ENTERPRISE ARCHITECT

User Guide Series

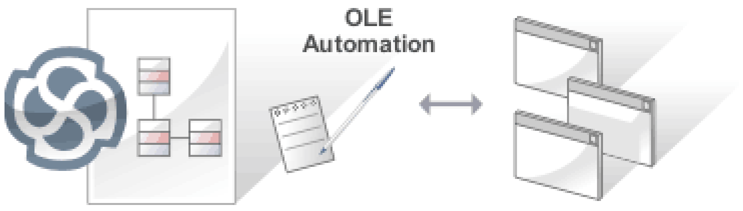
Enterprise Architect Object Model

Author: Sparx Systems
Date: 2025-05-05
Version: 17.1

CREATED WITH ENTERPRISE ARCHITECT

Table of Contents

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 convenient 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 metadata.

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 Browser window 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

This procedure caters for the syntax and frameworks of version 6.0. More recent versions have the same framework as other .Net languages with only syntax differences, and therefore use a similar process to that described for Microsoft C#, later in this topic.

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

Note that recent versions of Delphi are developed by Embarcadero.

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	See the general library documentation on the use of Classes. This example creates and opens a repository object:
	procedure TForm1.Button1Click(Sender: TObject);
	var
	r: TRepository;
	b: boolean;
	begin
	r:= TRepository.Create(nil);
	b:= r.OpenFile('c:\eatest.eap');
	end;

Microsoft C#

Step	Action
1	Select the 'Visual Studio Project | Add Reference' menu option.
2	Click on the 'Browse' tab.
3	Navigate to the folder in which you installed Enterprise Architect; usually: Program Files/Sparx Systems/EA Select Interop.EA.dll

4	See the general library documentation on the use of Classes. This example creates and opens a repository object:
	private void button1_Click(object sender, System.EventArgs e)
	{
	EA.Repository r = new EA.Repository();
	r.OpenFile("c:\\eatest.eap");
	}

Java

Step	Action

1	Copy the file: SSJavaCOM.dll from the Java API subdirectory of your installed directory, usually: Program Files/Sparx Systems/EA into any location within the Windows PATH windows\system32 directory. Note: The Java API loads the last installed Enterprise Architect and isn't affected when using either the 32 or 64 Version of DLL, as long as the SSJavaCOM dll can be found by the java runtime.
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:
	`public void OpenRepository()`
	`{`
	`org.sparx.Repository r = new org.sparx.Repository();`
	`r.OpenFile("c:\\eatest.eap");`
	`}`

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	Instructions for using the interface are provided through sample code. There are several sets of examples: • 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 **sparxsystems.com/resources/developers/autint_vb.html** • Additional samples are available from the Sparx Systems website; see the *Available Resources* topic
Tips and Tricks	Also note these tips and tricks: • 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	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.
	`repository.CloseFile();`
	`repository.Exit();`
	`repository = null;`
	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:
	GC.Collect();
	GC.WaitForPendingFinalizers();
	There are additional concerns when controlling a running instance of Enterprise Architect that loads Add-Ins - see the *Tricks and Traps* topic for details.

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 Browser window 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 > Appearance > Preferences > Other Options > 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.	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.	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.	sparxsystems.com/bin/CS_AddinFramework.zip
An extension on the CS_AddinFramework example showing how to export Tagged Values to a .csv file.	sparxsystems.com/bin/CS_AddinTaggedCSV.zip
A basic Add-In skeleton written in Delphi.	sparxsystems.com/bin/DelphiDemo.zip
A simple example Add-In written in C#.	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

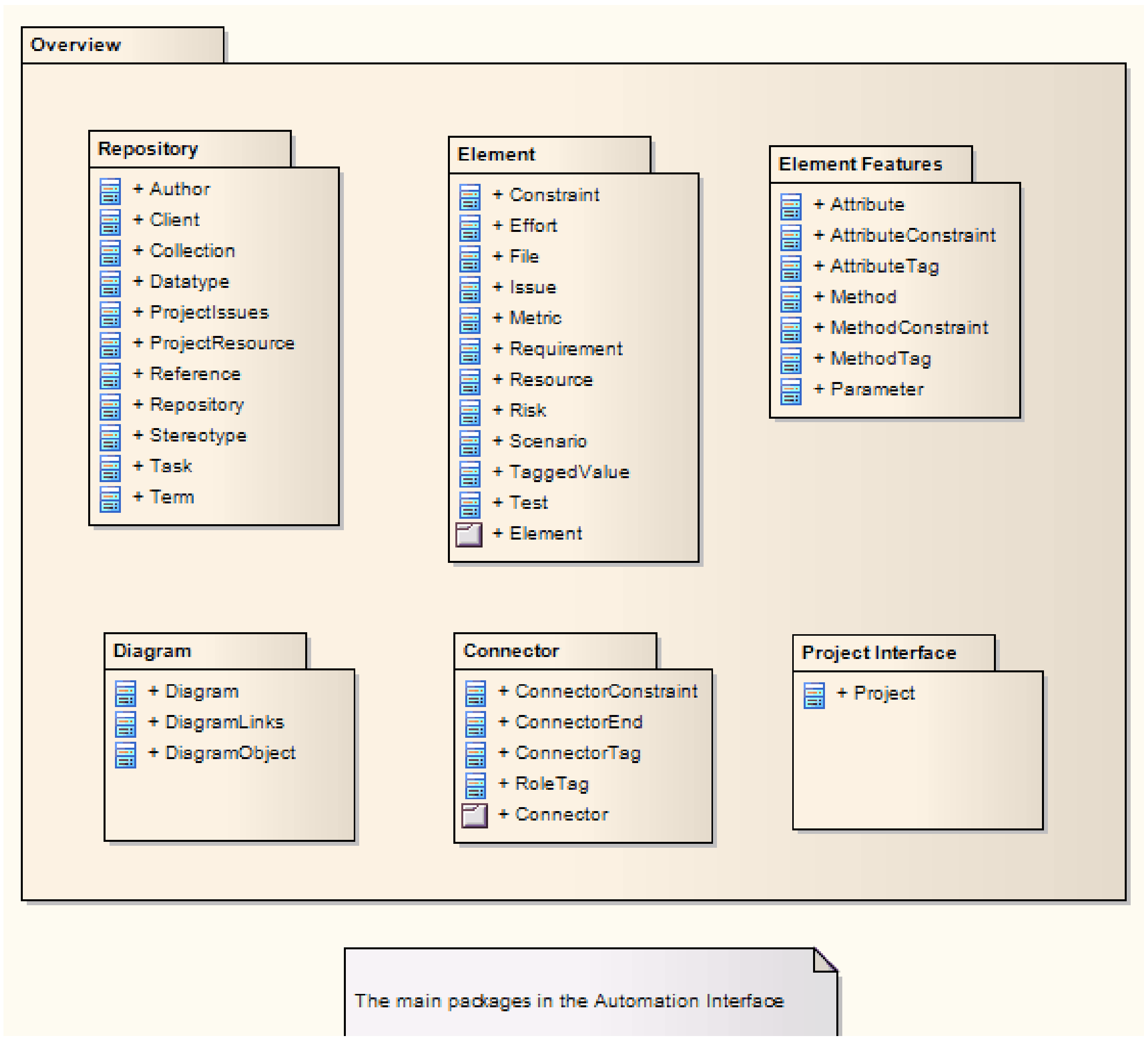
This section provides an overview of the main components of the Automation Interface.

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 createable 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 > Diagram Layout ' ribbon option.

