typically used after inserting or deleting messages in the diagram.</p>
SaveAsPDF (string FileName)	<p>Boolean</p> <p>Notes: Export the diagram to a PDF document. Returns True on success.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • FileName: String - full path to file location
SaveImagePage(long x, long y, long sizeX, long sizeY, string filename, long flags)	<p>Boolean</p> <p>Notes: Saves a page of the diagram to disk.</p> <p>Returns True if the operation is successful; returns False if the operation is unsuccessful.</p> <p>Use GetLastError() to retrieve error information.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • x: Long - the horizontal page • y: Long - the vertical page • sizeX: Long - currently unused; pass a value of 0 to ensure behavior does not change in a future build • sizeY: Long - currently unused; pass a value of 0 to ensure behavior does not change in a future build • filename: String - the filename and path to save the image • flags: Long - additional options, currently unused; pass a value of 0 to ensure behavior does not change in a future build <p>The image type is determined by the extension of the filename. Currently only .emf, .bmp and .png formats are supported.</p>
ShowAsElementList (bool ShowAsList, bool Persist)	<p>Boolean</p> <p>Notes: Toggles the diagram display between diagram format and Diagram List depending on the value of ShowAsList.</p> <p>If Persist is set, the display format is written to the database so the diagram always opens in that format (diagram or list). Otherwise, the display format falls back to the default (diagram) once the display is closed.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ShowAsList: Boolean - indicates diagram or Diagram List • Persist: Boolean - indicates set (maintain ShowAsList value) or not (revert to default)
Update ()	<p>Boolean</p> <p>Notes: Updates this diagram object after modification or appending a new item. If False is returned, use GetLastError() to retrieve error information.</p>
VirtualizeConnector (int ConnectorID, int Action, int X, int Y)	<p>Boolean</p> <p>Notes: Creates a virtual copy of the source or target element on a connector, and sets its location on the diagram as a waypoint on the connector. If the source element is being virtualized, the waypoint is created as the first on the connector, and if the target element is being virtualized, the waypoint is created as the last on the connector.</p> <p>If called again on the same connector, removes the virtual element. However, the</p>

	<p>waypoint remains in place.</p> <p>As waypoints and therefore virtual elements can only be created on connectors with the Custom line style, if the connector does not have this line style the method sets it. So, after this method executes, an Update function should be called for the connector as well as for the diagram. All parameters are required for the function to complete successfully.</p> <p>Returns True if the operation is successful; returns False if the operation is unsuccessful.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ConnectorID - Integer: the ID of the connector on which to create the virtual element Action - Integer: the element to be virtualized; 1 for the source element, 2 for the target element X - Integer: the position on the X axis that the element's center point will be aligned with Y - Integer: the position on the Y axis that the element's centre point will be aligned with <p>For example, to virtualize the source element of the selected connector:</p> <pre>function main() { var diagram as EA.Diagram; var conn as EA.Connector; diagram = Repository.GetCurrentDiagram(); if(diagram != null) { var connector as EA.Connector; connector = diagram.SelectedConnector; diagram.VirtualizeConnector(connector.ConnectorID, 1, 100, 150); connector.Update(); diagram.Update(); Repository.ReloadDiagram(diagram.DiagramID); } else { Session.Output("Script requires a diagram to be visible"); } } main();</pre>
WriteStyle (string StyleName, string StyleValue)	<p>Void</p> <p>Notes: Sets the value of the named diagram style.</p> <p>Use GetLastError() to retrieve error information.</p> <p>Parameters:</p> <ul style="list-style-type: none"> StyleName: String - the name of the diagram style whose value is to be retrieved; valid StyleNames are: <ul style="list-style-type: none"> Show Element Property String Show Connector Property String

	<ul style="list-style-type: none">- Show Feature Property String• StyleValue: String - the value to be set in the named diagram style; valid values for the StyleNames listed above are 0 and 1
--	--

DiagramLinks Class

A DiagramLink is an object that holds display information on a connector between two elements in a specific diagram. It includes, for example, the custom points and display appearance. It can be accessed from the Diagram DiagramLinks collection.

Associated table in .EAP file

t_diagramlinks

DiagramLinks Attributes

Attribute	Remarks
ConnectorID	Long Notes: Read/Write The ID of the associated connector.
DiagramID	Long Notes: Read/Write The local ID for the associated diagram.
Geometry	String Notes: Read/Write The geometry associated with the current connector in this diagram.
HiddenLabels	Boolean Notes: Indicates if this connector's labels are hidden on the diagram.
InstanceID	Long Notes: Read only The connector identifier for the current model.
IsHidden	Boolean Notes: Read/Write Indicates if this item is hidden or not.
LineColor	Long Notes: Sets the line color of the connector. Set to -1 to reset to the default color in the model.
LineStyle	Long Notes: Sets the line style of the connector. 1 = Direct 2 = Auto Routing

	3 = Custom Line 4 = Tree Vertical 5 = Tree Horizontal 6 = Lateral Vertical 7 = Lateral Horizontal 8 = Orthogonal Square 9 = Orthogonal Rounded
LineWidth	Long Notes: Sets the line width of the connector.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Path	String Notes: Read/Write The path of the connector in this diagram.
SourceInstanceUID	String Notes: Read only Returns the Unique Identifier of the source object.
SuppressSegment	Boolean Notes: Indicates whether the connector segments are suppressed.
Style	String Notes: Read/Write Additional style information; for example, color or thickness.
TargetInstanceUID	String Notes: Read only Returns the Unique Identifier of the target object.

DiagramLinks Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.
Update()	Boolean Notes: Update the current DiagramLink object after modification or appending a

	<p>new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>
--	---

DiagramObject Class

The DiagramObject Class stores presentation information that indicates what is displayed in a diagram and how it is shown.

Associated table in .EAP file

t_diagramobjects

DiagramObject Attributes

Attribute	Remarks
BackgroundColor	Long Notes: The background color of the object on the diagram. Set to -1 to re-set to the default color in the model.
BorderColor	Long Notes: The border line color of the object on the diagram. Set to -1 to re-set to the default color in the model.
BorderLineWidth	Long Notes: The border line width of the object on the diagram. Valid values are 1 (narrowest) to 5 (thickest); a default of 1 is applied if an invalid value is passed in.
Bottom	Long Notes: Read/Write The bottom edge position of the object on the diagram. Enterprise Architect uses a cartesian coordinate system, with {0,0} being the top-left corner of the diagram. For this reason, Y-axis values (Top and Bottom) should always be negative.
DiagramID	Long Notes: Read/Write The ID of the associated diagram.
ElementDisplayMode	Long Notes: Indicates how to adjust the element features if the element is resized. 1 = Resize to longest feature 2 = Wrap features 3 = Truncate features Defaults to 1 if an invalid value is supplied.
ElementID	Long Notes: Read/Write

	The ElementID of the object instance in this diagram.
FeatureStereotypesToHide	String Notes: Lists the stereotypes to hide on the object on the diagram.
FontBold	Boolean Notes: Get or Set the status of the object text font as Bold.
FontColor	Long Notes: The color of the font of the object text on the diagram.
FontItalic	Boolean Notes: Get or Set the status of the object text font as Italic.
FontName	String Notes: The name of the font used for the object text.
FontSize	String Notes: The size of the font used for the object text.
FontUnderline	Boolean Notes: Get or Set the status of the object text font as Underlined.
InstanceGUID	String Notes: The instance GUID for the object on the diagram (the DUID).
InstanceID	Long Notes: Read Holds the connector identifier for the current model.
IsSelectable	Boolean Notes: Indicates whether this object on the diagram can be selected.
Left	Long Notes: Read/Write The left edge position of the object on the diagram.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Right	Long Notes: Read/Write The right edge position of the object on the diagram.
Sequence	Long Notes: Read/Write The sequence position when loading the object into the diagram (this affects its Z

	order). The Z-order is one-based and the lowest value is in the foreground.
ShowComposedDiagram	Boolean Notes: Indicates whether the object's composite diagram should be displayed by default when the object is selected.
ShowConstraints	Boolean Notes: Show constraints for this object on the diagram.
ShowFormattedNotes	Boolean Notes: Show any formatting applied to the notes, for this object on the diagram. ShowNotes must be True for the formatted notes to be displayed.
ShowFullyQualifiedTags	Boolean Notes: Show fully qualified Tagged Values for this object on the diagram.
ShowInheritedAttributes	Boolean Notes: Show inherited attributes for this object on the diagram.
ShowInheritedConstraints	Boolean Notes: Show inherited constraints for this object on the diagram.
ShowInheritedOperations	Boolean Notes: Show inherited operations for this object on the diagram.
ShowInheritedResponsibilities	Boolean Notes: Show inherited responsibilities for this object on the diagram.
ShowInheritedTags	Boolean Notes: Show inherited Tagged Values for this object on the diagram.
ShowNotes	Boolean Note: Show the notes for this object on the diagram.
ShowPackageAttributes	Boolean Notes: Show Package attributes for this object on the diagram.
ShowPackageOperations	Boolean Notes: Show Package operations for this object on the diagram.
ShowPortType	Boolean Notes: Show the Port type.
ShowPrivateAttributes	Boolean Notes: Show private attributes for this object on the diagram.
ShowPrivateOperations	Boolean Notes: Show private operations for this object on the diagram.

ShowProtectedAttributes	Boolean Notes: Show protected attributes for this object on the diagram.
ShowProtectedOperations	Boolean Notes: Show protected operations for this object on the diagram.
ShowPublicAttributes	Boolean Notes: Show public attributes for this object on the diagram.
ShowPublicOperations	Boolean Notes: Show public operations for this object on the diagram.
ShowResponsibilities	Boolean Notes: Show responsibilities for this object on the diagram.
ShowRunstates	Boolean Notes: Show Runstates for this object on the diagram.
ShowStructuredCompartmentments	Boolean Note: Indicates whether to display the Structure Compartments for this object on the diagram.
ShowTags	Boolean Notes: Show Tagged Values for this object on the diagram.
Style	Variant Notes: Read/Write The style information for this object. Returns a semi-colon delimited string that defines the current style settings. Changing a value will completely overwrite the previously existing value, so caution is advised to avoid losing existing style information that you want to keep. <i>See Setting the Style.</i>
TextAlign	Long Notes: Indicates the alignment of text on a Text element on the diagram. 1 = Left aligned 2 = Center aligned 3 = Right aligned Defaults to 1 if an invalid value is supplied.
Top	Long Notes: Read/Write The top edge position of the object on the diagram. Enterprise Architect uses a cartesian coordinate system, with {0,0} being the top-left corner of the diagram. For this reason, Y-axis values (Top and Bottom) should always be negative.

DiagramObject Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
ResetFont	Notes: Resets the font of the object text on the diagram back to the model default.
SetFontStyle(FontName, FontSize, Bold, Italic, Underline)	Boolean Notes: Sets the font of the object text on the diagram to the specified values.
SetStyleEx(string Parameter, string Value)	Void Notes: Sets an individual parameter of the Style string. Parameters: <ul style="list-style-type: none"> Parameter: String - the name of the style parameter to modify; for example: <ul style="list-style-type: none"> "BCol" = background color "BFol" = font color "LCol" = line color "LWth" = line width Value: String - the new value for the style parameter
Update()	Boolean Notes: Updates the current DiagramObject after modification or appending a new item If False is returned, check the GetLastError function for more information.

Setting the Style

The Style attribute contains various settings that affect the appearance of a DiagramObject. However, it is not recommended to directly edit this attribute string. Instead, use either the SetStyleEx method or one of the individual DiagramObject attributes such as BackgroundColor, FontColor or BorderColor.

For example, the Style string might contain a series of values in a format such as:

```
BCol=n;BFol=n;LCol=n;LWth=n;
```

where:

- BCol = Background Color
- BFol = Font Color
- LCol = Line Color
- LWth = Line Width

The value assigned to each of the Style color properties is a decimal representation of the hex RGB value, where Red=FF, Green=FF00 and Blue=FF0000.

This code snippet shows how you might change the style settings for all of the objects in the current diagram, changing the background color to red (FF=255) and the font and line colors to yellow (FFFF=65535):

```
For Each aDiagObj In aDiag.DiagramObjects
```

```
aDiagObj.BackgroundColor=255  
aDiagObj.FontColor=65535  
aDiagObj.BorderColor=65535  
aDiagObj.BorderLineWidth=1  
aDiagObj.Update  
aRepos.ReloadDiagram aDiagObj.DiagramID
```

Next

SwimlaneDef Class

A SwimlaneDef object makes available attributes relating to a single row or column in a list of swimlanes.

SwimlaneDef Attributes

Attribute	Description
Bold	Boolean Notes: Read/Write Show the title text in bold.
FontColor	Long Notes: Read/Write The RGB color used to draw the titles.
HideClassifier	Boolean Notes: Read/Write Removes any classifier from the title display.
HideNames	Boolean Notes: Read/Write Set to True to hide the swimlane titles.
LineColor	Long Notes: Read/Write The RGB color used to draw swimlane borders.
LineWidth	Long Notes: Read/Write The width, in pixels, of the line used to draw swimlanes. Valid values are 1, 2 or 3.
Locked	Boolean Notes: Read/Write If set to True, disables user modification of the swimlanes via the diagram.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Orientation	String Notes: Read/Write Indicates whether the swimlanes are vertical or horizontal.
ShowInTitleBar	Boolean Notes: Read/Write

	Enables vertical swimlane titles to be shown in the title bar.
Swimlanes	Swimlanes Notes: Read/Write A list of individual swimlanes.

Swimlanes Class

A Swimlanes object is attached to a diagram's SwimlaneDef object and provides a mechanism to access individual swimlanes.

Swimlanes Attributes

Attribute	Description
Count	Long Notes: Read/Write Gives the number of swimlanes.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

Swimlanes Methods

Method	Description
Add(string Title, long Width)	Swimlane Notes: Adds a new swimlane to the end of the list, and returns a swimlane object representing the newly added entry. Parameters: <ul style="list-style-type: none">Title: String - The title text that appears at the top of the swimlane; this can be the same as an existing swimlane titleWidth: Long - The width of the swimlane in pixels
Delete(object Index)	Void Notes: Deletes a selected swimlane. If the string matches more than one entry, only the first entry is deleted. Parameter: <ul style="list-style-type: none">Index: Object - Either a string representing the title text or an integer representing the zero-based index of the swimlane to delete
DeleteAll()	Void Notes: Removes all swimlanes.
Insert(long Index, string Title, long Width)	Swimlane Notes: Inserts a swimlane at a specific position, and returns a swimlane object representing the newly added entry. Parameters: <ul style="list-style-type: none">Index: Long - The zero-based index of the existing Swimlane before which this

	<p>new entry is inserted</p> <ul style="list-style-type: none">• Title: String - The title text which appears at the top of the swimlane; this can be the same as an existing swimlane title• Width: Long - The width of the swimlane in pixels
Items(object Index)	<p>Swimlane collection</p> <p>Notes: Accesses an individual swimlane.</p> <p>If the string matches more than one swimlane title, the first matching swimlane is returned.</p> <p>Parameter:</p> <ul style="list-style-type: none">• Index: Object - Either a string representing the title text or an integer representing the zero-based index of the swimlane to get

Swimlane Class

A Swimlane object makes available attributes relating to a single row or column in a list of swimlanes.