Enum Values

Value	Meaning
lsCrossReduceAggressive	Perform aggressive Cross-reduction in the layout process (time consuming).
lsCycleRemoveDFS	Use the Depth First Cycle Removal algorithm.
lsCycleRemoveGreedy	Use the Greedy Cycle Removal algorithm.
lsDiagramDefault	Use existing layout options specified for this diagram.
lsInitializeDFSIn	Initialize the layout using the Depth First Search Inward algorithm.
lsInitializeNaive	Initialize the layout using the Naïve Initialize Indices algorithm.
lsInitializeDFSOut	Initialize the layout using the Depth First Search Outward algorithm.
lsLayeringLongestPathSink	Layer the diagram using the Longest Path Sink algorithm.
lsLayeringLongestPathSource	Layer the diagram using the Longest Path Source algorithm.
lsLayeringOptimalLinkLength	Layer the diagram using the Optimal Link Length algorithm.
lsLayoutDirectionDown	Direct connectors to point down.
lsLayoutDirectionLeft	Direct connectors to point left.
lsLayoutDirectionRight	Direct connectors to point right.
lsLayoutDirectionUp	Direct connectors to point up.
lsProgramDefault	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.
pageLandscape	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.piProfessional)
 ...
 else if (theEdition == EAEditionTypes.piCorporate)
...
```

The enumeration defines these formal values:

- piLite
- 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

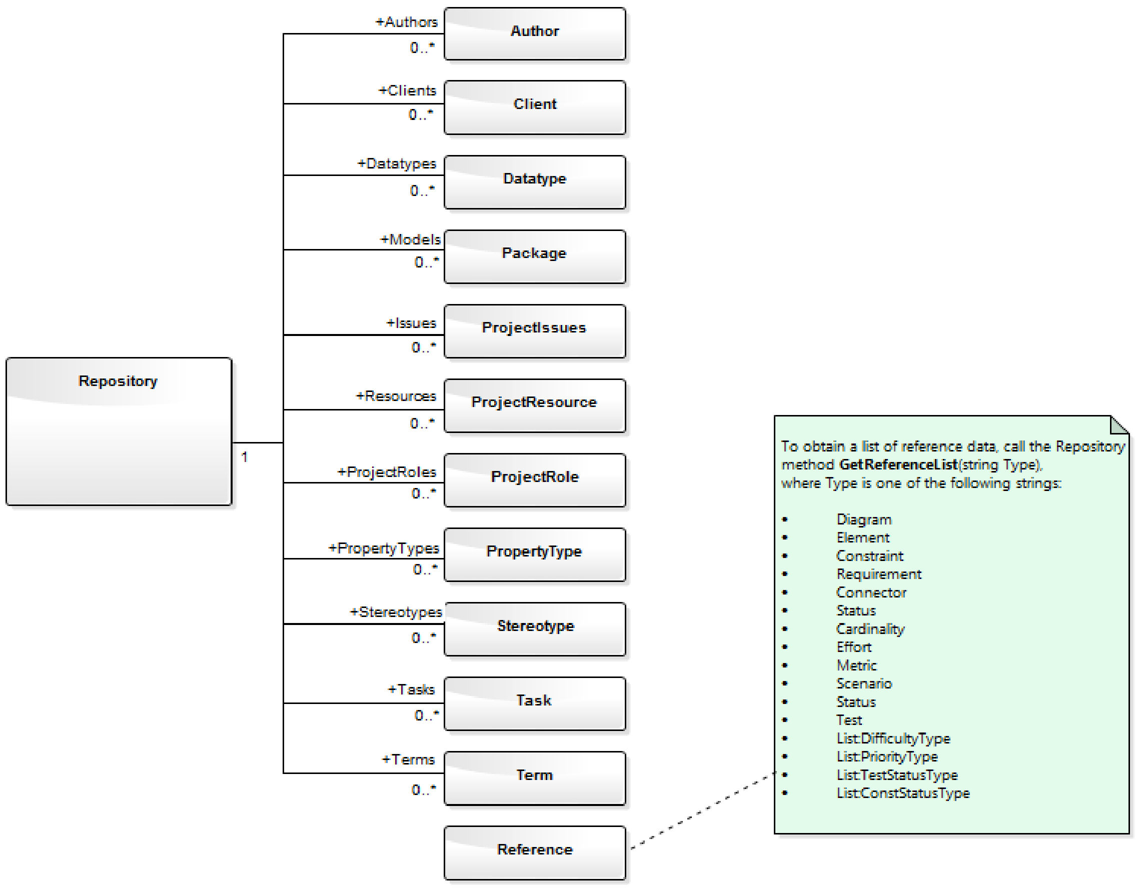
These enumeration values are used in the Project.ExportPackageXMI() and Project.ExportPackageXMIEx() methods, to specify the XMI export type.

- 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
- xmiEcore = 19
- xmiBPMN20 = 20
- xmiXPDL22 = 21
- xmiEA251 = 22
- xmiARCGIS = 23
- xmiNative = 24
- xmiEA2511 = 25
- xmiNativeXEA = 26

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.


Repository
1
+Authors 0..* Author
+Clients 0..* Client
+Datatypes 0..* Datatype
+Models 0..* Package
+Issues 0..* ProjectIssues
+Resources 0..* ProjectResource
+ProjectRoles 0..* ProjectRole
+PropertyTypes 0..* PropertyType
+Stereotypes 0..* Stereotype
+Tasks 0..* Task
+Terms 0..* Term
Reference

To obtain a list of reference data, call the Repository method **GetReferenceList**(string Type), where Type is one of the following strings:

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

Author Class

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

Associated table in repository

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 repository

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 the 'AddNew' function 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 function. If Update() is not called the object is left in an indeterminate state. When an error occurs an exception will be thrown, including when the user does not have Security permission to modify the specify type. Parameters:

	• Name: String • Type: String (up to 30 characters long)
Delete(short index)	Void Notes: Deletes the item at the selected reference. Parameters: • index: Short
DeleteAt(short index, boolean Refresh)	Void Notes: Deletes the item at the selected index. The second parameter is currently unused. Parameters: • index: Short • Refresh: Boolean
GetAt(short index)	Object Notes: Retrieves the array object using a numerical index. If the index is out of bounds, an error occurs. Parameters: • index: Short
GetByName(string Name)	Object Notes: Gets an item in the current collection by name. Supported for Model, Package, Element, Diagram and element TaggedValue collections. 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. Parameters: • Name: String
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Refresh()	Void 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.
Update()	Boolean Notes: Updates the current Collection object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

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 - *Not used.* 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 - *Not used*. 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: diagram.DiagramObjects.AddNew("l=200;r=400;t=200;b=600;", "") 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 - *Not used.*
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 repository

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: • 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: • 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 Browser window. Packages can be accessed either through the Repository Models collection (a Model is a special form of Package) or through the 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 repository

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

	Notes: Read/Write Indicates if XMI 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).
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. When setting this attribute, LastError (from the GetLastError method) will be non-empty on error.

TreePos	Long Notes: Read/Write The relative position in the tree compared to other Packages (use to sort Packages).
TypeInfoProperties	Notes: Read only Returns an interface pointer of TypeInfoProperties.
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 XMI.
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. User Security applies to the use of this function; if the user does not have permission to apply or release locks on elements, diagrams and Packages, the operation will fail. Returns True if the operation is successful; returns False if the operation is unsuccessful. Use GetLastError() to retrieve error information. Parameters: • 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. User Security applies to the use of this function; if the user does not have permission to apply or release locks on elements, diagrams and Packages, the operation will fail. Returns True if the operation is successful; returns False if the operation is unsuccessful. Use 'GetLastError()' to retrieve error information.

	Parameters • 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. User Security applies to the use of this function; if the user does not have permission to apply or release locks on elements, diagrams and Packages, the operation will fail. 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)	Boolean Notes: Applies user locks to the Package object, and all of the Packages, diagrams and elements contained within that Package. User Security applies to the use of this function; if the user does not have permission to apply or release locks on elements, diagrams and Packages, the operation will fail. Returns True if the operation is successful; returns False if the operation is unsuccessful. Use GetLastError() to retrieve error information. Parameters • 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	LDISPATCH Notes: Inserts a copy of the Package into the same parent as the original Package. Returns the newly-created Package.
FindObject (string DottedID)	LPDISPATCH Notes: Returns a Package, element, attribute or operation matching the parameter DottedID. If the DottedID is not found, an error is returned: *Can't find matching object.* Parameters • 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()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
GetTXAlias (string Code, long Flag)	String Notes: Returns the Alias of the element for a given language. Parameters • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long

	- 0 = Get the currently-stored translated Alias - 1 = Get the currently-stored translated Alias, and auto translate if the original Alias has changed - 2 = Always fetch the translated Alias from online
GetTXNote (string Code, long Flag)	String Returns the Notes of the element for a given language. Parameters • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long - 0 = Get the currently-stored translated Notes - 1 = Get the currently-stored translated Notes, and auto translate if the original Notes have changed - 2 = Always fetch the translated Notes from online
SetTXAlias (string Code, string Translation)	String Notes - Set the translated Alias of the element for a given language. • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated Alias
SetTXName (string Code, string Translation)	String Notes - Set the translated name of the element for a given language. • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated name
SetTXNote (string Code, string Translation)	String Notes - Set the translated Notes of the element for a given language. • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated Notes
GetTXName (string Code, long Flag)	String Notes: Returns the name of the element for a given language. Parameters • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long - 0 = Get the currently-stored translated name - 1 = Get the currently-stored translated name, and auto translate if the original name has changed - 2 = Always fetch the translated name from online
ReleaseUserLock ()	Boolean Notes: Releases user locks and group locks from the Package object, and all of the Packages, diagrams and elements contained within that Package. User Security applies to the use of this function; if the user does not have permission to apply or release locks on elements, diagrams and Packages, the operation will fail. Returns True if the operation is successful; returns False if the operation is

	unsuccessful. Use GetLastError() to retrieve error information.
ReleaseUserLockRecursive (boolean IncludeElements, boolean IncludeDiagrams, boolean IncludeSubPackages)	Boolean Notes: Releases user locks from the Package object, and all of the Packages, diagrams and elements contained within that Package. User Security applies to the use of this function; if the user does not have permission to apply or release locks on elements, diagrams and Packages, the operation will fail. Returns True if the operation is successful; returns False if the operation is unsuccessful. Use GetLastError() to retrieve error information. Parameters IncludeElements: Boolean - Recursively release user locks from child elements IncludeDiagrams: Boolean - Recursively release user locks from child diagrams IncludeSubPackages: Boolean - Recursively release user locks from child Packages
SetReadOnly (boolean ReadOnly, boolean IncludeSubPkgs)	Void Notes: Sets a Package Flag to mark a Package as ReadOnly=1. If Project Security is enabled, the user must have 'Configure Packages' permission to use this method. Throws an exception if the operation fails due to the user not having 'Configure Packages' permission; use 'GetLastError()' to retrieve error information. Parameters • 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 on just the object Package, or on the object Package and all of the nested sub-Packages that it contains; if: False, only the flag on the object Package is set or cleared 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 When working with Version Controlled Packages, the Read Only flag can be applied to Packages whether they are checked-in or checked-out. User Security applies to setting this flag - if you are prevented from editing the Package, you are also prevented from setting the flag.
Update ()	Boolean Notes: Updates the current Package object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information. 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. The element attribute contains information about, for example, Stereotypes, Constraints or Files - all the attributes of a typical element.
VersionControlAdd (string ConfigGuid, string XMLFile, string Comment, boolean KeepCheckedOut)	Void Notes: Places the Package under Version Control, using the specified Version Control Configuration and the specified XMI filename. Throws an exception if the operation fails; use GetLastError() to retrieve error

	information. It is recommended that the Package be saved using Update() before calling VersionControlAdd(), so that any outstanding changes are not lost. Parameters • 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)	Void Notes: Perform checkin of the Version Controlled Package (also see VersionControlCheckinEx). Throws an exception if the operation fails; use GetLastError() to retrieve error information. Parameters • Comment: String - Log message that is added to the Version Controlled file's history (where applicable)
VersionControlCheckinEx (string Comment, boolean PreserveCrossPkgRefs)	Void Notes: Perform check-in of the Version Controlled Package. Throws an exception if the operation fails; use GetLastError() to retrieve error information. Parameters • 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 *Learn more*
VersionControlCheckout (string Comment)	Void Notes: Perform checkout of the Version Controlled Package. Throws an exception if the operation fails; use GetLastError() to retrieve error information. Parameters: • Comment: String - Log message that is added to the Version Controlled file's history (where applicable) 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.
VersionControlGetLatest	Void

(boolean ForceImport)	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. Parameters: • 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: - 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 See also the menu option 'Version Control | Get Latest'. 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.

VersionControlGetStatus ()	Long
	Notes: Returns the Version Control status of the Package, as recorded in the current project database.
	Throws an exception if the operation fails; use GetLastError() to retrieve error information.
	Return value maps to this enumerated type:
	enum EnumCheckOutStatus
	{
	csUncontrolled = 0,
	csCheckedIn,
	csCheckedOutToThisUser,
	csReadOnlyVersion,
	csCheckedOutToAnotherUser,
	csOfflineCheckedIn,
	csCheckedOutOfflineByUser,
	csCheckedOutOfflineByOther,
	csDeleted,
	};
	• 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 Packagehas 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)	Void Notes: Perform a checkin of the Version Controlled Package, whilst keeping the Package checked-out. Throws an exception if the operation fails; use GetLastError() to retrieve error information. 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. Parameters: • Comment: String - Log message added to the Version Controlled file's history (where applicable)
VersionControlRemove ()	Void Notes: Removes Version Control from the Package. Throws an exception if the operation fails; use 'GetLastError()' to retrieve error information.
VersionControlResynchPkgStatus (boolean ClearSettings)	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. Parameters: • ClearSettings: Boolean - used if the Package file associated with the specified Package is reported by the Version Control provider as uncontrolled; if ClearSettings is: True, the Version Control settings are cleared from the Package False, the Version Control settings remain unchanged

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 repository

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 repository

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 repository

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 repository

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.

GetReferenceList (string Type)

Notes: Uses the list type to get a pointer to a Reference List object.

Parameters:

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

Example:

```
var statusList as EA.Reference;
statusList = Repository.GetReferenceList("Status");
Session.Output("Status Count: " + statusList.Count);
for (var i=0; i < statusList.Count; i++)
{
	Session.Output("#" + (i+1) + ": " + statusList.GetAt(i));
}
```

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 Notes: Get the item at the specified index. Parameters: • Index: Short - The index of the item to retrieve from the list
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Refresh()	Short Notes: Refresh the current list and return the count of items.

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 repository

<none>

Repository Attributes

Attribute	Remarks
Authors	Collection Notes: Read only 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. Use AddNew(), Delete() and GetAt() to manage Authors.
BatchAppend	Boolean Notes: Read/Write Set this property to True when your automation client has to rapidly insert many elements, operations, attributes and/or operation parameters. Set to False when work is complete. This can result in 10- to 20-fold improvement in adding new elements in bulk.
Clients	Collection Notes: Read only A list of Clients associated with the project. You can modify, delete and add new Client objects using this collection.
ConnectionString	String Notes: Read only The filename/connection string of the current Repository. 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: 0 - MYSQL 1 - SQLSVR 3 - ORACLE 4 - POSTGRES 8 - ACCESS2007 9 - FIREBIRD 11 - SQLITE

CurrentSelection	Notes: Read only Provides information on what is selected, and in what location without making any requests to the database.
DataMinerManager	Data Miner object Notes: Returns a pointer to the EA.DataMinerManager interface.
Datatypes	Collection Notes: Read only The Datatypes collection. This contains a list of Datatype objects, each representing a data type definition for either data modeling or code generation purposes.
EAEdition	EAEditionTypes Notes: Read only Returns the current level of core licensed functionality available. This property returns **Corporate** when the edition is Unified or Ultimate. Use 'EAEditionEx' to identify which of these extended editions is available.
EAEditionEx	EAEditionTypes Notes: Read only Returns the current level of extended licensed functionality available (Unified or Ultimate).
EnableCache	Boolean Notes: Read/Write An optimization for pre-loading Package objects when dealing with large sets of automation objects.
EnableUIUpdates	Boolean Notes: Read/Write 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()'.
FlagUpdate	Boolean Notes: Read/Write Instructs Enterprise Architect to update the Repository with the LastUpdate value.
InstanceGUID	String Notes: Read only The identifier string identifying the Enterprise Architect runtime session.
IsSecurityEnabled	Boolean Notes: Read only Indicates whether User Security is enabled for the current repository.
Issues	Collection Notes: Read only

	The System Issues list. Contains ProjectIssues objects, each detailing a particular issue as it relates to the project as a whole.
LastUpdate	String Notes: Read only 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 Browser window 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.
SearchWindow	Notes: Read only Returns a reference to the Enterprise Architect Search Window.

SecurityUser	Notes: Read only Provides information about the currently logged in security user.
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: • 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: • Name: String - the name of the view to activate
ActivateTechnology (string	Notes: Activates an enabled MDG Technology.

TechnologyID)	Parameters: • 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: • Toolbox: String - the name of the Toolbox page to activate • Options: Long - reserved for future use
AddDefinedSearches (string sXML)	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. Parameters: • 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)	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). Parameters: • 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)	Boolean Notes: Deprecated - no longer in use.
AddPropertiesTab (string TabName, string PropXML)	Notes: Create a Properties tab. Returns a PropertiesTab interface if a tab was created successfully, otherwise NULL. Parameters: • TabName: String - Name of the Properties tab • PropXML: String - An XML string defining the values in the tab **Example XML string.** <?xml version='1.0'?> <properties> <group name='theGroup1'> <property id='1' type='text' default='' readonly='false' > <name>TestText</name> <description>this has id=1</description> </property> <property id='2' type='combobox' default='' readonly='false' >