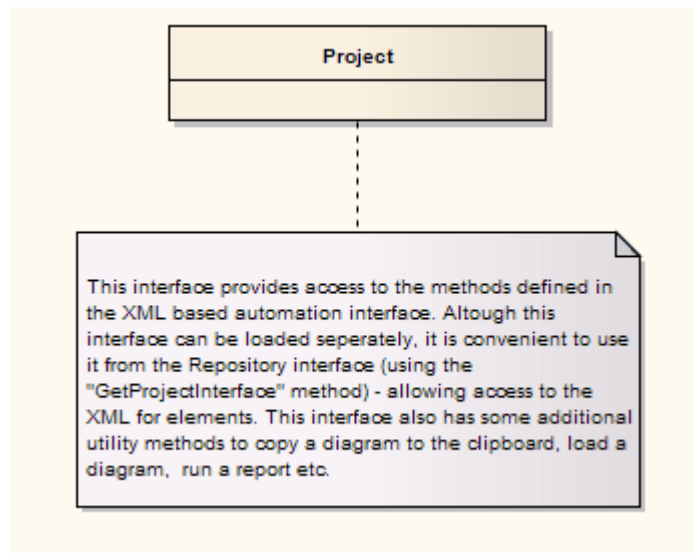
Swimlane Attributes

Attribute	Description
BackColor	Long Notes: Read/Write The RGB color that the swimlane is filled with.
ClassifiedGuid	String Notes: Read/Write The GUID of the classifier Class. This can be obtained from the corresponding element object via the ElementGUID property.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Title	String Notes: Read/Write The text at the head of the swimlane.
Width	Long Notes: Read/Write The width of the swimlane, in pixels.

Project Interface Package

The Enterprise Architect.Project interface. This is the interface to Enterprise Architect elements; it also includes some utility functions. You can get a pointer to this interface using the Repository.GetProjectInterface method.

Example



Project Class

The Project interface can be accessed from the Repository using `GetProjectInterface()`. The returned interface provides access to the XML-based Enterprise Architect Automation Interface. Use this interface to get XML for the various internal elements and to run some utility functions to perform tasks such as load diagrams or run reports.

Project Attributes

Attribute	Remarks
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

Project Methods

Method	Remarks
CancelValidation ()	Void Notes: Cancels a validation process.
CanValidate ()	Boolean Notes: Returns a value to indicate that the Model Validation component is loaded.
CreateBaseline (string PackageGUID, string Version, string Notes)	Boolean Notes: Creates a Baseline of a specified Package. Parameters: <ul style="list-style-type: none"> PackageGUID: String - the GUID (in XML format) of the Package to Baseline Version: String - the version of the Baseline Notes: String - any notes concerning the Baseline
CreateBaselineEx (string PackageGUID, string Version, string Notes, EA.CreateBaselineFlag Flags)	Boolean Notes: Creates a Baseline of a specified Package, with a flag to exclude Package contents below the first level. Parameters: <ul style="list-style-type: none"> PackageGUID: String - the GUID (in XML format) of the Package to be Baseline Version: String - the version of the Baseline Notes: String - any notes concerning the Baseline Flags: EA.CreateBaselineFlag - whether or not to exclude the Package contents below the first level
DefineRule (string CategoryID, EA.EnumMVErrType)	String Notes: Defines the individual rules that can be performed during model validation.

ErrorType, string ErrorMessage)	<p>It must be called once for each rule from the EA_OnInitializeUserRules broadcast handler.</p> <p>The return value is a RuleId, which can be used for reference purposes when an individual rule is executed by Enterprise Architect during model validation.</p> <p>See the <i>Model Validation Example</i> for a detailed example of the use of this method.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • CategoryId: String - should be passed the return value from the DefineRuleCategory method • ErrorType: EA.EnumMVErrortype - depending on the severity of the error being validated, can be: <ul style="list-style-type: none"> - mvErrorCritical - mvError - mvWarning, or - mvInformation • ErrorMessage: String - can contain a default error string, although this is probably overridden by the PublishResult call
DefineRuleCategory (string CategoryName)	<p>String</p> <p>Notes: Defines a category of rules that can be performed during model validation (there is typically one category per Add-In). It must be called once from the EA_OnInitializeUserRules broadcast handler.</p> <p>The return value is a CategoryId that must to be passed to the DefineRule method.</p> <p>See the <i>Model Validation Example</i> for a detailed example of the use of this method.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • CategoryName: String - a text string that is visible in the 'Model Validation Configuration' dialog
DeleteBaseline (string BaselineGUID)	<p>Boolean</p> <p>Notes: Deletes a Baseline, identified by the BaselineGUID, from the repository.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • BaselineGUID: String - the GUID (in XML format) of the Baseline to delete
DoBaselineCompare (string PackageGUID, string Baseline, string ConnectString)	<p>String</p> <p>Notes: Performs a Baseline comparison using the supplied Package GUID and Baseline GUID (obtained in the result list from GetBaselines).</p> <p>Optionally you can include the connection string required to find the Baseline if it exists in a different model file.</p> <p>This method returns a log file of the status of all elements found and compared in the difference procedure. You can use this log information as input to DoBaselineMerge - automatically merging information from the Baseline.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • PackageGUID: String - the GUID (in XML format) of the Package to run the comparison on • Baseline: String - the GUID (in XML format) of the Baseline to run the comparison on • ConnectString: String - the location of the external .eap file or DBMS to extract the Baseline from
DoBaselineMerge (string PackageGUID, string Baseline, string	<p>String</p> <p>Notes: Performs a batch merge based on instructions contained in an XML file (MergeInstructions). You can supply an optional connection string if the Baseline is</p>

MergeInstructions, string ConnectString)	<p>located in another model.</p> <p>In the MergeInstructions file, each MergeItem node supplies the GUID of a differenced item from the XML difference log. As the merge is uni-directional and actioned in only one possible way, no additional arguments are required. Enterprise Architect chooses the correct procedure based on the 'Difference' results.</p> <pre><Merge> <MergeItem guid="{XXXXXX}" /> <MergeItem guid="{XXXXXX}" /> </Merge></pre> <p>Alternatively, you can supply a single Mergeitem with a GUID of RestoreAll. In this case, Enterprise Architect batch-processes ALL differences.</p> <pre><Merge> <MergeItem guid="RestoreAll" changed="true" baselineOnly="true" modelOnly="true" moved="true" fullRestore="false" /> </Merge></pre> <p>Parameters:</p> <ul style="list-style-type: none"> PackageGUID: String - the GUID (in XML format) of the Package to merge the Baseline into Baseline: String - the GUID of the Baseline (in XML format) to merge into the Package MergeInstructions: String - the file containing the GUID of each differenced item from the XML difference log returned by DoBaselineCompare() ConnectString: String - the location of the EAP file or DBMS to get the Baseline from, if not in the same model as the Package
EnumDiagramElements (string DiagramGUID)	<p>protected abstract: String</p> <p>Notes: Gets an XML list of all elements in a diagram.</p> <p>Parameters:</p> <ul style="list-style-type: none"> DiagramGUID: String - the GUID (in XML format) of the diagram to get elements for
EnumDiagrams (string PackageGUID)	<p>protected abstract: String</p> <p>Notes: Gets an XML list of all diagrams in a specified Package.</p> <p>Parameters:</p> <ul style="list-style-type: none"> PackageGUID: String - the GUID (in XML format) of the Package to list diagrams for
EnumElements (string PackageGUID)	<p>protected abstract: String</p> <p>Notes: Gets an XML list of elements in a specified Package.</p> <p>Parameters:</p> <ul style="list-style-type: none"> PackageGUID: String - the GUID (in XML format) of the Package to get a list of elements for
EnumLinks (string ElementGUID)	<p>protected abstract: String</p> <p>Notes: Gets an XML list of connectors for a specified element.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ElementGUID: String - the GUID (in XML format) of the element to get all associated connectors for

EnumPackages (string PackageGUID)	protected abstract: String Notes: Gets an XML list of child Packages inside a parent Package. Parameters: <ul style="list-style-type: none">PackageGUID: String - the GUID (in XML format) of the parent Package
EnumProjects ()	protected abstract: String Notes: Gets a list of projects in the current file; corresponds to Models in Repository.
EnumViews ()	protected abstract: String Notes: Enumerates the Views for a project. Returned as an XML document.
EnumViewEx (string ProjectGUID)	protected abstract: String Notes: Gets a list of Views in the current project. Parameters: <ul style="list-style-type: none">ProjectGUID: String - the GUID (in XML format) of the project to get views for
Exit ()	protected abstract: String Notes: Exits the current instance of Enterprise Architect; this function is maintained for backward compatibility and should never be called. Enterprise Architect automatically exits when you are no longer using any of the provided objects.
ExportPackageXMI (string PackageGUID, enumXMIMType XMIMType, long DiagramXML, long DiagramImage, long FormatXML, long UseDTD, string FileName)	protected abstract: String Notes: Exports XMI for a specified Package. Parameters: <ul style="list-style-type: none">PackageGUID: String - the GUID (in XML format) of the Package to be exportedXMIMType: EnumXMIMType - specifies the XMI type and version information; see <i>XMIMType Enum</i> for accepted valuesDiagramXML: Long - True if XML for diagrams is required; accepted values:<ul style="list-style-type: none">0 = Do not export diagrams1 = Export diagrams2 = Export diagrams along with alternate imagesDiagramImage: Long - the format for diagram images to be created at the same time; accepted values:<ul style="list-style-type: none">-1 = NONE0 = EMF1 = BMP2 = GIF3 = PNG4 = JPGFormatXML: Long - True if XML output should be formatted prior to savingUseDTD: Long - True if a DTD should be usedFileName: String - the filename to output to
ExportPackageXMIMEx (string PackageGUID, enumXMIMType XMIMType, long DiagramXML, long	protected abstract: String Notes: Exports XMI for a specified Package, with a flag to determine whether the export includes Package content below the first level.

DiagramImage, long FormatXML, long UseDTD, string FileName, ea.ExportPackageXMIFlag Flags)	<p>Parameters:</p> <ul style="list-style-type: none"> PackageGUID: String - the GUID (in XML format) of the Package to be exported XMIType: EnumXMIType - specifies the XMI type and version information; see <i>XMIType Enum</i> for accepted values DiagramXML: Long - true if XML for diagrams is required; accepted values: 0 = Do not export diagrams 1 = Export diagrams 2 = Export diagrams along with alternate images DiagramImage: Long - the format for diagram images to be created at the same time; accepted values: -1 =NONE 0 =EMF 1 =BMP 2 =GIF 3 =PNG 4 =JPG FormatXML: Long - True if XML output should be formatted prior to saving UseDTD: Long - True if a DTD should be used. FileName: String - the filename to output to Flags: ea.ExportPackageXMIFlag - specify whether or not to include Package content below the first level (currently supported for xmiEADefault), whether or not to exclude tool-specific information from export
GenerateClass (string ElementGUID, string ExtraOptions)	<p>Boolean</p> <p>Notes: Generates the code for a single Class.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ElementGUID: String - the GUID (in XML format) of the element to generate ExtraOptions: String - enables extra options to be given to the command; currently unused
GenerateDiagramFromScenario (string ElementGUID, EnumScenarioDiagramType DiagramType, long OverwriteExistingDiagram)	<p>Boolean</p> <p>Notes: Generates various diagrams from the Structured Specification of an element.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ElementGUID: String - the GUID (in XML format) of the element containing the Structured Specification DiagramType: EnumScenarioDiagramType - the type of diagram to generate; see <i>ScenarioDiagramType Enum</i> for accepted values OverwriteExistingDiagram: Long - determines whether to overwrite the existing diagram or synchronize the existing elements with the scenario steps 0 = Delete the existing diagram and elements, and create a new diagram and elements 1 = Synchronize existing elements with the scenario steps and preserve the diagram layout 2 = Synchronize existing elements with the scenario steps and re-cast the diagram layout 3 = Do not generate a diagram if one already exists
GenerateElementDDL (string ElementGUID, string FileName, string ExtraOptions)	<p>Boolean</p> <p>Notes: Generates DDL for an element using the options that are currently set on the Generate DDL screen.</p>

GeneratePackage (string PackageGUID, string ExtraOptions)	<p>Boolean</p> <p>Notes: Generates the code for all Classes within a Package.</p> <p>For example:</p> <pre>recurse=1;overwrite=1;dir=C:\</pre> <p>Parameters:</p> <ul style="list-style-type: none"> PackageGUID: String - the GUID (in XML format) of the Package to generate code for ExtraOptions: String - enables extra options to be given to the command; currently enables: <ul style="list-style-type: none"> Generation of all sub-Packages (recurse) Force overwrite of all files (overwrite) and Specification to auto generate all paths (dir)
GeneratePackageDDL (string PackageGUID, string FileName, string ExtraOptions)	<p>Boolean</p> <p>Notes: Generates DDL for all elements in a Package using the options that are currently set on the Generate DDL screen.</p>
GenerateTestFromScenario (string ElementGUID, EnumScenarioTestType TestType)	<p>Boolean</p> <p>Notes: Generates a Vertical Test Suite, a Horizontal Test Suite, an Internal test or an External test from the Structured Specification of an element.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ElementGUID: String - the GUID (in XML format) of the element containing the Structured Specification TestType: EnumScenarioTestType - the type of test to generate; see <i>ScenarioTestType Enum</i> for accepted values
GenerateWSDL(string WSDLComponentGUID, string Filename, string Encoding, string ExtraOptions)	<p>Boolean</p> <p>Notes: Generates WSDL for the specified WSDL stereotyped Component.</p> <p>Parameters:</p> <ul style="list-style-type: none"> WSDLComponentGUID: String - the GUID (in XML format) of the WSDL stereotyped Component Filename: String - the target file path Encoding: String - the XML encoding for the code page instruction ExtraOptions: String - enables extra options to be given to the command; currently unused
GenerateXSD (string PackageGUID, string FileName, string Encoding, string Options)	<p>Boolean</p> <p>Notes: Creates an XML schema for a Package, specified by its GUID. Returns True on success.</p> <p>Parameters:</p> <ul style="list-style-type: none"> PackageGUID: String - the GUID (in XML format) of the Package FileName: String - the target filepath Encoding: String - the XML encoding for the code page instruction Options: String - enables extra options to be given to the command, in a comma-separated string; currently enables: <ul style="list-style-type: none"> GenGlobalElement - turn the generation of global elements for all global ComplexTypes On or Off; for example: GenGlobalElement=1 UseRelativePath - turns on or off the option to use a relative path in the XSD import or XSD include statement

	<p>when referencing external Package, provided the schemaLocation tag is empty on the referenced Packages; for example: UseRelativePath=1</p>
GetBaselines (string PackageGUID, string ConnectString)	<p>String</p> <p>Notes: Returns a list (in XML format) of Baselines associated with the supplied Package GUID. Optionally, you can provide a connection string to get Baselines from the same Package, but located in a different model file (or DBMS).</p> <p>Parameters:</p> <ul style="list-style-type: none"> PackageGUID: String - the GUID (in XML format) of the Package to get Baselines for ConnectString: String - the location of the EAP file or DBMS to get the Baselines from, if not in the same model as the Package
GetDiagram (string DiagramGUID)	<p>protected abstract: String</p> <p>Notes: Gets the diagram details, in XML format.</p> <p>Parameters:</p> <ul style="list-style-type: none"> DiagramGUID: String - the GUID (in XML format) of the diagram to get details for
GetElement (string ElementGUID)	<p>protected abstract: String</p> <p>Notes: Gets XML for the specified element.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ElementGUID: String - the GUID (in XML format) of the element to retrieve XML for
GetElementConstraints (string ElementGUID)	<p>protected abstract: String</p> <p>Notes: Gets constraints for an element, in XML format.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ElementGUID: String - the GUID (in XML format) of the element
GetElementEffort (string ElementGUID)	<p>protected abstract: String</p> <p>Notes: Gets efforts for an element, in XML format.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ElementGUID: String - the GUID (in XML format) of the element
GetElementFiles (string ElementGUID)	<p>protected abstract: String</p> <p>Notes: Gets metrics for an element, in XML format.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ElementGUID: String - the GUID (in XML format) of the element
GetElementMetrics (string ElementGUID)	<p>protected abstract: String</p> <p>Notes: Gets files for an element, in XML format.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ElementGUID: String - the GUID (in XML format) of the element
GetElementProblems (string ElementGUID)	<p>protected abstract: String</p> <p>Notes: Gets a list of issues (problems) associated with an element, in XML format.</p> <p>Parameters:</p>