```
   <name>TestCombo</name>
   <value>Two</value>
   <description>this has id=2</description>
   <valuelist>
    <item>One</item>
    <item>Two</item>
    <item>Three</item>
   </valuelist>
  </property>
  <property id='3' type='date' default='currentdate' showcheckbox='false'
readonly='false' >
   <name>TestDate</name>
   <value></value>
   <description>this has id=3</description>
  </property>
  <property id='4' type='checkbox' default='true' readonly='false' >
   <name>TestCheckbox</name>
   <description>this has id=4</description>
  </property>
  <property id='5' type='spin' default='1' min='0' max='100' readonly='false' >
   <name>TestSpin</name>
   <value>7</value>
   <description>this has id=5</description>
  </property>
  <property id='6' type='int' default='1' readonly='false' >
   <name>TestInt</name>
   <value>100</value>
   <description>this has id=6</description>
  </property>
  <property id='7' type='double' default='1' readonly='false' >
   <name>TestDouble</name>
   <value>3.333</value>
   <description>this has id=7</description>
  </property>
  <property id='8' type='memo' default='' readonly='false' >
   <name>TestMemo</name>
   <value></value>
   <description>this has id=8</description>
  </property>
 </group>
 <group name='theGroup2'>
  <property id='22' type='text' default='' readonly='false' >
   <name>Test1</name>
   <value></value>
   <description>this has id=22</description>
```

	<valuelist> <item></item> </valuelist> </property> </group> </properties>
AddTab (string TabName, string ControlID)	activeX custom control 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. Parameters: • TabName: String - used as the tab caption • ControlID: String - the ProgID of the control; for example, "CS_AddinFramework.UserControl1"
AddWindow (string WindowName, string ControlID)	activeX custom control 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. Parameters: • WindowName: String - used as the window title • ControlID: String - the ProgID of the control; for example, "CS_AddinFramework.UserControl1"
AdviseConnectorChange (long ConnectorID)	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. Parameters: • ConnectorID: Long - the ID of the connector
AdviseElementChange (long ObjectID)	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. Parameters: • ObjectID: Long - the ID of the element
CallSBPI (string sbpiPrefix, string Method, string packedParameters)	Notes: Returns a JSON string with the result from the external server. Parameters: • sbpiPrefix: String - Prefix value of the external server • Method: String - Name of the function to call on the external server • [Optional] packedParameters: String - For SBPI Integrations this must match the expected parameters for the specified method; for Custom Services this can pass generic data to the SBPI in any format, but it is suggested you use the packing methods to ensure a correct JSON string structure
ChangeLoginUser (string Name, string Password)	Boolean 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.

	Parameters: • Name: String - the name of the user • Password: String - the password of the user
ClearAuditLogs (Object StartDateTime, Object EndDateTime)	Boolean Notes: Clears all Audit Logs from the model. If StartDateTime and EndDateTime are not null then only log items that fall into this period are cleared. Returns True for success, False for failure. • 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 Parameters: • 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)	Notes: Removes all the text from a tab in the System Output window. Parameters: • Name: String - the name of the tab to remove text from
CloseAddins ()	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.
CloseDiagram (long DiagramID)	Notes: Closes a diagram in the current list of diagrams that Enterprise Architect has open. Parameters: • DiagramID: Long - the ID of the diagram to close
CloseFile ()	Notes: Closes any open file.
CodeMinerService()	Code Miner Service object Notes: Returns a pointer to the EA.CMService interface.
CreateDocumentGenerator( )	Document Generator Notes: Returns a pointer to the EA.DocumentGenerator interface.
CreateModel (CreateModelType CreateType, string FilePath, long ParentWnd)	Boolean 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. Returns True when the new file is created, otherwise returns False. Parameters: • 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 connection details • 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)	Notes: Creates a tab in the System Output window. Parameters: • Name: String - the name of the tab to create
DeletePerspective (string Perspective, long Options)	Boolean Notes: Deprecated - no longer in use.
DeleteTechnology (string ID)	Boolean Notes: Removes a specified MDG Technology resource from the repository. Returns True if the technology is successfully removed from the model. Returns False otherwise. • 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) Parameters: • ID: String - the ID of the technology
EnsureOutputVisible (string Name)	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. Parameters: • Name: String - the name of the tab to make visible
ExecutePackageBuildScript (long ScriptOptions, string PackageGuid)	Notes: Helps you to run the active Package build script based on your current selection in the Browser window. You can also run a script by passing in the Package GUID. Parameters: • ScriptOptions: Long - the script type; can be any one of these numerical values: 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	Notes: Shuts down Enterprise Architect immediately. Used by .NET programmers where the garbage collector does not immediately release all referenced COM objects.
ExtractImagesFromNote (string Notes, string WriteImagePath, string RelativeImagePath)	String 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. Parameters: • 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\")
ExtractSBPIParameter (string packedParameters, string name)	Notes: Returns the value of the parameter name as a string. Parameters: • packedParameters: String - The JSON string to append the Name/Value to; cannot be empty • name: String - The name of the parameter
GenerateMDGTechnology (string Filename)	Boolean Notes: Generates an MDG Technology file using the settings in the given MTS file. The returned value indicates success or failure. Parameters: • Filename: String - the name and path of the MTS file to use
GetActivePerspective ()	String Notes: Deprecated - no longer in use.
GetAttributeByGuid (string Guid)	Attribute 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. Parameters: • Guid: String - the GUID of the attribute to locate
GetAttributeByID (long AttributeID)	Attribute 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. Parameters: • AttributeID: Long - the ID of the attribute to locate
GetConnectorByGuid (string Guid)	Connector 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. Parameters: • Guid: String - the GUID of the connector to locate
GetConnectorByID (long ConnectorID)	Connector Notes: Searches the repository for a connector with a specific ID. Parameters: • ConnectorID: Long - the ID of the connector to locate
GetContextItem (object Item)	ObjectType Notes: Sets a pointer to an item in context within Enterprise Architect. Also returns the corresponding ObjectType. For additional information about ContextItems and the supported ObjectTypes see the 'GetContextItemType' method. Parameters: • Item: Object - the item to point to

GetContextItemType ()	ObjectType 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: • An item selected in the Browser window • An item selected in an open diagram • An item selected in certain dialogs, such as the attribute 'Properties' dialog The supported ObjectTypes can be any one of these values: • otElement • otPackage • otDiagram • otAttribute • otMethod • otConnector
GetContextObject ()	Object Notes: Returns the current context Object.
GetCounts ()	String 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.
GetCurrentDiagram ()	Diagram Notes: Returns a selected diagram.
GetCurrentLoginUser (boolean GetGuid)	String Notes: If security is not enabled in the repository, an error is generated. 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.
GetDiagramByGuid (string Guid)	Diagram 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 a diagram without using the collection GetAt() function. Parameters: • Guid: String - the GUID of the diagram to locate
GetDiagramByID (long DiagramID)	Diagram 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. Parameters: • DiagramID: Long - the ID of the diagram to locate
GetElementByGuid (string Guid)	Element 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. Parameters: • Guid: String - the GUID of the element to locate
GetElementByID (long ElementID)	Element 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. Parameters: • ElementID: Long - the ID of the element to locate
GetElementsByQuery (string QueryName, string SearchTerm)	Collection (of type Element) Notes: Helps you to run a search in Enterprise Architect, returning the result as a collection. For example: GetElementsByQuery('Simple','Class1'), where the results list elements with 'Class1' in the 'Name' and 'Notes' fields. Parameters: • QueryName: String - the name of the search to run, for example 'Simple' • SearchTerm: String - the term to search for
GetElementSet (string IDList, long Options)	Collection (of type Element) 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. Parameters • IDList: String - a comma-separated list of ElementID values • Options: Long - modifies default behavior of this method 1. Returns empty collection when empty IDList parameter is given. 2. Use IDList string as an SQL query to populate this collection. 3. Use IDList string as a Package ID to populate the collection with elements under the given package. (One level only).
GetFieldFromFormat (string Format, string Text)	String Notes: Converts a field from your preferred format to Enterprise Architect's internal format; returns the field in that format. Parameters: • Format: String - The format to convert the field from; valid formats are: - HTML - Full HTML - RTF - Rich Text Format - TXT - Plain text • Text: String - The field to be converted
GetFormatFromField (string Format, string Text)	String 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. Parameters: • Format: String - The format to convert the field to; valid formats are: - HTML - Full HTML - RTF - Rich Text Format

	- TXT - Plain text • Text: String - The field to be converted
GetFormattedName (string Guid, long FlagInclude, string Separator, long FlagFormat)	String Notes: Provides special formatting for the name of the specified object; for example, the fully qualified name of a specific element or feature. Parameters: • Guid: String - The GUID of the object to be formatted • FlagInclude: Long - Items to be included in the formatted name: - fiFeature = &H01 - fiClass = &H02 - fiParents = &H04 - fiPackage = &H08 - fiRootNS = &H10 - fiHiddenNS = &H20 - fiDiagram = &H40 - fiElemAlias = &H80 • Separator: String - The string to use for separating each included item (such as Packages or elements) • FlagFormat: Long - Additional formatting options: - ffReplaceSpaces = &H01 - ffLowercase = &H02 - ffURLEncode = &H04 Example: FormattedName = Repository.GetFormattedName (Element.ElementGUID, fiFeature Or fiClass Or fiParents Or fiPackage Or fiDiagram, "::", 0)
GetGapAnalysisMatrix ()	String Notes: Read Only Returns all Gap Analyses as an XML document.
GetLastError ()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
GetLocalPath (string Type, string Path)	String 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. Parameters: • 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 For example: Repository.GetLocalPath (Java, %Desk%\Javacode\Motor.java) This could return: C:\Users\fbloggs\Desktop\Javacode\Motor.java.
GetMailInterface ()	MailInterface Notes: Returns an instance of the EA.MailInterface; use this interface to automate the process of creating and sending Model Mail messages.

GetMethodByGuid (string Guid)	Method Notes: Returns a pointer to a method in the repository; this is usually found using the MethodGUID property of a method. Parameters: • Guid: String - the GUID of the method to look for
GetMethodByID (long MethodID)	Method Notes: Returns a pointer to a method in the repository; this is usually found using the MethodID property of a method. Parameters: • MethodID: Long - the ID of the method to look for
GetPackageByGuid (string Guid)	Package 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. 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: GetPackageByGuid(Element.ElementGUID) Parameters: • Guid: String - the GUID of the Package to look for
GetPackageByID (long PackageID)	Package Notes: Get a pointer to a Package using an absolute reference number (local ID). This is usually found using the PackageID property of a Package, and stored for later use to open a Package without using the collection GetAt () function. Parameters: • PackageID: Long - the ID of the Package to locate
GetProjectInterface ()	Project 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, running reports and so on.
GetPropertiesTab (string TabName)	Notes: Finds an existing Properties tab. Returns a PropertiesTab interface if the tab exists, otherwise NULL. Parameters: • TabName: String - The name of the 'Properties' tab.
GetReferenceList (string Type)	Reference Notes: Uses the list type to get a pointer to a Reference List object. Parameters: • 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 ()	String Notes: Returns an XML document (as a string), containing definitions of all Relationship Matrix profiles saved in the current model.
GetTechnologyVersion (string ID)	String Notes: Returns the version of a specified MDG Technology resource. Parameters: • ID: String - the specified technology ID
GetTreeSelectedElements ()	Collection Notes: Returns the set of elements currently selected in the Browser window as a collection.
GetTreeSelectedItem (object SelectedItem)	ObjectType Notes: Gets an object variable and type corresponding to the currently selected item in the tree view. 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. The object passed back through the parameter can be a Package, element, diagram, attribute or operation object. Parameters: • SelectedItem: Object - the object to get the variable and type for
GetTreeSelectedItemType ()	ObjectType Notes: Returns the type of the object currently selected in the tree. One of: • otDiagram • otElement • otPackage • otAttribute • otMethod
GetTreeSelectedObject ()	Object 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.
GetTreeSelectedPackage ()	Package Notes: Returns the Package in which the currently selected tree view object is contained.

HasPerspective (string Perspective)	String Notes: Deprecated - no longer in use.
HideAddinWindow ()	Notes: Hides the docked Add-In window.
ImportPackageAsMDGTechnology (string PackageGuid)	Notes: When PackageGuid contains the GUID of an <<mdg technology>> package, will build an MDG Technology from the contents of the package and import it into the current model. Returns: True on success Parameters: • PackageGuid: String - the GUID of the <<mdg technology>> Package
ImportPackageBuildScripts (string PackageGuid, string BuildScriptXML)	Notes: Imports build scripts into a Package in Enterprise Architect. Parameters: • 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
ImportRASAsset (string PackageGUID, string Protocol, string ServerName, string Model, string Storage, string RASGUID, string Password, string Version)	Notes: Imports the specified RAS asset. Returns True on success; check GetLastError on failure. Parameters: • PackageGUID: String - the GUID of the Package to import the asset to • Protocol: String - the protocol the server is using • ServerName: String - the name of the RAS server • Model: String - the name of the RAS model to use • Storage: String - the storage name of the RAS asset • RASGUID: String - the GUID of the RAS asset • Password: String - the password to access the RAS asset • Version: String - the version of the RAS asset to import
ImportTechnology (string Technology)	Boolean Notes: Installs a given MDG Technology resource into the repository. Returns True if the technology is successfully loaded into the model. Otherwise returns False. 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). Parameters: • Technology: String - the contents of the technology resource file
InsertSBPIParameter (string packedParameters, string name, string value)	Notes: Returns a JSON string. Parameters: • packedParameters: String - The JSON string to append the Name/Value to; cannot be empty • name: String - The name of the parameter • value: String - The value of the parameter
InvokeConstructPicker	

(string ElementFilter)	String
	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.
	For example:
	elementid=Repository.InvokeConstructPicker ("IncludedTypes=Class,Component;StereoType=foo,bar")
	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.
	Parameters:
	• 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
	  - IncludedTypes - (mandatory) comma separated list of element types that can be selected in the dialog; for example:
	    Package,Class,Component
	  - MultiSelect - (optional) when set to True ("MultiSelect=True;") allows the Construct picker to select multiple elements
	  - Selection (optional) - list of comma-separated element GUIDs 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
	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.
	Example:
	  val = Repository.InvokeConstructPicker ("IncludedTypes=Class; MultiSelect=True;");
	    while(val != 0)
	    {
	      val = Repository.InvokeConstructPicker("GetNext=True;");
	    }

InvokeFileDialog (string FilterString, long Filterindex, long Flags)	String 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. Parameters: • 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)	String 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 ()'. Returns: • 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 Parameters: • TabName: String - the name of the tab to check for; TabName is case sensitive
IsTechnologyEnabled (string ID)	Boolean Notes: Checks whether the specified string matches the ID of an enabled MDG Technology in Enterprise Architect. Returns True if the string matches the ID of an enabled Technology. Otherwise returns False. Parameters: ID: String - the technology ID to check for; built-in technology IDs include: • ArcGIS ArcGIS • BABOK BABOK • BIZBOK BIZBOK Guide • BPSim BPSim • BRM Business Rule Model • CMMN Case Management Model & Notation • CODEENG Code Engineering • Database Modeling Database Modeling • DMN1.1 DMN1.1 • EAExtended Core Extensions • ERD Entity Relationship Diagram • GML GML • MYSQLTECH MySqlTech • EAReview Review • SIMF SIMF Technology • SOAML SOAML • SysML1.1 SysML1.1 • SysML1.2 SysML1.2 • SysML1.3 SysML1.3 • SysML1.4 SysML1.5 • UML2 Basic UML2 Technology • SYSENG System Engineering • 262139 MDG Technology Builder • TOGAF TOGAF • UAF UAF • UPDM2 UPDM 2.0 • Win32UI Win 32 User Interface Modeling • ZF Zachman Framework Technically, any combination of technologies integrated with or added to Enterprise Architect - including user-developed technologies - could appear in this list. In practice you would only check for one or two technologies at a time.

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: • ID: String - the technology ID to check for
LoadAddins ()	Notes: Loads all Add-Ins from a repository when Enterprise Architect is opened from automation.
MarkupNotes (string Notes, string GlossaryType, string replacement)	String Notes: Returns a string containing the translation of the term. Parameters • Notes: String - a value to perform a translation markup on • GlossaryType: String - a comma-separated list of glossary types; for example, 'tx-french,tx-global' • replacement: String - the value to replace the TERM when found; "<span class=\"notranslate\">#TERM#,/span>"
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: • 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 or Cloud based repository, you will require a valid Enterprise Architect connection string. This can be obtained in one of two ways; both methods require you to first make and open a connection to the model in question with Enterprise Architect: 1) Using the 'Save as Shortcut' menu item, create a shortcut .eap file containing the database connection string; you can call this shortcut file to access the repository. 2) Alternatively, you can right-click on the model's connection entry in the 'Open Project' screen and select 'Edit connection string', this connection string can then be used direct by OpenFile. Parameters: • Filename: String - the filename (or connection string) 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: • Filepath: String - the file path of the Enterprise Architect project to open • Username: String - the user login ID • Password: String - the user password

OpenFileInEditor(string FilePath)	Boolean Notes: Displays a document or source code file in the EA editor Parameters: • FilePath: String - the file path of the document or file to display in the editor
OpenFileInEditorAtLine(string FilePath, integer LineNumber)	Boolean Notes: Displays a document or source code file in the EA editor Parameters: • FilePath: String - the file path of the document or file to display in the editor • LineNumber: Integer - the line number to highlight.
RefreshModelView (long PackageID)	Notes: Reloads a Package or the entire model, updating the user interface. Parameters: • 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)	Notes: Reloads the diagram contents for all open diagrams from the repository. Parameters: • 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)	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. Calling this method within a call to *EA_OnNotifyContextItemModified* is not supported Parameters: • DiagramID: Long - the ID of the diagram to be reloaded
ReloadPackage (long PackageID)	Notes: Reloads a Package and its open child diagrams. Parameters: PackageID: Long - The ID of the Package to reload; if a valid Package ID, only that Package is reloaded.
RemoveOutputTab (string Name)	Notes: Removes a specified tab from the System Output window. Parameters: • Name: String - the name of the tab to be removed
RemoveWindow (string WindowName)	Boolean Notes: Removes an Add-In window that matches the specified WindowName. Parameters: • WindowName: String - the name of the window to remove
RepositoryType ()	String Notes: Returns the currently open database/repository type. Can return one of these values: • 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)	Notes: Runs a search, displaying the results in Enterprise Architect's Model Search window. Parameters: • sQueryName: String - the name of the search to run, for example Simple • sSearchTerm: String - the term to search for • 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 *XML Format* 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)	Boolean Notes: Saves the Audit Logs contained within a model to a specified file. If 'StartDateTime' and 'EndDateTime' are not null then only log items that fall into this period are saved. Returns True for success, False for failure. • This might fail if the user logged into the model does not have the correct access permission Parameters: • 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)	Notes: Saves an open diagram; assumes the diagram is open in the main user interface Tab list. Parameters: • DiagramID: Long - the ID of the diagram to save