	<ul style="list-style-type: none"> ElementGUID: String - the GUID (in XML format) of the element
GetElementProperties (string ElementGUID)	<p>protected abstract: String</p> <p>Notes: Gets Tagged Values for an element, in XML format.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ElementGUID: String - the GUID (in XML format) of the element
GetElementRequirements (string ElementGUID)	<p>protected abstract: String</p> <p>Notes: Gets a list of requirements for an element, in XML format.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ElementGUID: String -the GUID (in XML format) of the element
GetElementResources (string ElementGUID)	<p>protected abstract: String</p> <p>Notes: Gets a list of resources for an element, in XML format.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ElementGUID: String - the GUID (in XML format) of the element
GetElementRisks (string ElementGUID)	<p>protected abstract: String</p> <p>Notes: Gets a list of risks associated with an element, in XML format.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ElementGUID: String - the GUID (in XML format) of the element
GetElementScenarios (string ElementGUID)	<p>protected abstract: String</p> <p>Notes: Gets a list of scenarios for an element, in XML format.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ElementGUID: String - the GUID (in XML format) of the element
GetElementTests (string ElementGUID)	<p>protected abstract: String</p> <p>Notes: Gets a list of tests for an element, in XML format.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ElementGUID: String - the GUID (in XML format) of the element
GetFileNameDialog (string Filename, string FilterString, long FilterIndex, long Flags, string InitialDirectory, long OpenOrSave)	<p>String</p> <p>Notes: Opens a standard 'File Open' or 'Save As' dialog and returns a string containing the full path to the selected file on success. Returns an empty string if the dialog was canceled.</p> <p>For example:</p> <pre> Filename = "" FilterString = "CSV Files (*.csv) *.csv All Files (*.*) *.* " Filterindex = 1 Flags = &H2 'OFN_OVERWRITEPROMPT InitialDirectory = "" OpenOrSave = 1 filepath = Project.GetFileNameDialog (Filename, FilterString, Filterindex, Flags, InitialDirectory, OpenOrSave) </pre> <p>In this example, the 'Save As' dialog will prompt for a CSV file.</p> <p>Parameters:</p>

	<ul style="list-style-type: none"> • Filename: String - default filename specified in the dialog • FilterString: String - delimited list of available file type filters • Filterindex: Long - one-based index of the filter to be used by default • Flags: Long - additional bit flags used to initialize the file dialog; see the OPENFILENAME structure in MSDN documentation for accepted values • InitialDirectory: String - directory path to open this dialog • OpenOrSave: Long - show dialog as an 'Open' or 'Save As' style dialog; accepted values: 0 = Open, 1 = Save As
GetLastError ()	<p>protected abstract: String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p>
GetLink (string LinkGUID)	<p>protected abstract: String</p> <p>Notes: Gets connector details, in XML format.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • LinkGUID: String - the GUID (in XML format) of the connector to get details of
GUIDtoXML (string GUID)	<p>String</p> <p>Notes: Changes an internal GUID to the form used in XML.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • GUID: String - the Enterprise Architect style GUID to convert to XML format
ImportDirectory (string PackageGUID, string Language, string DirectoryPath, string ExtraOptions)	<p>Boolean</p> <p>Notes: Imports a source code directory into the model.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • PackageGUID: String - the GUID (in XML format) of the Package to reverse engineer code into • Language: String - specifies the language of the code to be imported • DirectoryPath: String - specifies the path where the code is found on the computer • ExtraOptions: String - enables extra options to be given to the command; currently enables import of source from all child directories (recurse) - for example: recurse=1
ImportFile (string PackageGUID, string Language, string FileName, string ExtraOptions)	<p>Boolean</p> <p>Notes: Imports an individual file or binary module into the model, in a Package per namespace style import.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • PackageGUID: String - the GUID (in XML format) of the Package to reverse engineer code into; this is expected to be a namespace root Package • Language: String - specifies the language of the code to be imported Use the value 'DNPE' to import a binary module; this imports a .NET assembly or Java .class file, but not a .jar file • Filename: String - specifies the path where the code or module is found on the computer • ExtraOptions: String - enables extra options to be given to the command; currently unused

ImportPackageXMI (string PackageGUID, string Filename, long ImportDiagrams, long StripGUID)	<p>String</p> <p>Notes: Imports an XMI file at a point in the tree. Returns an empty string if successful, or returns an error message on failure.</p> <p>Parameters:</p> <ul style="list-style-type: none"> PackageGUID: String - the GUID (in XML format) of the target Package to import the XMI file into (or overwrite with the XMI file) Filename or XMLText: String - the name of the XMI file; if the String is of type filename it is interpreted as a source file, otherwise the String is imported as XML text ImportDiagrams: Long - 1 for importing diagrams and 0 to skip importing diagrams StripGUID: Long <ul style="list-style-type: none"> - 1 to replace the element UniqueIDs on import; if stripped, then a copy of the Package could be imported into the same Enterprise Architect model as two different versions - 0 to retain the element UniqueIDs on import; a duplicate copy of the Package cannot be created in the same model of Enterprise Architect
LayoutDiagram (string DiagramGUID, long LayoutStyle)	<p>Boolean</p> <p>Notes: Deprecated. it is recommended that LayoutDiagramEx is used instead. Calls the function to automatically layout a diagram in hierarchical fashion. It is only recommended for Class and Object diagrams.</p> <p>Parameters:</p> <ul style="list-style-type: none"> DiagramGUID: String - the GUID (in XML format) of the diagram to lay out LayoutStyle: Long - always ignored
LayoutDiagramEx (string DiagramGUID, long LayoutStyle, long Iterations, long LayerSpacing, long ColumnSpacing, boolean SaveToDiagram)	<p>Boolean</p> <p>Notes: Calls the function to automatically layout a diagram in hierarchical fashion. It is only recommended for Class and Object diagrams.</p> <p>LayoutStyle accepts these options</p> <ul style="list-style-type: none"> Default Options: <ul style="list-style-type: none"> - lsDiagramDefault - lsProgramDefault Cycle Removal Options: <ul style="list-style-type: none"> - lsCycleRemoveGreedy - lsCycleRemoveDFS Layering Options: <ul style="list-style-type: none"> - lsLayeringLongestPathSink - lsLayeringLongestPathSource - lsLayeringOptimalLinkLength Initialize Options: <ul style="list-style-type: none"> - lsInitializeNaive - lsInitializeDFSOut - lsInitializeDFSIn Crossing Reduction Option: <ul style="list-style-type: none"> - lsCrossReduceAggressive Layout Options - Direction

	<ul style="list-style-type: none"> - IsLayoutDirectionUp - IsLayoutDirectionDown - IsLayoutDirectionLeft - IsLayoutDirectionRight <p>Parameters:</p> <ul style="list-style-type: none"> • DiagramGUID: String - the GUID (in XML format) of the diagram to lay out • LayoutStyle: Long - the layout style • Iterations: Long - the number of layout iterations the Layout process should take to perform cross reduction (Default value = 4) • LayerSpacing: Long - the per-element layer spacing the Layout process should use (Default value = 20) • ColumnSpacing: Long - the per-element column spacing the Layout process should use (Default value = 20) • SaveToDiagram: Boolean - specifies whether or not Enterprise Architect should save the supplied layout options as default to the diagram in question
LoadControlledPackage (string PackageGUID)	<p>String</p> <p>Notes: Loads a Package that has been marked and configured as controlled. The filename details are stored in the Package control data.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • PackageGUID: String - the GUID (in XML format) of the Package to load
LoadDiagram (string DiagramGUID)	<p>protected abstract: Boolean</p> <p>Notes: Loads a diagram by its GUID.</p> <p>Parameter:</p> <ul style="list-style-type: none"> • DiagramGUID: String - the GUID (in XML format) of the diagram to load; if you retrieve the GUID using the Diagram interface, use the GUIDtoXML function to convert it to XML format
LoadProject (string FileName)	<p>protected abstract: Boolean</p> <p>Notes: Loads an Enterprise Architect project file.</p> <p>Do not use this method if you have accessed the Project interface from the Repository, which has already loaded a file.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • FileName: String - the name of the project file to load
Migrate (string GUID, string SourceType, string DestinationType)	<p>Void</p> <p>Notes: Migrates a model (or part of a model) from one BPMN, ArchiMate or SysML format to an upgraded format.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • GUID: String - the GUID of the Package or element for which the contents are to be migrated • SourceType: String - the type of model to be upgraded; accepted values: <ul style="list-style-type: none"> - BPMN - BPMN1.1 - UPDM - SysML1.1 - SysML1.2 - SysML1.3 - ArchiMate

	<ul style="list-style-type: none"> - ArchiMate2 - UPDM2 • DestinationType: String - the type of model to upgrade to; accepted values: <ul style="list-style-type: none"> - BPMN1.1 - BPMN1.1::BPEL - BPMN2.0 - UPDM2 - SysML1.2 - SysML1.3 - SysML1.4 - ArchiMate2 - ArchiMate3 - UAF
MigrateToBPMN11 (string GUID, string Type)	<p>Void</p> <p>Notes: Migrates every BPMN 1.0 construct in a Package or an element (including elements, attributes, diagrams and connectors) to BPMN 1.1.</p> <p>Parameters</p> <ul style="list-style-type: none"> • GUID: String - the GUID of the Package or element for which the contents are to be migrated to BPMN 1.1 • Type: String - the type of upgrade, either just to BPMN 1.1 or to BPMN 1.1 and BPEL. Accepted values are: <ul style="list-style-type: none"> - BPMN = migrate to BPMN 1.1 - BPEL = migrate to BPMN 1.1 and update: <ul style="list-style-type: none"> - any diagram with stereotype BPMN to BPEL - any element with stereotype BusinessProcess to BPELProcess <p>Migrating to BPEL is possible only in the Ultimate or Business and Software Engineering editions of Enterprise Architect.</p>
ProjectTransfer (string SourceFilePath, string TargetFilePath, string LogFilePath)	<p>Boolean</p> <p>Notes: Transfers the project from a .eap file or DBMS to a .eap file.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • SourceFilePath: String - the path of the source file to transfer • TargetFilePath: String - the path of the target file; Enterprise Architect creates a new Base project in this location • LogFilePath: String - the path of the log file where the status of the transfer process is updated <p>In automation, the target file does not have to exist; the file path is enough. Enterprise Architect creates a new, empty Base.eap file and transfers the source project into it.</p>
PublishResult (string CategoryID, EA.EnumMVErrortype ErrorType, string ErrorMessage)	<p>String</p> <p>Notes: Returns the results of each rule that can be performed during model validation. It must be called once for each rule from the EA_OnInitializeUserRules broadcast handler.</p> <p>The return value is a RuleId, which can be used for reference purposes when an individual rule is executed by Enterprise Architect during model validation. See the Model Validation Example for a detailed example of the use of this method.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • CategoryId: String - should be passed the return value from the DefineRuleCategory method • ErrorType: EA.EnumMVErrortype - depending on the severity of the error being validated, can be:

	<ul style="list-style-type: none"> - mvErrorCritical - mvError - mvWarning, or - mvInformation <ul style="list-style-type: none"> • ErrorMessage: String - contains an error string
PutDiagramImageOnClipboard (string DiagramGUID, long Type)	<p>protected abstract: Boolean</p> <p>Notes: Copies an image of the specified diagram to the clipboard.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • DiagramGUID: String - the GUID (in XML format) of the diagram to copy • Type: Long - the file type <ul style="list-style-type: none"> - If Type = 0 then it is a metafile - If Type = 1 then it is a Device Independent Bitmap
PutDiagramImageToFile (string Diagram GUID, string FileName, long Type)	<p>protected abstract: Boolean</p> <p>Notes: Saves an image of the specified diagram to file.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • DiagramGUID: String - the GUID (in XML format) of the diagram to save • FileName: String - the name of the file to save the diagram into • Type: Long - the file type <ul style="list-style-type: none"> - If type = 0 then it is a metafile - If type = 1 then it uses the file type from the name extension (that is, .bmp, .jpg, .gif, .png, .tga)
ReloadProject ()	<p>protected abstract: Boolean</p> <p>Notes: Reloads the current project.</p> <p>This is a convenient method to refresh the current loaded project (in case of outside changes to the .eap file).</p>
RunModelSearch (string Search, string SearchTerm, bool ShowInEA)	<p>Void</p> <p>Notes: Invokes the Model Search component.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Search: String - the name of an Enterprise Architect defined search • SearchTerm: String - the term to search for in the project • ShowInEA: Boolean - execute the search and output in the Model Search window
RunReport (string PackageGUID, string TemplateName, string Filename)	<p>protected abstract: Void</p> <p>Notes: Runs a named document report.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • PackageGUID: String - the GUID of the Package or master document to run the report on • TemplateName: String - the document report template to use; if the PackageGUID has a stereotype of MasterDocument, the template is not required • FileName: String - the file name and path to store the generated report; the file extension specified will determine the format of the generated document - for example, RTF, PDF
RunHTMLReport (string	String

PackageGUID, string ExportPath, string ImageFormat, string Style, string Extension)	<p>Notes: Runs an HTML report (as for 'Documentation HTML Documentation' when you right-click on a Package in the Project Browser).</p> <p>Parameters:</p> <ul style="list-style-type: none"> PackageGUID: String - the GUID (in XML format) of the Package or master document to run the report on ExportPath: String - the directory path to store the generated report files ImageFormat: String - file format in which to store images - .png or .gif Style: String - name of the web style template to apply; use <default> for the standard, system-provided template Extension: String - file extension for generated HTML files (example: .htm)
SaveControlledPackage (string PackageGUID)	<p>String</p> <p>Notes: Saves a Package that has been configured as a controlled Package, to XML. Only the Package GUID is required, Enterprise Architect picks the rest up from the Package control information.</p> <p>Parameter:</p> <ul style="list-style-type: none"> PackageGUID: String - the GUID (in XML format) of the Package to save
SaveDiagramImageToFile (string Filename)	<p>protected abstract: String</p> <p>Notes: Saves a diagram image of the current diagram to file.</p> <p>Parameters:</p> <ul style="list-style-type: none"> FileName: String - the filename of the image to save
ShowWindow (long Show)	<p>protected abstract: Void</p> <p>Notes: Shows or hides the Enterprise Architect User Interface.</p> <p>Parameters:</p> <ul style="list-style-type: none"> Show: Long
SynchronizeClass (string ElementGUID, string ExtraOptions)	<p>Boolean</p> <p>Notes: Synchronizes a Class with the latest source code.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ElementGUID: String - the GUID (in XML format) of the element to update from code ExtraOptions: String - enables extra options to be given to the command; currently unused
SynchronizePackage (string PackageGUID, string ExtraOptions)	<p>Boolean</p> <p>Notes: Synchronizes each Class in a Package with the latest source code.</p> <p>Parameters:</p> <ul style="list-style-type: none"> PackageGUID: String - the GUID (in XML format) of the Package containing the elements to update from code ExtraOptions: String - enables extra options to be given to the command; currently enables synchronization of all child Packages (children) - for example: children=1
TransformElement (string TransformName, string ElementGUID, string TargetPackage,	<p>Boolean</p> <p>Notes: Transforms an element into a Package.</p> <p>Parameters:</p> <ul style="list-style-type: none"> TransformName: String - specifies the transformation that should be executed

string ExtraOptions)	<ul style="list-style-type: none"> • ElementGUID: String - the GUID (in XML format) of the element to transform • TargetPackageGUID: String - the GUID (in XML format) of the Package to transform into • ExtraOptions: String - enables extra options to be given to the command: <ul style="list-style-type: none"> - GenCode=True / False - articulate code generation from the transformed elements; this option supercedes the current model setting
TransformPackage (string TransformName, string SourcePackage, string TargetPackage, string ExtraOptions)	<p>Boolean</p> <p>Notes: Runs a transformation on the contents of a Package.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • TransformName: String - specifies the transformation that should be executed • SourcePackageGUID: String - the GUID (in XML format) of the Package to transform • TargetPackageGUID: String - the GUID (in XML format) of the Package to transform into • ExtraOptions: String - enables extra options to be given to the command: <ul style="list-style-type: none"> - GenCode=True/False - articulate code generation from the transformed elements; this option supercedes the current model setting <ul style="list-style-type: none"> - SubPackages=True/False - specify if the child Packages are to be included whilst transforming a Package
ValidateDiagram (string DiagramGUID)	<p>Boolean</p> <p>Notes: Invokes the Enterprise Architect Model Validation component, then validates the diagram (for correctness) and the elements and connectors within the diagram.</p> <p>Output can be viewed through 'Show > Window > System Output > Model Validation'.</p> <p>Returns a boolean value to indicate the success or failure of the process, regardless of the results of the validation.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • DiagramGUID: String - the GUID of the Diagram Class object
ValidateElement (string ElementGUID)	<p>Boolean</p> <p>Notes: Invokes the Enterprise Architect Model Validation component, then validates the element and all child elements, diagrams, connectors, attributes and operations.</p> <p>Output can be viewed through 'Show > Window > System Output > Model Validation'.</p> <p>Returns a boolean value to indicate the success or failure of the process, regardless of the results of the validation.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ElementGUID: String - the GUID of the Element Class object
ValidatePackage (string PackageGUID)	<p>Boolean</p> <p>Notes: Invokes the Enterprise Architect Model Validation component, then validates the Package and all sub-Packages, elements, connectors and diagrams within it.</p> <p>Output can be viewed through 'Show > Window > System Output > Model</p>