SaveDiagramAsUMLProfile (string DiagramGUID, string Filename)	Boolean Notes: Saves a given diagram as a UML Profile, using the settings from the previous time that the specific diagram was saved manually. The returned value indicates success or failure. Parameters: • DiagramGUID: String - the GUID of the Profile diagram to save • Filename: String - the name and path of the file to create; if left blank, the method will use the filename from the previous time the specified diagram was saved

SavePackageAsUMLProfile (string PackageGUID, string Filename)	Boolean Notes: Saves a given Package as a UML Profile, using the settings from the previous time that the specific Package was saved manually. The returned value indicates success or failure. Parameters: • PackageGUID: String - the GUID of the Profile Package to save • Filename: String - the name and path of the file to create; if left blank, the method will use the filename from the previous time the specified Package was saved
ScanXMIAndReconcile ()	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. 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. 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. 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). 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. This functionality is invoked automatically as part of the 'Get All Latest' operation. 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. As a general rule, avoid running this function while you have uncommitted changes in your model. Generally, you: • 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)	Boolean Notes: Shows the docked Add-In window on the specified page. Returns True if a tab of the specified name is now displayed. Parameters • TabName: String - specifies the tab
ShowDynamicHelp (string Topic)	Notes: Shows a Help topic as a view. Parameters: • Topic: String - specifies the Help topic

ShowInProjectView (object Item)	Notes: Selects a specified object in the Browser window. Accepted object types are Package, Element, Diagram, Attribute, and Method; an exception is thrown if the object is of an invalid type. Parameters: • Item: Object - the object to highlight
ShowWindow (long Show)	Notes: Shows or hides the Enterprise Architect User Interface. Parameters: • Show: Long
ShutdownEA (long Flags)	Notes: Closes all open Views and exits Enterprise Architect. Parameters: • Flags: long - if set to 1 then all pending changes will be saved before closing. If set to 0 then all changes will be lost.
SQLQuery (string SQL)	String Notes: Enables execution of a SQL select statement against the current repository. Returns an XML formatted string value of the resulting record set. Parameters: • SQL: String - contains the SQL Select statement
SynchProfile (string Profile, string Stereotype)	Boolean Notes: Synchronizes Tagged Values and constraints of a UML Profile item using the 'Synch Profiled Elements' dialog. Parameters: • 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
VCRPS	Type **VersionControlResynchPkgStatuses (boolean ClearSettings)** Notes: Synchronizes the Version Control status of each Version Controlled Package within the current model with the status reported by your Version Control provider. Parameters: • ClearSettings: Boolean - 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)	Notes: Writes text to a specified tab in the System Output window, and associates the text with an ID. Parameters: • 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

SecurityUser Class

A SecurityUser object represents a named security user.

Associated table in repository

None.

SecurityUser Attributes

Attribute	Remarks
Department	String Notes: Read only Returns the current user's department.
FirstName	String Notes: Read only Returns the current user's first name.
FullName	String Notes: Read only Returns the current user's full name.
Login	String Notes: Read only Returns the current user's login name.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Surname	String Notes: Read only Returns the current user's surname.

SecurityUser Methods

Method	Remarks
IsMemberOf (string GroupId)	Boolean Returns True if the user is part of the specified security group.

	Parameter: • GroupId: String - Name of the security group to check.

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 repository

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

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

Stereotype 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 stereotype object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Task Class

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

Associated table in repository

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 repository

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()	Boolean Notes: Updates the current Term object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Properties Tab Package

The Properties Tab Package contains:

- A function to retrieve a pointer to the interface
- Functions to create or find a Properties tab
- Utility functions for modifying Properties values

You can get a pointer to this interface using the methods Repository.AddPropertiesTab and Repository.GetPropertiesTab.

PropertiesTab Class

PropertiesTab Attributes

Attribute	Remarks

PropertiesTab Methods

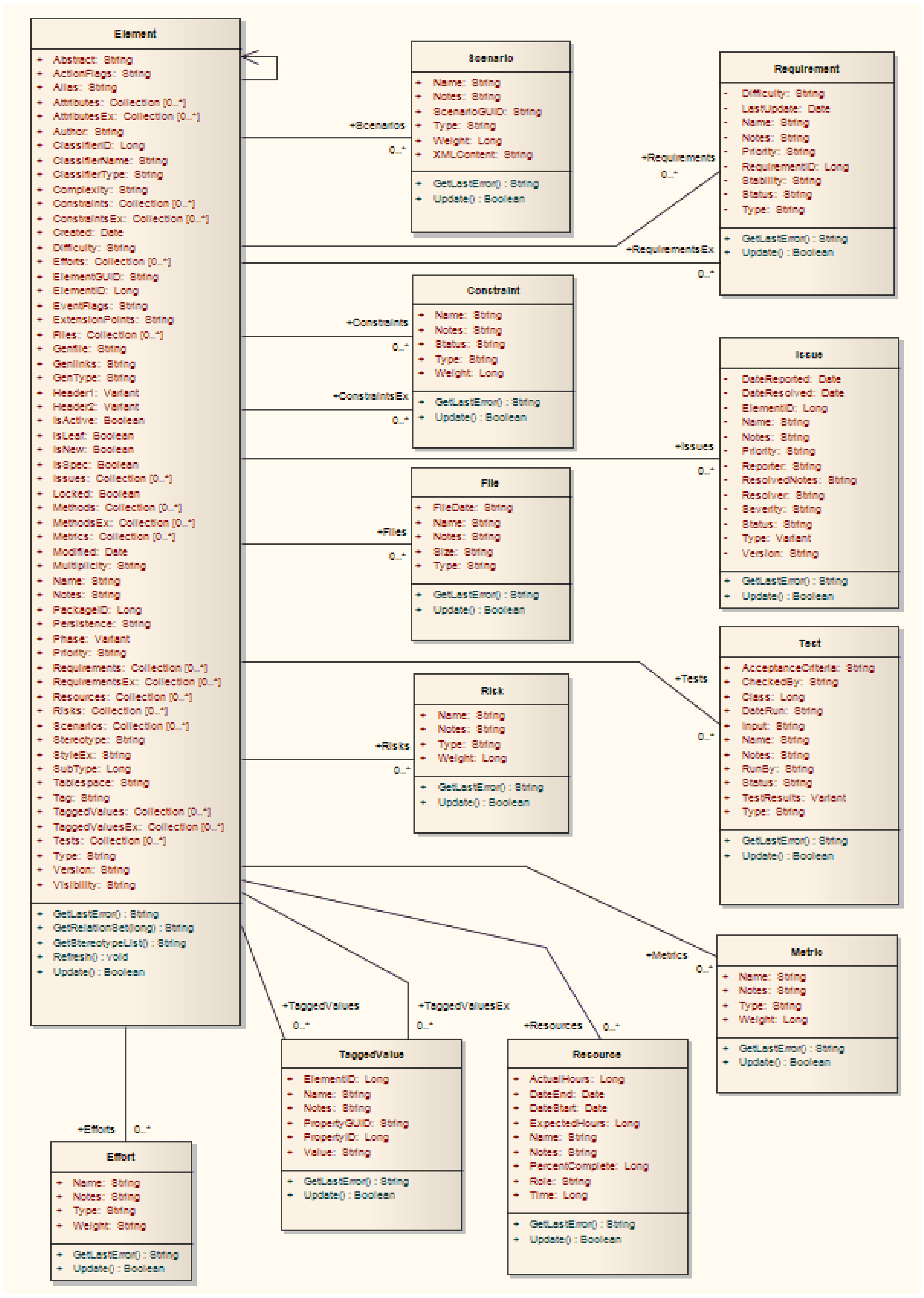
Method	Remarks
AddPropertiesTab (string TabName, string PropXML)	Adds a Properties tab. Returns TRUE if the tab was added. Parameters: • TabName: String - The name of the Properties tab • PropXML: String - An XML string defining the values in the tab
GetLastError ()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
GetPropertiesTab (string TabName)	Notes: Locates a Properties tab. Returns TRUE if the tab is found. Parameters: • TabName: String - The name of the Properties tab
GetPropertiesXML ()	Notes: Returns the XML string of the properties.
GetProperty (long PropID)	Notes: Returns a string of the Property value. Parameters: • PropID: long - The ID value of the property
RemovePropertiesTab ()	Notes: Removes a Properties tab. Returns TRUE if the tab is removed.
SetPropertiesXML (string PropXML)	Notes: Sets the Properties values in the tab. Returns TRUE if the properties were set successfully. Parameters: • PropXML: String - An XML string defining the values in the tab
SetProperty (long PropID, string Value)	Notes: Returns TRUE if the value was set successfully. Parameters: • PropID: long - The ID value of the property to set • Value: String - The value to set the property to

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


Element
+ Abstract: String
+ ActionFlags: String
+ Alias: String
+ Attributes: Collection [0..*]
+ AttributesEx: Collection [0..*]
+ Author: String
+ ClassifierID: Long
+ ClassifierName: String
+ ClassifierType: String
+ Complexity: String
+ Constraints: Collection [0..*]
+ ConstraintsEx: Collection [0..*]
+ Created: Date
+ Difficulty: String
+ Efforts: Collection [0..*]
+ ElementGUID: String
+ ElementID: Long
+ EventFlags: String
+ ExtensionPoints: String
+ Files: Collection [0..*]
+ Genfile: String
+ Genlinks: String
+ GenType: String
+ Header1: Variant
+ Header2: Variant
+ IsActive: Boolean
+ IsLeaf: Boolean
+ IsNew: Boolean
+ IsSpec: Boolean
+ Issues: Collection [0..*]
+ Locked: Boolean
+ Methods: Collection [0..*]
+ MethodsEx: Collection [0..*]
+ Metrics: Collection [0..*]
+ Modified: Date
+ Multiplicity: String
+ Name: String
+ Notes: String
+ PackageID: Long
+ Persistence: String
+ Phase: Variant
+ Priority: String
+ Requirements: Collection [0..*]
+ RequirementsEx: Collection [0..*]
+ Resources: Collection [0..*]
+ Risks: Collection [0..*]
+ Scenarios: Collection [0..*]
+ Stereotype: String
+ StyleEx: String
+ SubType: Long
+ Tablespace: String
+ Tag: String
+ TaggedValues: Collection [0..*]
+ TaggedValuesEx: Collection [0..*]
+ Tests: Collection [0..*]
+ Type: String
+ Version: String
+ Visibility: String
+ GetLastError() : String
+ GetRelationSet(long) : String
+ GetStereotypeList() : String
+ Refresh() : void
+ Update() : Boolean
+Scenarios
0..*
Scenario
+ Name: String
+ Notes: String
+ ScenarioGUID: String
+ Type: String
+ Weight: Long
+ XMLContent: String
+ GetLastError() : String
+ Update() : Boolean
+Requirements
0..*
+RequirementsEx
0..*
Requirement
- Difficulty: String
- LastUpdate: Date
- Name: String
- Notes: String
- Priority: String
- RequirementID: Long
- Stability: String
- Status: String
- Type: String
+ GetLastError() : String
+ Update() : Boolean
+Constraints
0..*
+ConstraintsEx
0..*
Constraint
+ Name: String
+ Notes: String
+ Status: String
+ Type: String
+ Weight: Long
+ GetLastError() : String
+ Update() : Boolean
+Issues
0..*
Issue
- DateReported: Date
- DateResolved: Date
- ElementID: Long
- Name: String
- Notes: String
- Priority: String
- Reporter: String
- ResolvedNotes: String
- Resolver: String
- Severity: String
- Status: String
- Type: Variant
- Version: String
+ GetLastError() : String
+ Update() : Boolean
+Files
0..*
File
+ FileDate: String
+ Name: String
+ Notes: String
+ Size: String
+ Type: String
+ GetLastError() : String
+ Update() : Boolean
+Tests
0..*
Test
+ AcceptanceCriteria: String
+ CheckedBy: String
+ Class: Long
+ DateRun: String
+ Input: String
+ Name: String
+ Notes: String
+ RunBy: String
+ Status: String
+ TestResults: Variant
+ Type: String
+ GetLastError() : String
+ Update() : Boolean
+Risks
0..*
Risk
+ Name: String
+ Notes: String
+ Type: String
+ Weight: Long
+ GetLastError() : String
+ Update() : Boolean
+Metrics
0..*
Metric
+ Name: String
+ Notes: String
+ Type: String
+ Weight: Long
+ GetLastError() : String
+ Update() : Boolean
+TaggedValues
0..*
+TaggedValuesEx
0..*
+Resources
0..*
TaggedValue
+ ElementID: Long
+ Name: String
+ Notes: String
+ PropertyGUID: String
+ PropertyID: Long
+ Value: String
+ GetLastError() : String
+ Update() : Boolean
Resource
+ ActualHours: Long
+ DateEnd: Date
+ DateStart: Date
+ ExpectedHours: Long
+ Name: String
+ Notes: String
+ PercentComplete: Long
+ Role: String
+ Time: Long
+ GetLastError() : String
+ Update() : Boolean
+Efforts
0..*
Effort
+ Name: String
+ Notes: String
+ Type: String
+ Weight: String
+ GetLastError() : String
+ Update() : Boolean

Constraint Class

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

Associated table in repository

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 repository

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

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

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

Associated table in repository

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

	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	Boolean Notes: Read/Write Indicates whether or not the element is a leaf node (and therefore cannot be a parent for any other elements). 1 = True, 0 = False.
IsNew	Boolean Notes: Read/Write Boolean value indicating whether the element is new or not. 1 = True, 0 = False.
IsRoot	Boolean Notes: Read/Write Indicates whether or not the element is a root node (and therefore cannot be descended from another element). 1 = True, 0 = False.
IsSpec	Boolean Notes: Read/Write; Note that this attribute is no longer used in UML 2.0 and later releases, and is provided only to support models maintained in releases of UML prior to 2.0. Boolean value indicating whether the element is a specification or not. 1 = True, 0 = False.
Issues	Collection Notes: Read only Collection of Issue objects.
Locked	Boolean Notes: Read/Write Indicates if the element has been locked against further change.
MetaType	String Notes: Read only The element's domain-specific meta type, as defined by an applied stereotype from an MDG Technology.
Methods	Collection Notes: Read only Collection of Method objects for current element.
MethodsEx	Collection Notes: Read only Collection of Method objects belonging to the current element and its parent elements.
Metrics	Collection Notes: Read only

	Collection of Metric elements for current element.
MiscData	String Notes: Read only This low-level property provides 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: • MiscData(0) corresponds to PData1 • MiscData(1) to PData2, and so on
Modified	Date Notes: Read/Write The date the element was last modified.
Multiplicity	String Notes: Read/Write Multiplicity value for this element.
Name	String Notes: Read/Write The element name; should be unique within the current Package.
Notes	String Notes: Read/Write Further descriptive text about the element.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
PackageID	Long Notes: Read/Write A local ID for the Package containing this element.
ParentID	Long Notes: Read/Write If this element is a child of another, used to set or retrieve the ElementID of the other element; if not, returns 0.
Partitions	Collection Notes: Read only List of logical partitions into which an element can be divided. Only valid for elements that support partitions, such as Activities and States.
Persistence	String Notes: Read/Write

	The persistence associated with this element; can be Persistent or Transient.
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. The string consists of a set of statements in the form: string = '@VAR;Variable=<string>;Value=<string>;Op=<string>;@ENDVAR;' Where: Op = ['=','>','<','>=','<=', '!=','<>'] For example: A set of run states can be created by looping through a set of attributes and forming a concatenated string: eRunState = eRunState + "@VAR;Variable="+ attrib.name + ";Value=" + attrib.value +";Op==;@ENDVAR;";
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. When setting this attribute, LastError (for the GetLastError method) will be non-empty if an error occurs.
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. When setting this attribute, LastError (for the GetLastError method) will be non-empty if an error occurs.
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 • For Event: 0 = Receiver, 1 = Sender • For Class: 1 = Parameterised, 2 = Instantiated, 3 = Both, 0 = Neither, 17 = Association Class 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. • For Note: 1 = Note linked to connector, 2 = Constraint linked to connector • For StateNode: 100 = ActivityIntitial, 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 Note that there are many more Types than indicated in these examples.
Tablespace	String Notes: Read/Write Associated tablespace for a Table element.
Tag	String Notes: Read/Write Corresponds to the 'Keywords' field in the Enterprise Architect user interface.
TaggedValues	Collection Notes: Read only Returns a collection of TaggedValue objects.
TaggedValuesEx	Collection Notes: Read only Returns a collection of TaggedValue objects belonging to the current element and the elements specialized or realized by the current element.
TemplateParameters	Collection Notes: Read Only A collection of TemplateParameter objects.
Tests	Collection Notes: Read only A collection of Test objects for the current element.
TreePos	Long Notes: Read/Write Sets or gets the tree position.
Type	String Notes: Read/Write The element type (such as Class, Component). Note that Type is case sensitive inside Enterprise Architect and should be provided

	with an initial capital (proper case); valid types are: • Action • Activity • ActivityPartition • ActivityRegion • Actor • Artifact • Association • Boundary • Change • Class • Collaboration • Component • Constraint • Decision • 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
TypeInfoProperties	Notes: Read only Returns an interface pointer of TypeInfoProperties.
Version	String Notes: Read/Write The version of the element.
Visibility	String Notes: Read/Write The Scope of this element within the current Package. Valid values are: Public, Private, Protected or Package.

Element Methods

Method	Remarks
ApplyGroupLock(string aGroupName)	Boolean Notes: Applies a group lock to the element 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: • aGroupName: String - the name of the user group for which to set the group lock
ApplyUserLock()	Boolean Notes: Applies a user lock to the element 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.
Clone ()	LDISPATCH Notes: Inserts a copy of the selected element under the same parent as the selected element. Returns the newly-created element.
CreateAssociationClass(lon	

g ConnectorID)	Boolean 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. AssociationClasses are created only where: • 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 Parameters: • ConnectorID: Long - the Connector ID of an Association connector
DeleteLinkedDocument()	Boolean Notes: Removes the Linked Document for the element. This method does not display a confirmatory prompt. Returns True if a document was deleted.
GetBusinessRules()	String Notes: Read Only. Returns all the Business Rules for the element.
GetChart	LDISPATCH Notes: For chart elements returns an interface to the chart
GetDecisionTable()	String Notes: Provides read-only access to a Decision Table XML string. Returns the XML data for the Decision Table as a string.
GetElementGrid()	String Notes: Returns an object of type ElementGrid (a Custom Table Artifact element).
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.
GetTXAlias (string Code, long Flag)	String Notes: Returns the Alias of the element for a given language. Parameters • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long - 0 = Get the currently-stored translated Alias - 1 = Get the currently-stored translated Alias, and auto translate if the original Alias has changed - 2 = Always fetch the translated Alias from online
GetTXName (string Code, long Flag)	String Notes: Returns the name of the element for a given language. Parameters • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long - 0 = Get the currently-stored translated name - 1 = Get the currently-stored translated name, and auto translate if the original name has changed - 2 = Always fetch the translated name from online
GetTXNote (string Code, long Flag)	String Returns the Notes of the element for a given language. Parameters • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long - 0 = Get the currently-stored translated Notes - 1 = Get the currently-stored translated Notes, and auto translate if the original Notes have changed - 2 = Always fetch the translated Notes from online
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: • Stereotype: String - the name of the stereotype to search for
IsAssociationClass	Boolean Notes: Returns whether or not the current element is an AssociationClass.
LoadLinkedDocument(stri	Boolean

ng Filename)	Notes: Loads the document from the specified file into the element's Linked Document. Parameters: • 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 Browser window. Usually called after adding or deleting attributes or methods, when the user interface is required to be updated as well.
ReleaseUserLock()	Boolean Notes: Releases a user lock or group lock on the element object. Returns True if the operation is successful; returns False if the operation is unsuccessful. Use GetLastError() to retrieve error information.
SaveLinkedDocument(string Filename)	Boolean 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. Parameters: • 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)	Void Notes: Sets the visual appearance of the element. Parameters: • 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 *Setting The Style* in the *DiagramObject Class* 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()	Boolean Notes: Sets the composite diagram of the element. Parameters: • GUID: String - the GUID of the composite diagram; a blank GUID will remove the link to the composite diagram
SetCreated(Date NewVal)	Void Notes: Deprecated This method is no longer supported.
SetModified(Date NewVal)	Void

	Notes: Deprecated This method is no longer supported.
SetTXAlias (string Code, string Translation)	String Notes - Set the translated Alias of the element for a given language. • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated Alias
SetTXName (string Code, string Translation)	String Notes - Set the translated name of the element for a given language. • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated name
SetTXNote (string Code, string Translation)	String Notes - Set the translated Notes of the element for a given language. • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated Notes
SynchConstraints(string Profile, string Stereotype)	Boolean Notes: Synchronizes the constraints of a UML Profile item for this element, only if the specified stereotype has been applied. Parameters: • 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)	Boolean Notes: Synchronizes the Tagged Values of a UML Profile item for this element, only if the specified stereotype has been applied. Parameters: • 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	Boolean Notes: Performs the opposite of CreateAssociationClass().
Update()	Boolean Notes: Updates the current element object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

ElementGrid Class

The ElementGrid object represents a Custom Table, which is used to display custom data in tabular format on a diagram, the data being provided by the user rather than generated by the system.

The ElementGrid object is accessible from an Element object, using the GetElementGrid() method.

Associated table in repository

t_object

ElementGrid Methods

Method	Remarks
GetCell (int nrow, int ncell)	Variant Notes: The cell value is return as a variant value. Parameters: • nRow: Integer - the number of the row containing the cell • nCell: Integer - the number of the cell in the row (the column number)
GetColumnCount ()	Integer Notes: Returns the number of columns in the grid.
GetRowCount ()	Integer Notes: Returns the number of rows in the grid.
SetCell (int nRow, int nCell, variant sValue)	Boolean Notes: Sets a value in the specified cell. Parameters: • nRow: Integer - specifies the row into which to insert the value • nCell: Integer - specifies the cell (column number) into which to insert the value • sValue: Variant - specifies the value to set in the cell
SetGridSize (int nRows, int nColumns)	Boolean Notes: Sets the size of the grid in rows and columns. The size can be set and reset; any data outside the bounds of the new grid size will be lost on resize. Parameters: • nRows: Integer - the number of rows in the table grid • nColumns: Integer - the number of columns in the table grid

File Class

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

Associated table in repository

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()	Boolean Notes: Updates the current File object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

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 repository

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 Task.
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 repository

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

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

Requirement Class

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

Associated table in repository

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 repository

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 repository

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 repository

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 '1' 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