	<p>Validation'.</p> <p>Returns a boolean value to indicate the success or failure of the process, regardless of the results of the validation.</p> <p>Parameters:</p> <ul style="list-style-type: none">• PackageGUID: String - the GUID of the Package Class object
XMLtoGUID (string GUID)	<p>String</p> <p>Notes: Changes a GUID in XML format to the form used inside Enterprise Architect.</p> <p>Parameters:</p> <ul style="list-style-type: none">• GUID: String - the XML style GUID to convert to Enterprise Architect internal format

Notes

- These methods all require input GUIDs in XML format; use **GUIDtoXML** to change the Enterprise Architect GUID to an XML GUID

Document Generator Interface Package

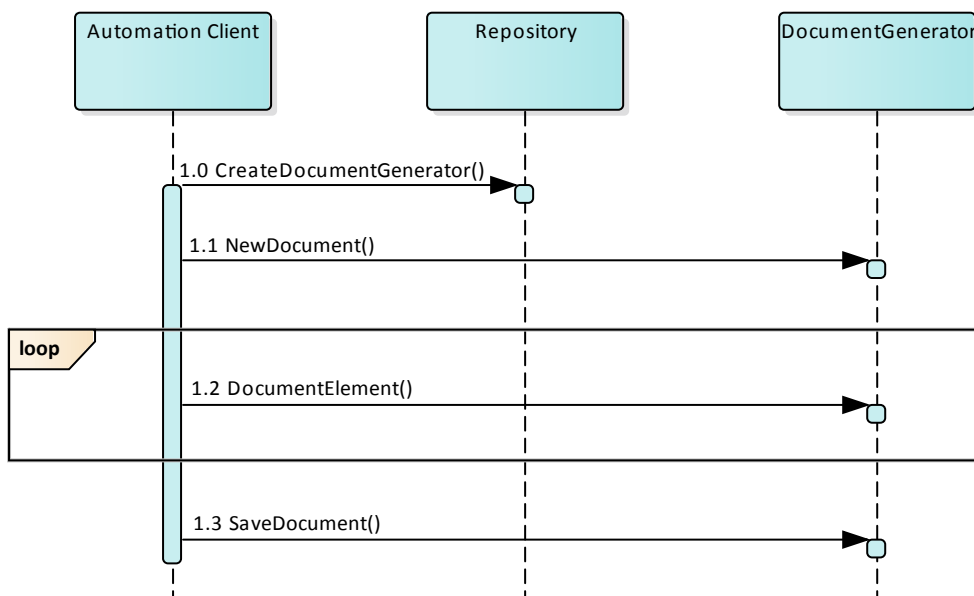
The DocumentGenerator Class provides an interface to the document and web reporting facilities, which you can use to generate reports on specific Packages, diagrams and elements in your model.

Access

You can create a pointer to this interface using the Repository.CreateDocumentGenerator method.

Example

This diagram illustrates how you might use the Document Generator interface in generating a report through the Automation Interface.



Also look at the:

- Document Generation scripting example in the Scripting window (Code > Tools > Scripting, then expand the 'Local Scripts' folder and double-click on 'JScript - Documentation Example')
- RunReport method in the Project Interface

DocumentGenerator Class

The DocumentGenerator Class provides an interface to the document and web reporting facilities, which you can use to generate reports on specific Packages, diagrams and elements in your model. This Class is accessed from the Repository Class using the CreateDocumentGenerator() method.

DocumentGenerator Attributes

Attribute	Remarks
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

DocumentGenerator Methods

Method	Remarks
DocumentConnector (long connectorID, long nDepth, string templateName)	Boolean Notes: Documents a connector. Parameters: <ul style="list-style-type: none">connectorId: Long - the ID of the connectornDepth: Long - the depth by which to adjust the heading leveltemplateName: String - the name of a template to use when documenting connectors; this can be blank
DocumentCustomData (string XML, long nDepth, string templateName)	Boolean Notes: Documents information based on the data supplied. Parameters: <ul style="list-style-type: none">XML: String - the XML of the data to be documentednDepth: Long - the depth by which to adjust the heading leveltemplateName: String - the name of a template to use when documenting custom data; this can be blank
DocumentDiagram (long diagramID, long nDepth, string templateName)	Boolean Notes: Documents a diagram. Parameters: <ul style="list-style-type: none">diagramId: Long - the ID of the diagramnDepth: Long - the depth by which to adjust the heading leveltemplateName: String - the name of a template to use when documenting diagrams; this can be blank
DocumentElement (long elementID, long nDepth,	Boolean Notes: Documents an element.

string templateName)	<p>Parameters:</p> <ul style="list-style-type: none"> • elementId: Long - the ID of the element • nDepth: Long - the depth by which to adjust the heading level • templateName: String - the name of a template to use when documenting elements; this can be blank
DocumentModelAuthor (string name, long nDepth, string templateName)	<p>Boolean</p> <p>Notes: Documents a model author.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • name: String - the name of the author • nDepth: Long - the depth by which to adjust the heading level • templateName: String - a template to use when documenting model authors; this can be blank
DocumentModelClient (string name, long nDepth, string templateName)	<p>Boolean</p> <p>Notes: Documents a single model client.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • name: String - the name of the client • nDepth: Long - the depth by which to adjust the heading level • templateName: String - a template to use when documenting model clients; this can be blank
DocumentModelGlossary (long id, long nDepth, string templateName)	<p>Boolean</p> <p>Notes: Documents a single model glossary term.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • id: Long - the ID of the term • nDepth: Long - the depth by which to adjust the heading level • templateName: String - a template to use when documenting model glossary terms; this can be blank
DocumentModelIssue (long id, long nDepth, string templateName)	<p>Boolean</p> <p>Notes: Documents a single model issue.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • id: Long - the ID of the issue • nDepth: Long - the depth by which to adjust the heading level • templateName: String - a template to use when documenting model issues; this can be blank
DocumentModelResource (string name, long nDepth, string templateName)	<p>Boolean</p> <p>Notes: Documents a single model resource.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • name: String - the name of the resource • nDepth: Long - the depth by which to adjust the heading level • templateName: String - a template to use when documenting model resources; this can be blank
DocumentModelRole (string name, long nDepth,	<p>Boolean</p> <p>Notes: Documents a single model role.</p>

string templateName)	<p>Parameters:</p> <ul style="list-style-type: none"> • name: String - the name of the role • nDepth: Long - the depth by which to adjust the heading level • templateName: String - a template to use when documenting model roles; this can be blank
DocumentModelTask (long id, long nDepth, string templateName)	<p>Boolean</p> <p>Notes: Documents a single model task.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • id: Long - the ID of the task • nDepth: Long - the depth by which to adjust the heading level • templateName: String - a template to use when documenting model tasks; this can be blank
DocumentPackage (long packageID, long nDepth, string templateName)	<p>Boolean</p> <p>Notes: Documents a Package.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • packageId: Long - the ID of the Package • nDepth: Long - the depth by which to adjust the heading level • templateName: String - a template to use when documenting Packages; this can be blank
GetDocumentAsRTF()	<p>Read Only.</p> <p>Returns a string value of the document in raw Rich Text Format.</p>
GetProjectConstant (string nameVal)	<p>String</p> <p>Notes: Returns the value of a Project Constant.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • nameVal: String - the name of the Project Constant for which to extract the value.
GetLastError ()	<p>String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p>
InsertBreak (long breakType)	<p>Boolean</p> <p>Notes: Inserts a break into the report at the current location.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • breakType: Long - 0 = page break, 1 = section break
InsertCoverPageDocument (string Name)	<p>Boolean</p> <p>Notes: Inserts the Coverpage into the document at the current location.</p> <p>The style sheet is applied to the document before it is insert into the generated document.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Name: String - the name of the Cover page document found in the Resource tree
InsertLinkedDocument	

(string guid)	<p>Boolean</p> <p>Notes: Inserts a linked document into the report at the current location.</p> <p>A linked document can used to set the header and footer of the report. These are taken from the first linked document added to the report.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • guid: String - the GUID of the element that has a linked document
InsertTableOfContents	<p>Boolean</p> <p>Notes: Inserts a Table of Contents at the current position.</p>
InsertTeamReviewPost (string path)	<p>Boolean</p> <p>Notes: Inserts a Team Review posting into the report at the current location.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • path: String - the path of the Team Review post
InsertTemplate (string templateName)	<p>Notes: Inserts the contents of the template directly into the report.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • templateName: String - the name of the template to use
InsertText (string text, string style)	<p>Boolean</p> <p>Notes: Inserts static text into the report at the current location.</p> <p>A carriage return is not included; if you need to use one, you can add it manually.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • text: String - the static text to be inserted • style: String - the name of the style in the template; defaults to Normal style
InsertTOCDocument (string name)	<p>Boolean</p> <p>Notes: Inserts the Table of Contents into the document at the current location.</p> <p>Note: The stylesheet is applied to the document before it is insert into the generated document.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • name: String - the name of the Table of Contents document found in the Resource tree
LoadDocument(string FileName)	<p>Boolean</p> <p>Notes: Inserts an external document into the currently generated file.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • FileName: String - the filename of an external document file to insert into the document.
SetProjectConstant (string newNameVal, string newValue)	<p>Boolean</p> <p>Notes: Sets a Project Constant for the documentation generator; this is saved in the current model.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • newNameVal: String - the name of the Project Constant • newValue: String - the value of the Project Constant
NewDocument (string	<p>Boolean</p>

templateName)	<p>Notes: Starts a new document; you call this before attempting to document anything else.</p> <p>Parameters:</p> <ul style="list-style-type: none"> templateName: String - the name of a template to use when documenting elements; this can be blank
ReplaceField (string fieldname, string fieldvalue)	<p>Boolean</p> <p>Notes: Replaces the 'Section' field identified by the fieldname parameter with the value provided in fieldvalue. For example:</p> <p>ReplaceField ("Element.Alias", "MyAlias")</p> <p>If you call this function more than once with the same fieldname, the field only has the most recent value set.</p> <p>Parameters:</p> <ul style="list-style-type: none"> fieldname: String - the field name to find (this does not include the {} braces) fieldvalue: String - the value to insert into the field; this can be a constant or a derived value
SaveDocument (string filename, long nDocType)	<p>Boolean</p> <p>Notes: Saves the document to disk.</p> <p>Parameters:</p> <ul style="list-style-type: none"> filename: String - the filename to save the file to nDocType: Long - 0 = RTF, 1 = HTML, 2 = PDF, 3 = DOCX
SetPageOrientation (long pageOrientation)	<p>Boolean</p> <p>Notes: Sets the current page orientation.</p> <p>Parameters:</p> <ul style="list-style-type: none"> pageOrientation: Long - 0 = Portrait, 1 = Landscape
SetStyleSheetDocument (string name)	<p>Boolean</p> <p>Notes: Sets the Stylesheet to be used for TOC, Coverpage and templates used. This can be called before NewDocument (above).</p> <p>Parameters:</p> <ul style="list-style-type: none"> name: String - the name of the stylesheet found in the Resource tree

Mail Interface Package

The MailInterface Package contains:

- A function to retrieve a pointer to the interface
- Functions to create and send a mail message within the current mode
- Utility functions for creating hyperlinks to selected model elements

You can get a pointer to this interface using the method `Repository.GetMailInterface`.

MailInterface Class

The MailInterface interface can be accessed from the Repository using GetMailInterface(). The returned interface provides access to the Enterprise Architect Model Mail Interface. Use this interface to automate the process of creating and sending messages using Enterprise Architect's Model Mail system.

MailInterface Attributes

Attribute	Remarks
MessagingEnabled	Boolean Notes: Read Only Advises whether messaging is enabled on the current model.
ObjectType	ObjectType Notes: Read Only Distinguishes objects referenced through a dispatch interface.

MailInterface Methods

Method	Remarks
ComposeMailMessage(string InitialRecipientGUID, string InitialSubject, messageflag InitialFlag, string InitialMessageText)	Boolean Notes: Creates a new mail message using the values specified in the input parameters; the message is displayed in the composition window, ready for sending. This method does NOT send the message. Parameters: <ul style="list-style-type: none"> InitialRecipientGUID: String - Initial value for the GUID of the addressee user (an Enterprise Architect user defined in the current model) InitialSubject: String - Initial value for the Subject text to display for this message InitialFlag: MessageFlag - Initial value for the flag type/color to attach to this message InitialMessageText: String - Initial value for the text that is the body of the message
GetAttributeHyperlink(string AttributeGUID, string LinkText)	String Notes: Returns a string containing a hyperlink to the attribute specified by the input parameter AttributeGUID. Parameters: <ul style="list-style-type: none"> AttributeGUID: String - The GUID of the attribute for which a hyperlink is required LinkText: String - The text to display for the hyperlink (such as the attribute name)

GetDiagramHyperlink (string DiagramGUID, string LinkText)	String Notes: Returns a string containing a hyperlink to the diagram specified by the input parameter DiagramGUID. Parameters: <ul style="list-style-type: none">• DiagramGUID: String - The GUID of the diagram for which a hyperlink is required• LinkText: String - The text to display for the hyperlink (such as the diagram name)
GetElementHyperlink (string ElementGUID, string LinkText)	String Notes: Returns a string containing a hyperlink to the element specified by the input parameter ElementGUID. Parameters: <ul style="list-style-type: none">• ElementGUID: String - The GUID of the element for which a hyperlink is required• LinkText: String - The text to display for the hyperlink (such as the element name)
GetFileHyperlink (string FilePath, string LinkText)	String Notes: Returns a string containing a hyperlink to the file specified by the input parameter FilePath. Parameters: <ul style="list-style-type: none">• FilePath: String - The path name of the file for which a hyperlink is required• LinkText: String - The text to display for the hyperlink (such as the file name)
GetLastError ()	String Notes: Returns the last error message set for the MailInterface.
GetMethodHyperlink (string MethodGUID, string LinkText)	String Notes: Returns a string containing a hyperlink to the method specified by the input parameter MethodGUID. Parameters: <ul style="list-style-type: none">• MethodGUID: String - The GUID of the method for which a hyperlink is required• LinkText: String - The text to display for the hyperlink (such as the method name)
GetPackageHyperlink (string PackageGUID, string LinkText)	String Notes: Returns a string containing a hyperlink to the Package specified by the input parameter PackageGUID. Parameters: <ul style="list-style-type: none">• PackageGUID: String - The GUID of the Package for which a hyperlink is required• LinkText: String - The text to display for the hyperlink (such as the Package name)
GetRecipientGUID (string UserName)	String Notes: Returns the GUID of the specified Enterprise Architect user. Parameters:

	<ul style="list-style-type: none">• UserName: String - The name of a user defined in the current model
GetWebHyperlink (string URL, string LinkText)	<p>String</p> <p>Notes: Returns a string containing a hyperlink to the URL specified by the input parameter URL.</p> <p>Parameters:</p> <ul style="list-style-type: none">• URL: String - The URL of the item for which a hyperlink is required• LinkText: String - The text to display for the hyperlink
SendMailMessage (string RecipientGUID, string Subject, messageflag Flag, string MessageText)	<p>Boolean</p> <p>Notes: Creates and sends a new mail message using the values specified in the input parameters.</p> <p>Parameters:</p> <ul style="list-style-type: none">• RecipientGUID: String - The GUID of the addressee user (an Enterprise Architect user defined in the current model)• Subject: String - The Subject text to display for this message• Flag: MessageFlag - The flag type/color to attach to this message• MessageText: String - The text that is the body of the message

Simulation Package

The Simulation Package contains:

- An attribute to set, increase and decrease the speed of the simulation
- A function to check if a simulation is currently running
- Functions to Start, Stop, Step Into, Step Out of, Step Over and Pause a simulation
- A function to send a broadcast signal to the simulation that is currently running

Simulation Class

The Simulation Class provides an interface to the Enterprise Architect Model Simulation facilities.

Simulation Attributes

Attribute	Description
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Speed	Long Notes: Read/Write Retrieve or set the current simulation running speed.

Simulation Methods

Method	Description
BroadcastSignal(string sSignalName, string sParameters)	Boolean Notes: Send a signal into the running simulation. If the simulation is stopped, do nothing. Parameters: <ul style="list-style-type: none">sSignalName: String - the name of the signal OR the GUID of the Signal elementsParameters: String - a string of one or more signal parameters, in this format: {parameter1: 5, parameter2: "test", parameter3: 3.2}
IsSimulatorRunning()	Boolean Notes: Check the state of the simulation. Returns True if the simulation is running; returns False if the simulation is stopped.
Pause()	Boolean Notes: Pause the simulation if it is running.
Start()	Boolean Notes: Start the simulation based on the current selection. If the current simulation is in a paused state, then the simulation is resumed.
StepIn()	Boolean Notes: Step In to the routine in the current simulation.
StepOut()	Boolean

	Notes: Step Out of the routine in the current simulation.
StepOver()	Boolean Notes: Step Over the routine in the current simulation.
Stop()	Boolean Notes: Stop the simulation.

Schema Composer Package

The Schema Composer can be accessed from the Enterprise Architect automation interface. A client (script or Add-In) can obtain access to the interface using the SchemaComposer property of the Repository object. This interface is available when a Schema Composer has a profile loaded.

SchemaProperty Class

SchemaProperty Attributes

Attribute	Description
TypeID	long Notes: Read only The classifier ID of the property.
PropID	long Notes: Read only The property ID.
Guid	string Notes: Read only The unique model GUID of the property.
Name	string Notes: Read only The name of the property.
Cardinality	string Notes: Read only The cardinality of the element.
UMLType	string Notes: Read only The UML type, such as attribute, association or aggregation.
Parent	long Notes: Read only The classifier of the owner Class.
PrimitiveType	string Notes: Read only The property's primitive type if property represents a simple type.
Annotation	string Notes: Read only The model notes for the property.
Stereotype	string Notes: Read only The stereotype of the property.

Choices	SchemaTypeEnum Returns an iterator allowing navigation of choice elements defined for this property in the Schema Composer
TypeName	string Returns a string naming the type of the property
Type	SchemaType Returns an interface to the property's type for complex types.

SchemaProperty Methods

Method	Description
IsInline	boolean If true, the property is marked as 'Inline'. XML schema generators would emit an inline definition when detecting this attribute.
IsPrimitive	boolean Returns true for a property whose type is maps to a built in type such as xs:integer, xs:string, xs:date or other XML Schema built-in type.
IsByReference	boolean Returns true for a property marked as 'By Reference' in the profile.

SchemaComposer Class

The SchemaComposer Class provides the interface to the Enterprise Architect Schema Composer facility.

SchemaComposer Attributes

Attribute	Description
ModelReference	String Notes: The model ref listed in the Schema Composer for the current profile.
Namespace	String Notes: The namespace listed in the Schema Composer for the current profile.
NamespacePrefix	String Notes: The namespace prefix listed in the Schema Composer for the current profile.
TargetDirectory	String Notes: The target directory selected by the user after clicking on the Generate button.
SchemaName	String Notes: Returns the name of the schema profile currently being generated.
SchemaSet	String Notes: Returns the schema set used when the schema was created.
SchemaType	String Notes: The schema type listed in the Schema Composer for the current profile, either 'schema' or 'transform'.
SchemaTypes	SchemaTypeEnum Notes: Read only Enumerator for the type collection represented in the currently open schema.

SchemaComposer Methods

Method	Description
FindInSchema(long typeId)	SchemaType Notes: Obtains an interface to a Class as represented in the schema for a given model class ID. Parameters: <ul style="list-style-type: none">typeID: the model Class ID

FindInModel(long typeId)	<p>ModelType</p> <p>Notes: Obtains an interface to a Class as represented in the UML model for a given model Class ID</p> <p>Parameters:</p> <ul style="list-style-type: none">• typeId: the model Class ID
FindSchemaTypeByName(string typename)	<p>SchemaType</p> <p>Notes: Returns an interface to the schema type that matches the type specified or null if no type exists.</p> <p>Parameters:</p> <ul style="list-style-type: none">• name : the name of the type

SchemaProfile Class

The interface representing the technology governing the naming and design rules on which the schema is built.

SchemaProfile Methods

Method	Description
AddExportFormat(string description)	<p>void</p> <p>Notes: Use this function to add entries that are offered by the Schema Composer when the user clicks on the Generate button.</p> <p>Parameters:</p> <ul style="list-style-type: none">description: describes the export format provided by the Add-in
SetCapability(string name,boolean enabled)	<p>void</p> <p>Notes: Use this function to enable/disable capabilities.</p> <p>Parameters:</p> <ul style="list-style-type: none">name: name of the capabilityenabled: True or False <p>Capabilities:</p> <p>'allowCardinality' - allows/denies restrictions to cardinality</p> <p>'allowRootElement' - allows/denies setting root element</p> <p>'allowPropByRef' - allows/denies By Reference restriction</p> <p>'allowRedefine' - allows/denies ability to redefine an element</p>
SetProperty(string name, string value)	<p>void</p> <p>Notes: Sets properties displayed in the Schema Composer.</p> <p>Parameters:</p> <ul style="list-style-type: none">name: property namevalue: property value <p>Properties:</p> <p>'Namespace' - Target namespace for XML schema</p> <p>'Namespace Prefix' - Namespace prefix for XML schema</p> <p>'Qualifier' - string qualifier that prepends schema type names</p>

ModelType Class

Provides an interface to the Class of a schema type as represented in the model.

ModelType Attributes

Attribute	Description
PropertyCount	long Notes: Read only The total number of properties for this Class available in the Properties collection.
Properties	SchemaPropEnum Notes: Enumerator Collection of properties for the Class as defined in the model.
TypeID	long Notes: Read only The Class ID of the type.
Guid	string Notes: Read only A GUID that uniquely identifies a type in the model.
Typename	string Notes: Read only The name of the type as represented in the model.
ClassifierPath	string Notes: Read only The qualified path of the type in the model.
ClassifierPathID	string Notes: Read only A GUID that uniquely identifies a ClassifierPath in the model.
Stereotype	string Notes: Read only The stereotype of the Class as defined in the model.
Annotation	string Notes: Read only Any notes present in the model describing the Class.

ModelType Methods

Method	Description
GetSuperClassEnum(SearchType searchtype)	<p>ModelTypeEnum</p> <p>Notes: Enumerator</p> <p>Returns an enumerator that can be used to traverse the Class ancestry.</p> <p>Parameters:</p> <ul style="list-style-type: none">searchtype: the type of traversal to use, breadth first or depth first
GetSubClassEnum(SearchType searchType)	<p>ModelTypeEnum</p> <p>Notes: Enumerator</p> <p>Returns an enumerator that can be used to iterate over any descendents of the Class.</p> <p>Parameters:</p> <ul style="list-style-type: none">searchtype: the type of traversal to use, breadth first or depth first

ModelTypeEnum Class

An enumerator interface for schema types as represented in the UML model.

ModelTypeEnum Methods

Method	Description
GetCount()	long Returns the number of types present in the collection.
GetFirst()	ModelType Returns the first type interface in a collection of types.
GetNext()	ModelType Returns the next type in the collection of types or null if end is reached.

SchemaType Class

Represents a type as it is defined in the schema.

Methods

Method	Description
GetFacet(BSTR name)	Returns the value of the named facet. 'Root', for example' returns a value indicating whether a type is a root element.
GetRestriction(BSTR guid)	Returns the restriction as a string for the property having the supplied guid.
IsRoot()	True if class is marked as 'root' in the Composer.

Properties

Property	Description
PropertyCount [type: long]	Returns the number of properties held by 'type'.
Properties [type: IEASchemaPropEnum]	Returns an enumerator for 'type's' properties.
TypeID	The model class ID.
Guid	The unique model GUID of the type.
Typename	The type's name.
Parent	The parent type if any that this class extends. May be null depending on composition method.

SchemaTypeEnum Class

An enumerator interface for schema types as represented in XML schema.

Methods

Method	Description
GetCount()	Returns the number of properties for an element.
GetFirst()	Returns the first property for the element in alphabetical order.
GetNext()	Returns the first property for the element in alphabetical order or null if no more are present.

SchemaPropEnum Class

An enumerator for properties of a UML model type or XML schema type.

Methods

Method	Description
GetCount()	Returns the number of properties for an element.
GetFirst()	Returns the first property for the element in alphabetical order.
GetNext()	Returns the first property for the element in alphabetical order or null if no more are present.

SearchType Enumeration

SearchType Attributes

Attribute	Description
searchDepthFirst	Navigate children before siblings.
searchBreadthFirst	Navigate siblings before children.

Code Samples

As you write or edit code for using the Automation Interface, you might want to review these public Object examples, written in VB.Net.

Examples

Name
Open the Repository
Iterate Through a .eap File
Add and Manage Packages
Add and Manage Elements
Add a Connector
Add and Manage Diagrams
Add and Delete Features
Element Extras
Repository Extras
Stereotypes
Work with Attributes
Work with Methods

Open the Repository

This is an example of the VB.Net code to open an Enterprise Architect repository.

```
Public Class AutomationExample
    "Class level variable for Repository
    Public m_Repository As Object

    Public Sub Run()
        try
            "create the repository object
            m_Repository = CreateObject("EA.Repository")

            "open an EAP file
            m_Repository.OpenFile("F:\Test\EAAuto.EAP")

            "use the Repository in any way required
            "DumpModel

            "close the repository and tidy up
            m_Repository.Exit()
            m_Repository = Nothing

            catch e as exception
                Console.WriteLine(e)
            End try
        End Sub
    end Class
```

Iterate Through a .EAP File

This is an example of the VB.Net code to iterate through a .eap file starting at the Model level, after the repository has been opened.

```
Sub DumpModel()  
    Dim idx as Integer  
    For idx=0 to m_Repository.Models.Count-1  
        DumpPackage("",m_Repository.Models.GetAt(idx))  
    Next  
End Sub  
  
"output package name, then element contents, then process child packages  
Sub DumpPackage(Indent as String, Package as Object)  
    Dim idx as Integer  
    Console.WriteLine(Indent + Package.Name)  
    DumpElements(Indent + "  ", Package)  
  
    For idx = 0 to Package.Packages.Count-1  
        DumpPackage(Indent + "  ", Package.Packages.GetAt(idx))  
    Next  
End Sub  
  
"dump element name  
Sub DumpElements(Indent as String, Package as Object)  
    Dim idx as Integer  
    For idx = 0 to Package.Elements.Count-1  
        Console.WriteLine(Indent + "::" + Package.Elements.GetAt(idx).Name)  
    Next  
End Sub
```

Add and Manage Packages

This example illustrates how to add a Model or a Package to the project.

Sub TestPackageLifecycle

Dim idx as integer

Dim idx2 as integer

Dim package as object

Dim model as object

Dim o as object

"first add a new Model

model = m_Repository.Models.AddNew("AdvancedModel", "")

If not model.Update() Then

Console.WriteLine(model.GetLastError())

End If

"refresh the models collection

m_Repository.Models.Refresh

"now work through models collection and add a package

For idx = 0 to m_Repository.Models.Count - 1

o = m_Repository.Models.GetAt(idx)

Console.WriteLine(o.Name)

If o.Name = "AdvancedModel" Then

package = o.Packages.Addnew("Subpackage", "Nothing")

If not package.Update() Then

Console.WriteLine(package.GetLastError())

End If

package.Element.Stereotype = "system"

package.Update

"for testing purposes just delete the

"newly created Model and its contents

"m_Repository.Models.Delete(idx)

End If

Next

End Sub

Add and Manage Elements

This is an example of the code for adding and deleting elements in a Package.

```
Sub ElementLifeCycle
```

```
    Dim package as Object
```

```
    Dim element as Object
```

```
    package = m_Repository.GetPackageByID(2)
```

```
    element = package.elements.AddNew("Login to Website","UseCase")
```

```
    element.Stereotype = "testcase"
```

```
    element.Update
```

```
    package.elements.Refresh()
```

```
    Dim idx as integer
```

```
    "Note the repeated calls to "package.elements.GetAt."
```

```
    "In general you should make this call once and assign to a local
```

```
    "variable - in this example, Enterprise Architect loads the
```

```
    "element required every time a call is made - rather than loading once
```

```
    "and keeping a local reference.
```

```
    For idx = 0 to package.elements.count-1
```

```
        Console.WriteLine(package.elements.GetAt(idx).Name)
```

```
        If (package.elements.GetAt(idx).Name = "Login to Website" and _
```

```
            package.elements.GetAt(idx).Type = "UseCase") Then
```

```
            package.elements.deleteat(idx, false)
```

```
        End If
```

```
    Next
```

```
End Sub
```

Add a Connector

This is an example of code to add a connector and set its values.

Sub ConnectorTest

Dim source as object

Dim target as object

Dim con as object

Dim o as object

Dim client as object

Dim supplier as object

"Use ElementIDs to quickly load an element in this example

"... you must find suitable IDs in your model

source = m_Repository.GetElementByID(129)

target = m_Repository.GetElementByID(169)

con = source.Connectors.AddNew ("test link 2", "Association")

"again, replace ID with a suitable one from your model

con.SupplierID = 169

If not con.Update Then

Console.WriteLine(con.GetLastError)

End If

source.Connectors.Refresh

Console.WriteLine("Connector Created")

o = con.Constraints.AddNew ("constraint2", "type")

If not o.Update Then

Console.WriteLine(o.GetLastError)

End If

o = con.TaggedValues.AddNew ("Tag", "Value")

If not o.Update Then

Console.WriteLine(o.GetLastError)

End If

"Use the client and supplier ends to set
"additional information

```
client = con.ClientEnd
client.Visibility = "Private"
client.Role = "m_client"
client.Update
supplier = con.SupplierEnd
supplier.Visibility = "Protected"
supplier.Role = "m_supplier"
supplier.Update
```

```
Console.WriteLine("Client and Supplier set")
```

```
Console.WriteLine(client.Role)
Console.WriteLine(supplier.Role)
```

End Sub

Add and Manage Diagrams

This is an example of the code for creating a diagram and adding an element to it. Note the optional use of the element rectangle setting using left, right, top and bottom dimensions in the AddNew call.

```
Sub DiagramLifeCycle
```

```
    Dim diagram as object
```

```
    Dim v as object
```

```
    Dim o as object
```

```
    Dim package as object
```

```
    Dim idx as Integer
```

```
    Dim idx2 as integer
```

```
    package = m_Repository.GetPackageByID(5)
```

```
    diagram = package.Diagrams.AddNew("Logical Diagram","Logical")
```

```
    If not diagram.Update Then
```

```
        Console.WriteLine(diagram.GetLastError)
```

```
    End if
```

```
    diagram.Notes = "Hello there this is a test"
```

```
    diagram.update()
```

```
    o = package.Elements.AddNew("ReferenceType","Class")
```

```
    o.Update
```

```
    " add element to diagram - supply optional rectangle co-ordinates
```

```
    v = diagram.DiagramObjects.AddNew("l=200;r=400;t=200;b=600;", "")
```

```
    v.ElementID = o.ElementID
```

```
    v.Update
```

```
    Console.WriteLine(diagram.DiagramID)
```

```
End Sub
```

Add and Delete Features

An example of code to add and delete Features of an object.

```
Dim element as object
```

```
Dim idx as integer
```

```
Dim attribute as object
```

```
Dim method as object
```

```
'just load an element by ID - you must
```

```
'substitute a valid ID from your model
```

```
element = m_Repository.GetElementByID(246)
```

```
"create a new method
```

```
method = element.Methods.AddNew("newMethod", "int")
```

```
method.Update
```

```
element.Methods.Refresh
```

```
'now loop through methods for Element - and delete our addition
```

```
For idx = 0 to element.Methods.Count-1
```

```
    method =element.Methods.GetAt(idx)
```

```
    Console.WriteLine(method.Name)
```

```
    If(method.Name = "newMethod") Then
```

```
        element.Methods.Delete(idx)
```

```
    End if
```

```
Next
```

```
'create an attribute
```

```
attribute = element.attributes.AddNew("NewAttribute", "int")
```

```
attribute.Update
```

```
element.attributes.Refresh
```

```
'loop through and delete our new attribute
```

```
For idx = 0 to element.attributes.Count-1
```

```
    attribute =element.attributes.GetAt(idx)
```

```
    Console.WriteLine(attribute.Name)
```

```
    If(attribute.Name = "NewAttribute") Then
```

```
        element.attributes.Delete(idx)
```

```
    End If
```

```
Next
```

Element Extras

These are examples of code to access and use element extras, such as scenarios, constraints and requirements.

Sub ElementExtras

```
Dim element as object
```

```
Dim o as object
```

```
Dim idx as Integer
```

```
Dim bDel as boolean
```

```
bDel = true
```

```
try
```

```
    element = m_Repository.GetElementByID(129)
```

```
    'manage constraints for an element
```

```
    'demonstrate addnew and delete
```

```
    o = element.Constraints.AddNew("Appended","Type")
```

```
    If not o.Update Then
```

```
        Console.WriteLine("Constraint error:" + o.GetLastError())
```

```
    End if
```

```
    element.Constraints.Refresh
```

```
    For idx = 0 to element.Constraints.Count -1
```

```
        o = element.Constraints.GetAt(idx)
```

```
        Console.WriteLine(o.Name)
```

```
        If(o.Name="Appended") Then
```

```
            If bDel Then element.Constraints.Delete (idx)
```

```
        End if
```

```
    Next
```

```
    'efforts
```

```
    o = element.Efforts.AddNew("Appended","Type")
```

```
    If not o.Update Then
```

```
        Console.WriteLine("Efforts error:" + o.GetLastError())
```

```
    End if
```

```
    element.Efforts.Refresh
```

```
    For idx = 0 to element.Efforts.Count -1
```

```
        o = element.Efforts.GetAt(idx)
```

```
        Console.WriteLine(o.Name)
```

```
        If(o.Name="Appended") Then
```

```
            If bDel Then element.Efforts.Delete (idx)
```

```
End if
Next

'Risks
o = element.Risks.AddNew("Appended", "Type")
If not o.Update Then
    Console.WriteLine("Risks error:" + o.GetLastError())
End if
element.Risks.Refresh
For idx = 0 to element.Risks.Count - 1
    o = element.Risks.GetAt(idx)
    Console.WriteLine(o.Name)
    If(o.Name="Appended") Then
        If bDel Then element.Risks.Delete (idx)
    End if
Next

'Metrics
o = element.Metrics.AddNew("Appended", "Change")
If not o.Update Then
    Console.WriteLine("Metrics error:" + o.GetLastError())
End if
element.Metrics.Refresh
For idx = 0 to element.Metrics.Count - 1
    o = element.Metrics.GetAt(idx)
    Console.WriteLine(o.Name)
    If(o.Name="Appended") Then
        If bDel Then element.Metrics.Delete (idx)
    End if
Next

'TaggedValues
o = element.TaggedValues.AddNew("Appended", "Change")
If not o.Update Then
    Console.WriteLine("TaggedValues error:" + o.GetLastError())
End if
element.TaggedValues.Refresh
For idx = 0 to element.TaggedValues.Count - 1
    o = element.TaggedValues.GetAt(idx)
    Console.WriteLine(o.Name)
    If(o.Name="Appended") Then
        If bDel Then element.TaggedValues.Delete (idx)
```

```
End if
Next

'Scenarios
o = element.Scenarios.AddNew("Appended","Change")
If not o.Update Then
    Console.WriteLine("Scenarios error:" + o.GetLastError())
End if
element.Scenarios.Refresh
For idx = 0 to element.Scenarios.Count -1
    o = element.Scenarios.GetAt(idx)
    Console.WriteLine(o.Name)
    If(o.Name="Appended") Then
        If bDel Then element.Scenarios.Delete (idx)
    End if
Next

'Files
o = element.Files.AddNew("MyFile","doc")
If not o.Update Then
    Console.WriteLine("Files error:" + o.GetLastError())
End if
element.Files.Refresh
For idx = 0 to element.Files.Count -1
    o = element.Files.GetAt(idx)
    Console.WriteLine(o.Name)
    If(o.Name="MyFile") Then
        If bDel Then element.Files.Delete (idx)
    End if
Next

'Tests
o = element.Tests.AddNew("TestPlan","Load")
If not o.Update Then
    Console.WriteLine("Tests error:" + o.GetLastError())
End if
element.Tests.Refresh
For idx = 0 to element.Tests.Count -1
    o = element.Tests.GetAt(idx)
    Console.WriteLine(o.Name)
    If(o.Name="TestPlan") Then
        If bDel Then element.Tests.Delete (idx)
```

```
End if
Next

'Defect
o = element.Issues.AddNew("Broken","Defect")
If not o.Update Then
    Console.WriteLine("Issues error:" + o.GetLastError())
End if
element.Issues.Refresh
For idx = 0 to element.Issues.Count -1
    o = element.Issues.GetAt(idx)
    Console.WriteLine(o.Name)
    If(o.Name="Broken") Then
        If bDel Then element.Issues.Delete (idx)
    End if
Next
```

```
'Change
o = element.Issues.AddNew("Change","Change")
If not o.Update Then
    Console.WriteLine("Issues error:" + o.GetLastError())
End if
element.Issues.Refresh
For idx = 0 to element.Issues.Count -1
    o = element.Issues.GetAt(idx)
    Console.WriteLine(o.Name)
    If(o.Name="Change") Then
        If bDel Then element.Issues.Delete (idx)
    End if
Next
```

```
catch e as exception
    Console.WriteLine(element.Methods.GetLastError())
    Console.WriteLine(e)
End try
```

```
End Sub
```

Repository Extras

These are examples of code for accessing repository collections for system-level information.

Sub RepositoryExtras

```
Dim o as object
Dim idx as integer

'issues
o = m_Repository.Issues.AddNew("Problem","Type")
If(o.Update=false) Then
    Console.WriteLine (o.GetLastError())
End if
o = nothing
m_Repository.Issues.Refresh
For idx = 0 to m_Repository.Issues.Count-1
    Console.WriteLine(m_Repository.Issues.GetAt(idx).Name)
    If(m_Repository.Issues.GetAt(idx).Name = "Problem") then
        m_Repository.Issues.DeleteAt(idx,false)
        Console.WriteLine("Delete Issues")
    End if
Next

'tasks
o = m_Repository.Tasks.AddNew("Task 1","Task type")
If(o.Update=false) Then
    Console.WriteLine ("error - " + o.GetLastError())
End if
o = nothing
m_Repository.Tasks.Refresh
For idx = 0 to m_Repository.Tasks.Count-1
    Console.WriteLine(m_Repository.Tasks.GetAt(idx).Name)
    If(m_Repository.Tasks.GetAt(idx).Name = "Task 1") then
        m_Repository.Tasks.DeleteAt(idx,false)
        Console.WriteLine("Delete Tasks")
    End if
Next

'glossary
o = m_Repository.Terms.AddNew("Term 1","business")
```

```
If(o.Update=false) Then
    Console.WriteLine ("error - " + o.GetLastError())
End if
o = nothing
m_Repository.Terms.Refresh
For idx = 0 to m_Repository.Terms.Count-1
    Console.WriteLine(m_Repository.Terms.GetAt(idx).Term)
    If(m_Repository.Terms.GetAt(idx).Term = "Term 1") then
        m_Repository.Terms.DeleteAt(idx,false)
        Console.WriteLine("Delete Terms")
    End if
Next

'authors
o = m_Repository.Authors.AddNew("Joe B","Writer")
If(o.Update=false) Then
    Console.WriteLine (o.GetLastError())
End if
o = nothing
m_Repository.Authors.Refresh
For idx = 0 to m_Repository.authors.Count-1
    Console.WriteLine(m_Repository.Authors.GetAt(idx).Name)
    If(m_Repository.authors.GetAt(idx).Name = "Joe B") then
        m_Repository.authors.DeleteAt(idx,false)
        Console.WriteLine("Delete Authors")
    End if
Next

o = m_Repository.Clients.AddNew("Joe Sphere","Client")
If(o.Update=false) Then
    Console.WriteLine (o.GetLastError())
End if
o = nothing
m_Repository.Clients.Refresh
For idx = 0 to m_Repository.Clients.Count-1
    Console.WriteLine(m_Repository.Clients.GetAt(idx).Name)
    If(m_Repository.Clients.GetAt(idx).Name = "Joe Sphere") then
        m_Repository.Clients.DeleteAt(idx,false)
        Console.WriteLine("Delete Clients")
    End if
Next
```

```
o = m_Repository.Resources.AddNew("Joe Worker","Resource")
If(o.Update=false) Then
    Console.WriteLine (o.GetLastError())
End if
o = nothing
m_Repository.Resources.Refresh
For idx = 0 to m_Repository.Resources.Count-1
    Console.WriteLine(m_Repository.Resources.GetAt(idx).Name)
    If(m_Repository.Resources.GetAt(idx).Name = "Joe Worker") then
        m_Repository.Resources.DeleteAt(idx,false)
        Console.WriteLine("Delete Resources")
    End if
Next

End Sub
```

Stereotypes

This is some example code for adding and deleting stereotypes.

Sub TestStereotypes

Dim o as object

Dim idx as integer

"add a new stereotype to the Stereotypes collection

o = m_Repository.Stereotypes.AddNew("funky","class")

If(o.Update=false) Then

 Console.WriteLine (o.GetLastError())

End if

o = nothing

"make sure you refresh

m_Repository.Stereotypes.Refresh

"then iterate through - deleting our new entry in the process

For idx = 0 to m_Repository.Stereotypes.Count-1

 Console.WriteLine(m_Repository.Stereotypes.GetAt(idx).Name)

 If(m_Repository.Stereotypes.GetAt(idx).Name = "funky") then

 m_Repository.Stereotypes.DeleteAt(idx,false)

 Console.WriteLine("Delete element")

 End if

Next

End Sub

Work With Attributes

This is an example of code for working with attributes.

Sub AttributeLifecycle

Dim element as object

Dim o as object

Dim t as object

Dim idx as Integer

Dim idx2 as integer

try

element = m_Repository.GetElementByID(129)

For idx = 0 to element.Attributes.Count -1

Console.WriteLine("attribute=" + element.Attributes.GetAt(idx).Name)

o = element.Attributes.GetAt(idx)

t = o.Constraints.AddNew("> 123", "Precision")

t.Update()

o.Constraints.Refresh

For idx2 = 0 to o.Constraints.Count-1

t = o.Constraints.GetAt(idx2)

Console.WriteLine("Constraint: " + t.Name)

If(t.Name="> 123") Then

o.Constraints.DeleteAt(idx2, false)

End if

Next

For idx2 = 0 to o.TaggedValues.Count-1

t = o.TaggedValues.GetAt(idx2)

If(t.Name = "Type2") Then

'Console.WriteLine("deleteing")

o.TaggedValues.DeleteAt(idx2, true)

End if

Next

t = o.TaggedValues.AddNew("Type2", "Number")

t.Update

o.TaggedValues.Refresh

```
For idx2 = 0 to o.TaggedValues.Count-1
    t = o.TaggedValues.GetAt(idx2)
    Console.WriteLine("Tagged Value: " + t.Name)
Next
```

```
If(element.Attributes.GetAt(idx).Name = "m_Tootle") Then
    Console.WriteLine("delete attribute")
    element.Attributes.DeleteAt(idx, false)
End If
```

```
Next
```

```
catch e as exception
    Console.WriteLine(element.Attributes.GetLastError())
    Console.WriteLine(e)
```

```
End try
```

```
End Sub
```

Work With Methods

This is an example of code for working with the Methods collection of an element and with Method collections.

Sub MethodLifeCycle

```
Dim element as object
```

```
Dim method as object
```

```
Dim t as object
```

```
Dim idx as Integer
```

```
Dim idx2 as integer
```

```
try
```

```
    element = m_Repository.GetElementByID(129)
```

```
    For idx = 0 to element.Methods.Count - 1
```

```
        method = element.Methods.GetAt(idx)
```

```
        Console.WriteLine(method.Name)
```

```
        t = method.PreConditions.AddNew("TestConstraint","something")
```

```
        If t.Update = false Then
```

```
            Console.WriteLine("PreConditions: " + t.GetLastError)
```

```
        End if
```

```
        method.PreConditions.Refresh
```

```
        For idx2 = 0 to method.PreConditions.Count-1
```

```
            t = method.PreConditions.GetAt(idx2)
```

```
            Console.WriteLine("PreConditions: " + t.Name)
```

```
            If t.Name = "TestConstraint" Then
```

```
                method.PreConditions.DeleteAt(idx2,false)
```

```
            End If
```

```
        Next
```

```
        t = method.PostConditions.AddNew("TestConstraint","something")
```

```
        If t.Update = false Then
```

```
            Console.WriteLine("PostConditions: " + t.GetLastError)
```

```
        End if
```

```
        method.PostConditions.Refresh
```

```
        For idx2 = 0 to method.PostConditions.Count-1
```

```
            t = method.PostConditions.GetAt(idx2)
```

```
        Console.WriteLine("PostConditions: " + t.Name)
        If t.Name = "TestConstraint" Then
            method.PostConditions.DeleteAt(idx2, false)
        End If
    Next

    t = method.TaggedValues.AddNew("TestTaggedValue","something")
    If t.Update = false Then
        Console.WriteLine("Tagged Values: " + t.GetLastError)
    End if

    For idx2 = 0 to method.TaggedValues.Count-1
        t = method.TaggedValues.GetAt(idx2)
        Console.WriteLine("Tagged Value: " + t.Name)
        If(t.Name= "TestTaggedValue") Then
            method.TaggedValues.DeleteAt(idx2,false)
        End If
    Next

    t = method.Parameters.AddNew("TestParam","string")
    If t.Update = false Then
        Console.WriteLine("Parameters: " + t.GetLastError)
    End if

    method.Parameters.Refresh
    For idx2 = 0 to method.Parameters.Count-1
        t = method.Parameters.GetAt(idx2)
        Console.WriteLine("Parameter: " + t.Name)
        If(t.Name="TestParam") Then
            method.Parameters.DeleteAt(idx2, false)
        End If
    Next

    method = nothing
Next
catch e as exception
    Console.WriteLine(element.Methods.GetLastError())
    Console.WriteLine(e)
End try

End Sub
```

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	Extend > Configure > Manage Add-Ins
--------	-------------------------------------

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' check box 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 the 'Extensions <add-in name>' context menu optionAny defined Add-In windows are populated with information; select 'Extend > Configure > Add-In Windows'
Disable an Add-In	<p>To disable an Add-In so that it is not available for use, clear the 'Load on Startup' check box 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 provides instructions on how to create, test, deploy and manage Add-Ins.

Create an Add-In

Task
Create an Add-In.
Define Menu Items.
Respond to Menu Events.
Handle Add-In Events.

Deploy your Add-In

Task
Potential Pitfalls.

Manage Add-Ins

Task
Register an Add-In (developed in-house or brought-in).
The Add-In Manager.

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:

- 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

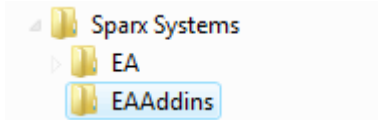
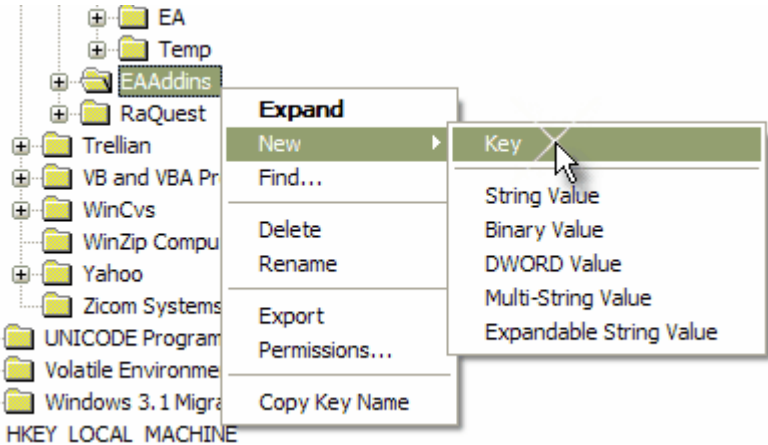
Tasks

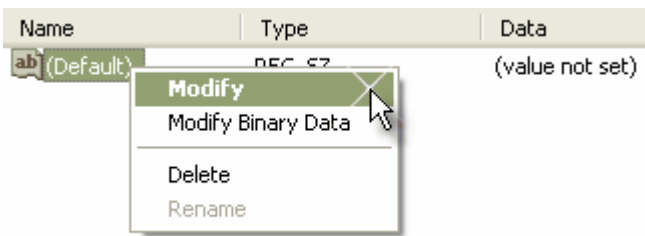
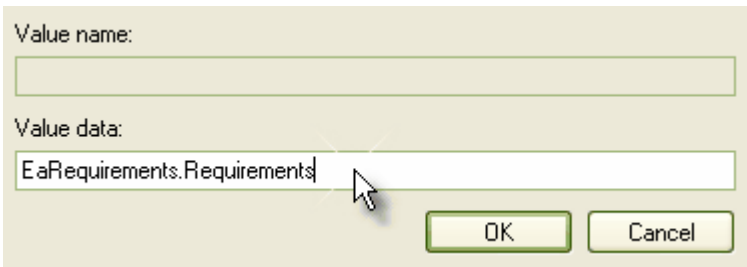
Task	Detail
Define Menu Items	<p>Menu items are defined by responding to the GetMenuItems event.</p> <p>The first time this event is called, MenuName is an empty string, representing the top-level menu. For a simple Add-In with just a single menu option you can return a string.</p> <pre> Function EA_GetMenuItems(Repository 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 like this:</p> <pre> Function EA_GetMenuItems(Repository as EA.Repository, MenuLocation As String, MenuName As String) As Variant 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 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 </pre>

	End Function
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" 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	Add the Add-In DLL file to an appropriate directory on the user's computer; that is: C:\Program Files\new dir
2	Register the DLL as appropriate to your platform: <ul style="list-style-type: none"> • If compiled as a native Win32 DDL, such as VB6 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\EAAddin\EAAddin.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	Add a new key 'EAAddIns' under one of these locations: <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. 

	<p>(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.</p>
6	<p>Specify the default value by modifying the default value of the key.</p> 
7	<p>Enter the value of the key by typing in the (project name).(class name), such as: EaRequirements.Requirements where <i>EaRequirements</i> is the project name, as shown in this example:</p> 

Tricks and Traps

Considerations

Item	Detail
Visual Basic 5/6 Users Note	<p>Visual Basic 5/6 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> <pre>Reference=*\G{64FB2BF4-9EFA-11D2-8307-C45586000000}#2.2#0#...\Program Files\Sparx Systems\EA\EA.TLB#Enterprise Architect Object Model 2.02</pre> <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>
Add-In Fails to Load	<p>From Enterprise Architect release 7.0, 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.</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 one event and retrieved in another. There are some dangers in doing this:</p> <ul style="list-style-type: none"> 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) <p>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.</p>
Enterprise Architect Not Closing	<p>.NET Specific Issues</p> <p>Automation checks the use of objects and won't allow any of them to be destroyed until they are no longer being used.</p> <p>As noted in the Automation Interface 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 which in turn hook back into Enterprise Architect, it is possible to get into a deadlock situation where Enterprise Architect and the Add-Ins won't let go of one another and keep each other active. An Add-In might retain hooks into</p>

	<p>Enterprise Architect because:</p> <ul style="list-style-type: none"> • It keeps a private reference to an Enterprise Architect object (see <i>Holding State Information</i> above), or • It has been created by .NET and the GC mechanism hasn't got around to releasing it <p>There are two actions required to avoid deadlock situations:</p> <ul style="list-style-type: none"> • Automation controllers must call Repository.CloseAddins() at some point (presumably at the end of processing) • Add-Ins must release all references to Enterprise Architect in the Disconnect() event; see the <i>Add-In Events</i> topic for details <p>It is possible that your Automation client controls a running instance of Enterprise Architect where the Add-Ins have not complied with the rule above. In this case you could call Repository.Exit() to terminate Enterprise Architect.</p> <p>Miscellaneous</p> <p>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.</p> <p>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> <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>
Concurrent Calls	<p>In Enterprise Architect releases up to release 7.0, there is a possibility that Enterprise Architect could call two Add-In methods concurrently if the Add-In calls:</p> <ul style="list-style-type: none"> • A message box • A modal dialog • VB DoEvents, .NET Application DoEvents or the equivalent in other languages <p>In such cases, Enterprise Architect could initiate a second Add-In method before the first returns (re-entrancy). In release 7.0. and subsequent releases, Enterprise Architect cannot make such concurrent calls.</p> <p>If developing Add-Ins, ensure that the Add-In users are running Enterprise Architect release 7.0 or a later release to avoid any risk of concurrent method calls.</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.

The method that runs the search must be structured like this:

Function <method name> (ByVal Rep As Repository, ByVal SearchText As String, ByRef XMLResults As String) As Variant

- Rep - EA.Repository - IN - The current open repository
- SearchText - String - IN - An optional field that you can fill in through the Model Search
- XMLResults - String - OUT - At completion of the method, this should contain the results for the search; the results should be an XML string that conforms to the Search Data Format

Return Value

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

Advanced Usage

In addition to the displayed results, two additional hidden fields can be passed into the XML that provide special functionality.

- 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 Project Browser and open the Properties window for the element by double-clicking in the Model Search

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.
<i>EA_MenuClick</i> - received by an Add-In in response to user selection of a menu option.
<i>EA_OnOutputItemClicked</i> - informs Add-Ins that the user has clicked on a list entry in the system tab or one of the user defined output tabs.
<i>EA_OnOutputItemDoubleClicked</i> - 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.
<i>EA_ShowHelp</i> - Add-Ins can use this to show a Help topic for a particular menu option.

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.
""	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 particularly 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
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.
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

One of these types:

- A string indicating the label for a single menu option
- An array of strings indicating a 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
IsChecked	Boolean Direction: OUT Description: Set to True to check this particular menu option.
IsEnabled	Boolean Direction: OUT Description: Set to False to disable this particular menu option.
ItemName	String Direction: IN Description: The name of the option actually clicked; for example, 'Create a New Invoice'.
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 must be defined. In the case of the top-level menu it is an empty string.
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_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
- Configure
- Construct
- Code
- Simulate
- Execute
- Extend

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 Extend 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
ItemName	String Direction: IN Description: The name of the option actually clicked; for example, 'Create a New Invoice'.
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.
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_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
ID	Long Direction: IN Description: The ID value specified in the original call to Repository.WriteOutput().
LineText	String Direction: IN Description: The text that had been supplied as the String parameter in the original call to 'Repository.WriteOutput()'.
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()'.

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
ID	Long Direction: IN Description: The ID value specified in the original call to Repository.WriteOutput().
LineText	String Direction: IN Description: The text that had been supplied as the String parameter in the original call to 'Repository.WriteOutput()'.
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()'.

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

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 Licence Management Events
Compartment Events
Context Item Events
File Close Event
File New Event
File Open Event
Model Validation Broadcasts
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 Broadcasts
Tagged Value Broadcasts
Technology Events

Transformation Event

Schema Composer Broadcasts

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: 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.
profile	Type: EA.SchemaProfile Direction: IN Description: An EA.SchemaProfile 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.

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
AddinKey	String Direction: IN Description: The Add-in license key that has been entered in the 'License Management' dialog.
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 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
AddinKey	String Direction: IN Description: The Add-In license key that Enterprise Architect requires a description for.
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 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=

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

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
GUID	String Direction: IN Description: Contains the GUID of the new context item. The value corresponds to these properties, depending on the value of the <code>ot</code> parameter: <ul style="list-style-type: none">• <code>ot (ObjectType)</code> - GUID value• <code>otElement</code> - <code>Element.ElementGUID</code>• <code>otPackage</code> - <code>Package.PackageGUID</code>• <code>otDiagram</code> - <code>Diagram.DiagramGUID</code>• <code>otAttribute</code> - <code>Attribute.AttributeGUID</code>• <code>otMethod</code> - <code>Method.MethodGUID</code>• <code>otConnector</code> - <code>Connector.ConnectorGUID</code>• <code>otRepository</code> - NOT APPLICABLE, the GUID is an empty string
ot	EA.ObjectType Direction: IN Description: Specifies the type of the new context item.
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_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 Project Browser 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
GUID	String Direction: IN Description: Contains the GUID of the new context item. The value corresponds to these properties, depending on the value of the ot parameter: <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	EA.ObjectType Direction: IN Description: Specifies the type of the new context item.
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 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
GUID	String Direction: IN Description: Contains the GUID of the new context item. The value corresponds to these properties, depending on the value of the ot parameter: <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	EA.ObjectType Direction: IN Description: Specifies the type of the new context item.
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_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
DiagramID	Integer Direction: IN Description: Contains the Diagram ID of the diagram that was closed.
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_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
DiagramID	Integer Direction: IN Description: Contains the Diagram ID of the diagram that was opened.
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_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
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty Objects for the transform performed: <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
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

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
DiagramID	Long Direction: IN Description: The diagram ID, or 0 if switched to an Add-In tab.
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.

Return Value

None

Model Validation Broadcasts

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

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
Args	ParamArray of Variant Direction: IN Description: Contains a list of Rule Categories that are active for the current invocation of model validation.
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_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
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.
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_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
Element	EA.Element Direction: IN Description: The element to potentially perform validation on.
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.

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
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.
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.

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
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.
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.

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
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.
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.

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
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.
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.

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
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.
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.

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
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.
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.
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.

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 IDiagramID)
{
    theRules.RunDiagramRule(Repository, RuleID, IDiagramID);
}
```

```

public void EA_OnRunConnectorRule(EA.Repository Repository, string RuleID, long IConnectorID)
{
    theRules.RunConnectorRule(Repository, RuleID, IConnectorID);
}
public void EA_OnRunAttributeRule(EA.Repository Repository, string RuleID, string AttGUID, long IObjectID)
{
    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.EnumMVErrType.mvError, GetRuleStr(cRule01)),
cRule01);
AddToMap(Project.DefineRule(m_sCategoryID, EA.EnumMVErrType.mvError, GetRuleStr(cRule02)),
cRule02);
AddToMap(Project.DefineRule(m_sCategoryID, EA.EnumMVErrType.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 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
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the new element: <ul style="list-style-type: none">• ElementID: A long value corresponding to Element.ElementID
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 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 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
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the new connector: <ul style="list-style-type: none">ConnectorID: A long value corresponding to Connector.ConnectorID
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 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
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
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 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 from the Project Browser or Resources 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
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the new element: <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
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 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 Project Browser, using the 'Attributes' tab of the 'Features' dialog, 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
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the new attribute: <ul style="list-style-type: none">AttributeID: A long value corresponding to Attribute.AttributeID
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 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 Project Browser, 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
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the new method: <ul style="list-style-type: none">MethodID: A long value corresponding to Method.MethodID
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 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 window onto a diagram, or by selecting the New Package icon from the Project Browser.

Syntax

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

The EA_OnPostNewPackage function syntax contains these parameters.

Parameter	Type
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the new Package: <ul style="list-style-type: none">PackageID: A long value corresponding to Package.PackageID
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 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
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the new glossary term: <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
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 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 Project Browser 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
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the element to be deleted: <ul style="list-style-type: none">• ElementID: A long value corresponding to Element.ElementID
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 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 Project Browser. 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
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the attribute to be deleted: <ul style="list-style-type: none">AttributeID: A long value corresponding to Attribute.AttributeID
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 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 Project Browser. 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
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the method to be deleted: <ul style="list-style-type: none">MethodID: A long value corresponding to Method.MethodID
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 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
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the connector to be deleted: <ul style="list-style-type: none">ConnectorID: A long value corresponding to Connector.ConnectorID
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 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 Project Browser. 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
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the diagram to be deleted: <ul style="list-style-type: none">DiagramID: A long value corresponding to Diagram.DiagramID
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 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
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the element to be deleted: <ul style="list-style-type: none">ID: A long value corresponding to DiagramObject.ElementID
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 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 Project Browser. 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
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the Package to be deleted: <ul style="list-style-type: none">PackageID: A long value corresponding to Package.PackageID
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 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
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the glossary term to be deleted: <ul style="list-style-type: none">TermID: A long value corresponding to Term.TermID
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 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 Project Browser.
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 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
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the element to be created: <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
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 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 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
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the connector to be created: <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
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 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
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the diagram to be created: <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
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 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 from the Enterprise Architect Project Browser or Resources 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
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the object to be created: <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
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 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 Project Browser 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
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the element to be created: <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
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 True to allow the default behavior to be executed. Return False if you are overriding 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 Project Browser, using the 'Attributes' tab of the 'Features' dialog, 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
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the attribute to be created: <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
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 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 Project Browser, using the 'Operations' tab of the 'Features' dialog, 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
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the method to be created: <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
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 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 window onto a diagram, or by selecting the New Package icon from the Project Browser. 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
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the Package to be created: <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
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 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
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the glossary term to be created: <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
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 True to enable addition of the new glossary term to the model. Return False to disable addition of the new glossary term.

Tagged Value Broadcasts

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 Broadcast 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
AttributeID	Long Direction: IN Description: The ID of the attribute that this Tagged Value is on.
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.
TagName	String Direction: IN Description: The name of the Tagged Value to edit.
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.
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.

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
ConnectorID	Long Direction: IN Description: The ID of the connector that this Tagged Value is on.
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.
TagName	String Direction: IN Description: The name of the Tagged Value to edit.
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.
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.

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
ObjectID	Long Direction: IN Description: The ID of the object (element) that this Tagged Value is on.
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.
TagName	String Direction: IN Description: The name of the Tagged Value to edit.
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.
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.

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
MethodID	Long Direction: IN Description: The ID of the method that this Tagged Value is on.
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.
TagName	String Direction: IN Description: The name of the Tagged Value to edit.
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.
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.

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

Deprecated - refers to importing a technology into the Resources 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
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
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.

Custom Views

Enterprise Architect enables custom windows to be inserted as a Diagram 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

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();
    }
}
```

Add a Portal

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.

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.

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 'Extend > Configure > Add-In Windows'.

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-Ins that have additional features and extra requirements, for Add-In authors who want to contribute to Enterprise Architect's goal of Model Driven Generation. Two examples of MDG Add-Ins are the MDG Link for Eclipse and MDG Link for Visual Studio, both integrated with the Enterprise Architect installer.

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_Build Project

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 > Run > 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
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.
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.

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
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.
PackageID	Long Direction: IN Description: The PackageID of the Enterprise Architect Package the user has requested to have connected to an external project.
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 a non-zero to indicate that a connection has been made; a zero indicates 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
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.
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 a non-zero to indicate that a disconnection has occurred enabling Enterprise Architect to update the user interface. A zero indicates 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
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 above for the possible values.
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

See above.

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
ExportFiles	Variant Direction: OUT Description: A string array containing the list of files for each model object chosen for export by the Add-In. 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).
ExportObjects	Variant Direction: OUT 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.
IgnoreLocked	String Direction: OUT Description: A value indicating whether to ignore any files locked by the code project (that is, 'True' or 'False').
ImportFiles	Variant Direction: OUT Description: A string array containing the list of code files made available to the code project to be newly imported to the model. Enterprise Architect imports each file listed in this array for import into the connected model branch.
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.
PackageGuid	String

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

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 and a zero value 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
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.
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.
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 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
FileContents	String Direction: IN Description: A string containing the proposed contents of the file.
FilePath	String Direction: IN Description: The path of the file Enterprise Architect intends to overwrite.
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.
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

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
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.
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 a zero value if the post-merge process has failed. A non-zero return indicates that the post-merge has been successful. 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 value to abort generation. Any other value enables 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
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.
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.

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
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.
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 return value of zero indicates that the merge process can not occur. If the value is not zero 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
FilePaths	String array Direction: IN Description: An array of filepaths pointed to the files that are to be reverse engineered.
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.
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.

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.

Return Value

None.

Syntax:

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

The MDG_RunExe function syntax contains these parameters.

Parameter	Type
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.
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.

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
CodeID	String Direction: IN Description: Identifies the code element in this format: <type>ElementPart<type>ElementPart... where each element is proceeded with a token identifying its type: @ - namespace # - Class \$ - attribute % - operation 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: @Name1#Class1%m_Name
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.
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 a non-zero value to indicate that the Add-In has processed the request. Returning a zero value results in Enterprise Architect employing the standard viewing process which is to launch the associated source file.

Workflow Scripts

Workflow scripts validate user work and actions against the policy and procedures within your model, providing a robust approach to applying company policy and strengthening project development guidelines.

Project Administrators can write workflow scripts to manage the way users interact with a model, such as managing security, staff compliance and model access, and monitoring changes made by users. Administrators can also use the scripts to control a user's capacity to change a model element, taking into account factors such as access rights, group membership and even the value of a proposed change.

Access

Open the Scripting window using one of the methods outlined here, then click on the New Group button to create a new Workflow script group, before clicking on the New Script button to create a new script.

Ribbon	Code > Tools > Scripting
--------	--------------------------

Application of Workflow scripts

Consideration	Description
Project Governance	<p>Good corporate governance relies on well written and transparent project development guidelines and company policy.</p> <p>A project might be compromised if the appropriate policies and procedures are poorly understood and not followed correctly - effective governance can be hampered by human error and the costs of recovering from the inadequate compliance of developers.</p>
Policies, Procedures and Development	<p>Company policy and procedures can be integrated with the development process to manage workflows, determine access rights, extend role based security permissions and respond to property change events.</p> <p>This approach reduces compliance costs, enhances collaborative development and gives you confidence that projects are being developed correctly the first time around.</p> <p>Development teams can adhere to best practice guidelines that govern model validation, change management, access controls and general development principles.</p>
Script Implementation	<p>When a model is launched, the Workflow Engine is initialized with the current user and group memberships; this information determines who can access and modify parts of a given model.</p> <p>When a selected event occurs, the script engine is initialized with values including the author's name and access rights, and the element name and version details.</p> <p>The workflow script implements rules governing change management, access control and model validation; if a user attempts to make changes in violation of company policy, the script denies the update.</p> <p>The user is notified why the validation failed and the activity is logged.</p> <p>These reminders help to reinforce company policy, reduce human error and provide management with valuable project feedback.</p>

Notes

- Workflow Scripting is available in the Corporate, Business and Software Engineering, System Engineering and Ultimate editions of Enterprise Architect
- Workflow Scripting requires User Security to be enabled in order to function
- You need Admin Workflow permission to develop and manage workflow scripts

Workflow Script Functions

Workflow scripts are created in the Scripting window, under the Workflow group type as VBScripts. They are executed by the Enterprise Architect workflow engine, to manage user input.

You can make use of a range of functions and data structures to develop your scripts.

Access

Ribbon	Code > Tools > Scripting
--------	--------------------------

Workflow functions and data structures

Function	Description
Functions for User Input	These are functions that Enterprise Architect calls to validate and control user input. For each of the functions that Enterprise Architect calls, a set of objects are filled.
Functions to create a Search	These are functions that Enterprise Architect calls to create a search with user tasks.
Workflow Data Structures Enterprise Architect fills	These are workflow data structure objects that Enterprise Architect fills.
Workflow Data Structures you fill	These are Workflow data structure objects that you can fill.
Functions you call	These are functions that Enterprise Architect provides for you to call.

Notes

- If you make changes to a workflow script listed in the Scripting window, click on the Refresh Scripts button in the Scripting window toolbar to reload the script with the changes

Functions - Validate and Control User Input

Enterprise Architect calls a number of functions to validate and control user input. For each function a set of objects is filled.

Validate/Control User Input

Function	Action
AllowPhaseUpdate(OldValue, NewValue)	Validate a change a user has made to a phase. Return Value: <ul style="list-style-type: none"> • True to allow this user to make this change • False to disallow the change and revert to the previous value
AllowStatusUpdate(OldValue, NewValue)	Validate a change a user has made to a status. Return Value: <ul style="list-style-type: none"> • True to allow this user to make this change • False to disallow the change and revert to the previous value
AllowTagUpdate(TagName, OldValue, NewValue)	Validate a change a user has made to a Tagged Value. Return Value: <ul style="list-style-type: none"> • True to allow this user to make this change • False to disallow the change and revert to the previous value
AllowVersionUpdate(OldValue, NewValue)	Validate a change a user has made to a version. Return Value: <ul style="list-style-type: none"> • True to allow this user to make this change • False to disallow the change and revert to the previous value
CanEditPhase()	Enable or disable the control for editing a phase Return Value: <ul style="list-style-type: none"> • True to allow this user to make changes by enabling the control • False to completely disable edit of this property by disabling the control
CanEditStatus()	Enable or disable the control for editing a status. Return Value: <ul style="list-style-type: none"> • True to allow this user to make changes by enabling the control • False to completely disable edit of this property by disabling the control
CanEditTag(TagName)	Enable or disable the control for editing a Tagged Value. Return Value: <ul style="list-style-type: none"> • True to allow this user to make changes by enabling the control • False to completely disable edit of this property by disabling the control
CanEditVersion()	Enable or disable the control for editing a version. Return Value:

	<ul style="list-style-type: none"> • True to allow this user to make changes by enabling the control • False to completely disable edit of this property by disabling the control
OnPreNewElement(ElementType, ElementStereotype)	<p>Allow or disallow the creation of the specified element.</p> <p>Return Value:</p> <ul style="list-style-type: none"> • True to allow this user to create the element/connector • False to prevent this user from creating the element
OnPreNewConnector(ConnectorType, ConnectorSubType, ConnectorStereotype)	<p>Allow or disallow the creation of the specified connector.</p> <p>Return Value:</p> <ul style="list-style-type: none"> • True to allow this user to create the element/connector • False to prevent this user from creating the element
PreAllowPhaseUpdate(OldValue, NewValue)	<p>Determine what information is required to validate this change.</p> <p>Return Value: Semi-colon separated list of additional data required in order to validate this change.</p> <p>Supported Data Type:</p> <ul style="list-style-type: none"> • Tests - fill the Tests array in the WorkflowContext object
PreAllowStatusUpdate(OldValue, NewValue)	<p>Determine what information is required to validate this change.</p> <p>Return Value: Semi-colon separated list of additional data required in order to validate this change.</p> <p>Supported Data Type:</p> <p>Tests - fill the Tests array in the WorkflowContext object</p>
PreAllowTagUpdate(TagName, OldValue, NewValue)	<p>Determine what information is required to validate this change.</p> <p>Return Value: Semi-colon separated list of additional data required in order to validate this change.</p> <p>Supported Data Type:</p> <p>Tests - fill the Tests array in the WorkflowContext object</p>
PreAllowVersionUpdate(OldValue, NewValue)	<p>Determine what information is required to validate this change.</p> <p>Return Value: Semi-colon separated list of additional data required in order to validate this change.</p> <p>Supported Data Type:</p> <p>Tests - fill the Tests array in the WorkflowContext object</p>

Functions - Create a Search With User Tasks

These are functions that Enterprise Architect calls to create a search with user tasks.

Functions

Function	Action
GetWorkflowTasks	Describe the searches that this user must run. Return Value: Ignored

Filled Workflow Data Structures

These are the workflow data structures (objects) that Enterprise Architect fills.

Data Structures

Workflow Data Structure	Description
WorkflowUser	<p>This object provides information about the user currently logged in to the model. It is filled by Enterprise Architect before any function is called by Enterprise Architect; it has these properties:</p> <ul style="list-style-type: none"> Username - the username for login to the system (if using Windows Authentication, this matches the Windows username) Firstname - as found in the 'Security Users' dialog Surname - as found in the 'Security Users' dialog Fullname - the combination <Firstname> <Surname> (the form Enterprise Architect uses for 'Author' fields and similar) Department - the department in which the user works, as found in the 'Security Users' dialog <p>Calls: This object calls the IsMemberOf(GroupName) function.</p>
WorkflowContext	<p>This object provides information about the object currently in context. It is filled by Enterprise Architect before any searches except GetWorkflowTasks are run; it has these properties:</p> <ul style="list-style-type: none"> MetaType - the type of the current object, either an Enterprise Architect core type or a profile-specified metatype Name - as found in the object 'Properties' dialog Status - as found in the object 'Properties' dialog Phase - as found in the object 'Properties' dialog Version - as found in the object 'Properties' dialog Stereotypes - an array of strings for the stereotypes applied to this object Tags - an array of Tagged Values, providing: <ul style="list-style-type: none"> Name - the Tagged Value name Value - the Tagged Value value Tests - an array of tests; only filled during an Allow* call after the PreAllow* call has specified that tests are required; provides these details, as found in the Testing window: <ul style="list-style-type: none"> Name Status RunBy CheckedBy TestClass TestType <p>Calls: This object calls the TagValue(TagName) function.</p>

Functions

Function	Action
IsMemberOf(GroupName)	<p>Check the group membership of the current user.</p> <p>Return Value: Returns the value True if the current user is a member of the group with the specified name.</p>
TagValue(TagName)	<p>Get the value from a named tag.</p> <p>Return Value: Returns the value of the first Tagged Value with that name, or an empty string if no Tagged Value with that name exists.</p>

Workflow Data Structures You Fill

These are the workflow data structures (objects) that you can fill.

Data Structures

Workflow Data Structure	Description
WorkflowStatus	<p>Use this data structure to provide information on the status of the object.</p> <ul style="list-style-type: none">• LogEntry - set to True or False to indicate whether or not a log item should be recorded• Reason - indicate what reason should be recorded in the log• Action - indicate how to display the log message; valid values are: MessageBox, StatusBar and Output (default)
WorkflowSearches	<p>Use this data structure to provide an array of searches.</p> <p>Use Redim WorkflowSearches(x) to specify the number of searches being provided.</p> <p>Each search has these attributes:</p> <ul style="list-style-type: none">• Name - the name of this search• Group - the name of the group that this search should appear under in the 'Search' combo box• ID - the GUID for this search• Tasks - the array of tasks that this search looks for; an entry describes how to find all objects required to meet a particular task:<ul style="list-style-type: none">- Name - the name of the task, as displayed in the Model Search view; workflow searches are grouped by this field by default- Conditions - an array of conditions, all of which must be matched for an object to be included in this task; a condition is a comparison of a single field to a value:<ul style="list-style-type: none">- Column - the name of the field- Operator - operator types, either = (provide matching values only) or <> (provide non-matching values only)- Value - if this contains a comma, the string is treated as a comma separated list of values to compare against; otherwise the string is a single value to compare against

Functions You Call

These are functions that Enterprise Architect provides for you to call.

Functions

Function	Action
NewSearch(name, group, guid, taskcount)	Create a new search object to be included in WorkflowSearches. Initialize each member. Return Value: The created search
NewTask(name, conditioncount)	Create a new task object to be included in a search. Initialize each member. Return Value: The created task
NewCondition(column, operator, value)	Create a new condition object to be included in a task. Initialize each member. Return Value: The created condition
SetLastError(message, outputMethod)	<p>Called on user input to these element properties:</p> <ul style="list-style-type: none"> • Status • Phase • Version, and • Tagged Values <p>It logs and/or reports the provided message to the user. It can be called within the functions:</p> <ul style="list-style-type: none"> • AllowPhaseUpdate • AllowStatusUpdate • AllowTagUpdate • AllowVersionUpdate • preAllowPhaseUpdate • preAllowStatusUpdate • preAllowTagUpdate • preAllowVersionUpdate <p>For example:</p> <pre>public function AllowPhaseUpdate(OldValue, NewValue) AllowPhaseUpdate = false SetLastError "No updating to phase allowed", "messagebox" end function</pre> <p>Parameters:</p> <ul style="list-style-type: none"> • message: Text • outputMethod: can be "messagebox", "statusbar" or "outputwindow"; this parameter is case sensitive

	Return Value: The message
--	---------------------------