	Notes: Read/Write 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.

Scenario 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 Scenario object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

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.

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 Scenarios view '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.

ScenarioExtension 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 ScenarioExtension 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 repository

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

	The value assigned to this tag. 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. When reading existing Tagged Values, if 'Value" = "<memo>" then the developer should read the actual body of text from the 'Notes' attribute.

TaggedValue Methods

Method	Description
GetAttribute(string propName)	String Notes: Returns the text of a single named property within a structured Tagged Value. Parameters: • propName: String - the name of the property for which the text is being returned
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. Parameters: • propName: String - the name of the property for which the text is being set • propValue: the value of the property
Update()	Boolean Notes: Updates the current TaggedValue object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

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 repository

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.

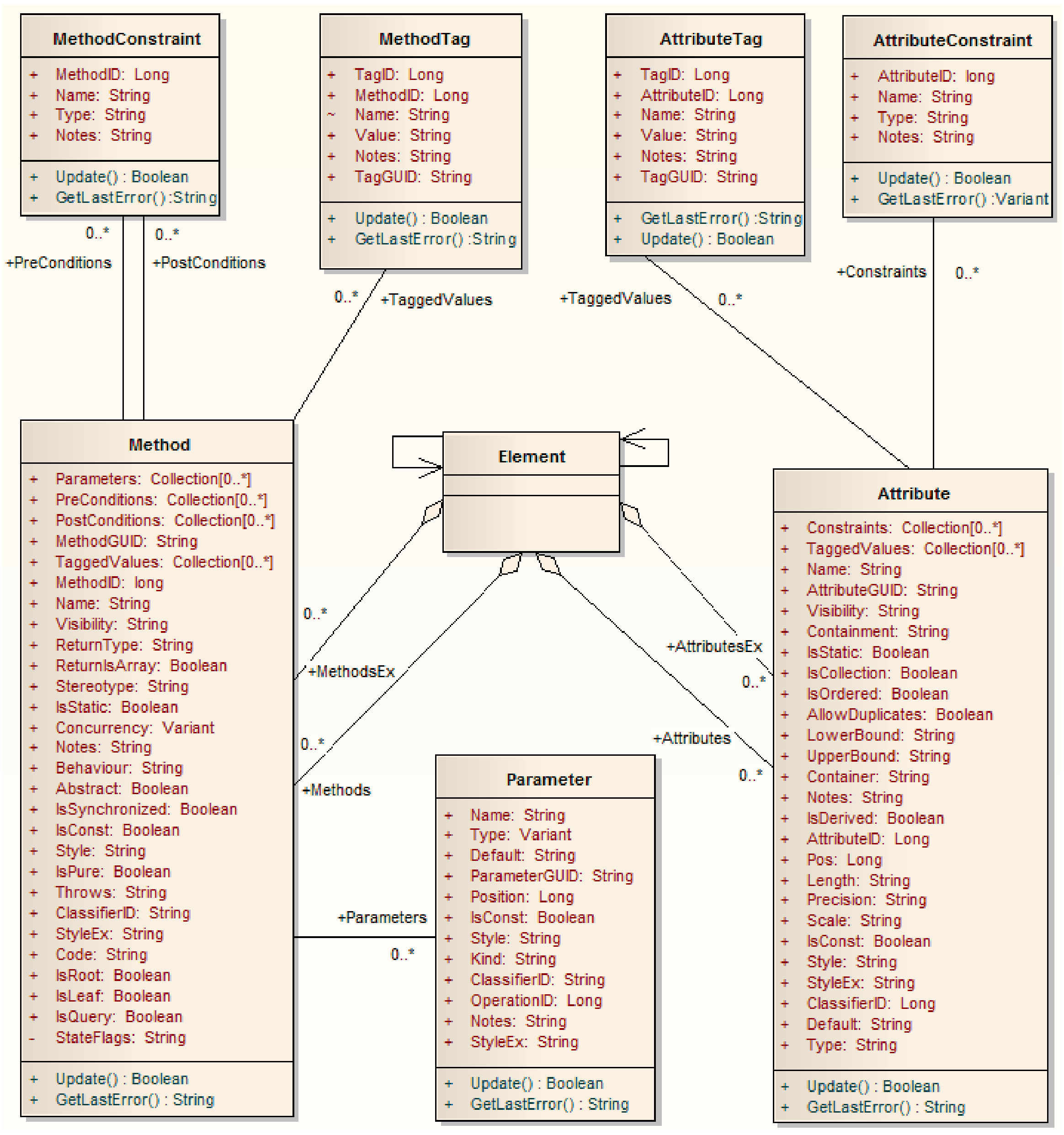
Test Methods

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 their associated constraints and Tagged Values. 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 repository

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.
Constraints	Collection Notes: Read only A collection of AttributeConstraint objects, used to access and manage constraints associated with this attribute.
Container	String Notes: Read/Write The container type.

Containment	String Notes: Read/Write The type of containment - Not Specified, By Reference or By Value.
Default	String Notes: Read/Write The initial value assigned to this attribute.
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.
IsCollection	Boolean Notes: Read/Write Indicates if the current feature is a collection or not. If the attribute represents a database column this, when set, represents a Foreign Key.
IsConst	Boolean Notes: Read/Write A flag indicating if the attribute is Const or not.
IsDerived	Boolean Notes: Read/Write Indicates if the attribute is derived (that is, a calculated value).
IsID	Boolean Notes: Read/Write Indicates if the attribute uniquely identifies an instance of the containing Class, or not.
IsOrdered	Boolean Notes: Read/Write Indicates if a collection is ordered or not. If the attribute represents a database column this, when set, represents a Primary Key.
IsStatic	Boolean Notes: Read/Write 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.
Length	String Notes: Read/Write The attribute length, where applicable.
LowerBound	String Notes: Read/Write A value for the collection lower boundary.

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 *redefinedProperty* 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. When setting this attribute, LastError (for the GetLastError method) will be non-empty if an error occurs.
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. When setting this attribute, LastError (for the GetLastError method) will be non-empty if an error occurs.

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 *subsettedProperty* 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 *ClassifierID*).
TypeInfoProperties	Notes: Read only Returns an interface pointer of TypeInfoProperties.
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.
GetTXAlias (string Code, long Flag)	String Notes: Returns the Alias of the element for a given language. Parameters • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long - 0 = Get the currently-stored translated Alias - 1 = Get the currently-stored translated Alias, and auto translate if the original Alias has changed - 2 = Always fetch the translated Alias from online
GetTXName (string Code, long Flag)	String Notes: Returns the name of the element for a given language. Parameters • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long - 0 = Get the currently-stored translated name - 1 = Get the currently-stored translated name, and auto translate if the original name has changed - 2 = Always fetch the translated name from online
GetTXNote (string Code, long Flag)	String Returns the Notes of the element for a given language. Parameters • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long - 0 = Get the currently-stored translated Notes - 1 = Get the currently-stored translated Notes, and auto translate if the original Notes have changed - 2 = Always fetch the translated Notes from online
SetTXAlias (string Code, string Translation)	String Notes - Set the translated Alias of the element for a given language. • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated Alias
SetTXName (string Code, string Translation)	String Notes - Set the translated name of the element for a given language. • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated name
SetTXNote (string Code, string Translation)	String Notes - Set the translated Notes of the element for a given language. • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog)

	• Translation: String - The translated Notes
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 repository

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 repository

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.

	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. When reading existing Tagged Values, if 'Value' = "<memo>" then the developer should read the actual body of text from the 'Notes' attribute.

AttributeTag 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. This function is rarely used as an exception is thrown when an error occurs.
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 AttributeTag object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

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: • 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

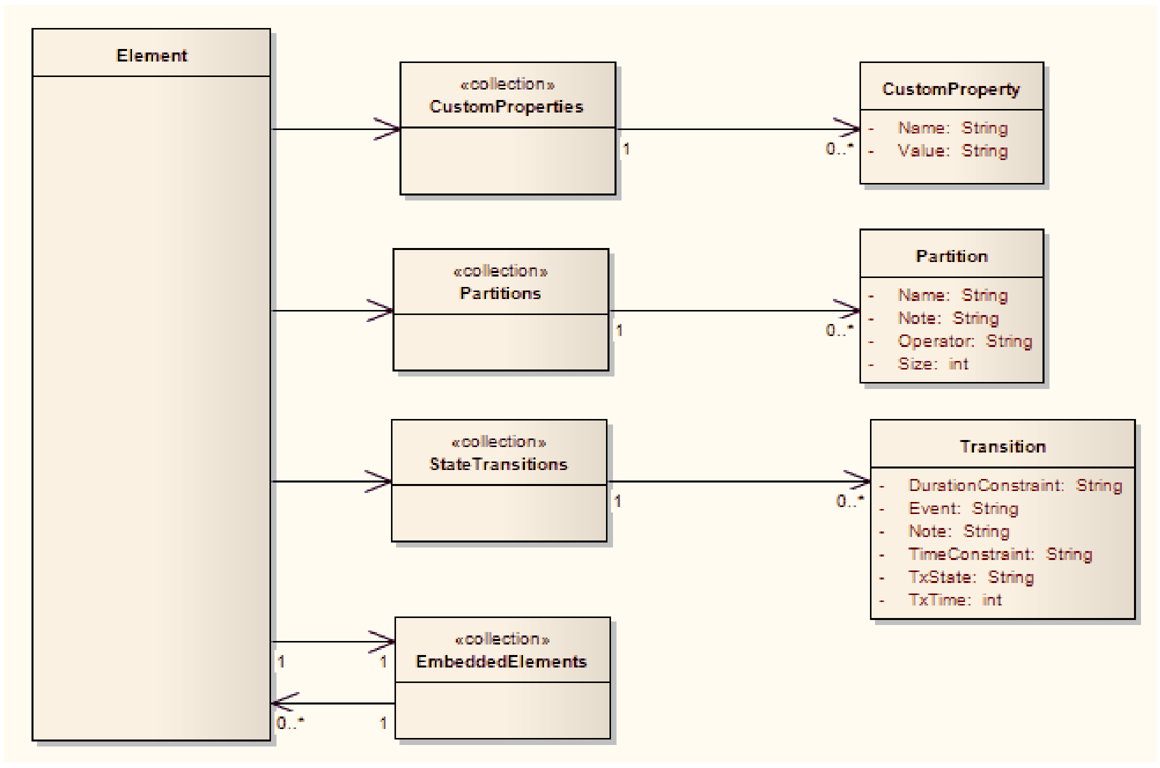
- The number and type of properties vary depending on the actual element

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 repository

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

ParentID	Long Notes: Read only Returns the ElementID of the element that this method belongs to.
Pos	Long Notes: Read/Write Specifies the position of the method within the set of operations defined for a Class.
PostConditions	Collection Class Notes: Read only The PostConditions (constraints) as they apply to this method. This returns a MethodConstraint object of type 'post'.
PreConditions	Collection Class Notes: Read only The PreConditions (constraints) as they apply to this method. This returns a MethodConstraint object of type 'pre'.
ReturnIsArray	Boolean Notes: Read/Write A flag to indicate that the return value is an array.
ReturnType	String Notes: Read/Write The return type for the method; this can be a primitive data type or a Class or Interface type.
StateFlags	String Notes: Read/Write Some flags as applied to methods in State elements.
Stereotype	String Notes: Read/Write The method stereotype (optional). When setting this attribute, LastError (for the GetLastError method) will be non-empty if an error occurs.
StereotypeEx	String Notes: Read/Write 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. When setting this attribute, LastError (for the GetLastError method) will be non-empty if an error occurs.
Style	String Notes: Read/Write Contains the Alias property for this method.

StyleEx	String Notes: Read/Write Advanced style settings, reserved for the use of Sparx Systems.
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: • GUID String - the GUID of an element in the model or a comma-separated list of element GUIDS • <none> - removes the existing Throws set
TypeInfoProperties	Notes: Read only Returns an interface pointer of TypeInfoProperties.
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.
GetTXAlias (string Code, long Flag)	String Notes: Returns the Alias of the element for a given language. Parameters • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long - 0 = Get the currently-stored translated Alias - 1 = Get the currently-stored translated Alias, and auto translate if the original Alias has changed - 2 = Always fetch the translated Alias from online
GetTXName (string Code, long Flag)	String Notes: Returns the name of the element for a given language. Parameters • Code: String - Two-letter language code (found on the 'Translations' page of

	the 'Manage Model Options' dialog) • Flag: Long - 0 = Get the currently-stored translated name - 1 = Get the currently-stored translated name, and auto translate if the original name has changed - 2 = Always fetch the translated name from online
GetTXNote (string Code, long Flag)	String Returns the Notes of the element for a given language. Parameters • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long - 0 = Get the currently-stored translated Notes - 1 = Get the currently-stored translated Notes, and auto translate if the original Notes have changed - 2 = Always fetch the translated Notes from online
SetTXName (string Code, string Translation)	String Notes - Set the translated name of the element for a given language. • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated name
SetTXAlias (string Code, string Translation)	String Notes - Set the translated Alias of the element for a given language. • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated Alias
SetTXNote (string Code, string Translation)	String Notes - Set the translated Notes of the element for a given language. • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated Notes
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 repository

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

MethodTag Class

A MethodTag is a Tagged Value associated with a method.

Associated table in repository

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.

	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. When reading existing Tagged Values, if 'Value' = "<memo>" then the developer should read the actual body of text from the 'Notes' attribute.

MethodTag 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. This function is rarely used as an exception is thrown when an error occurs.
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 MethodTag object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Parameter Class

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

Associated table in repository

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. When setting this attribute, LastError (for the GetLastError method) will be non-empty if an error occurs.
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. When setting this attribute, LastError (for the GetLastError method) will be non-empty if an error occurs.
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.
TypeInfoProperties	Notes: Read only Returns an interface pointer of TypeInfoProperties.

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

ParamTag Class

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

Associated table in repository

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: • 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 repository

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()	Boolean Notes: Updates the current TemplateParameter object after modifying or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

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.

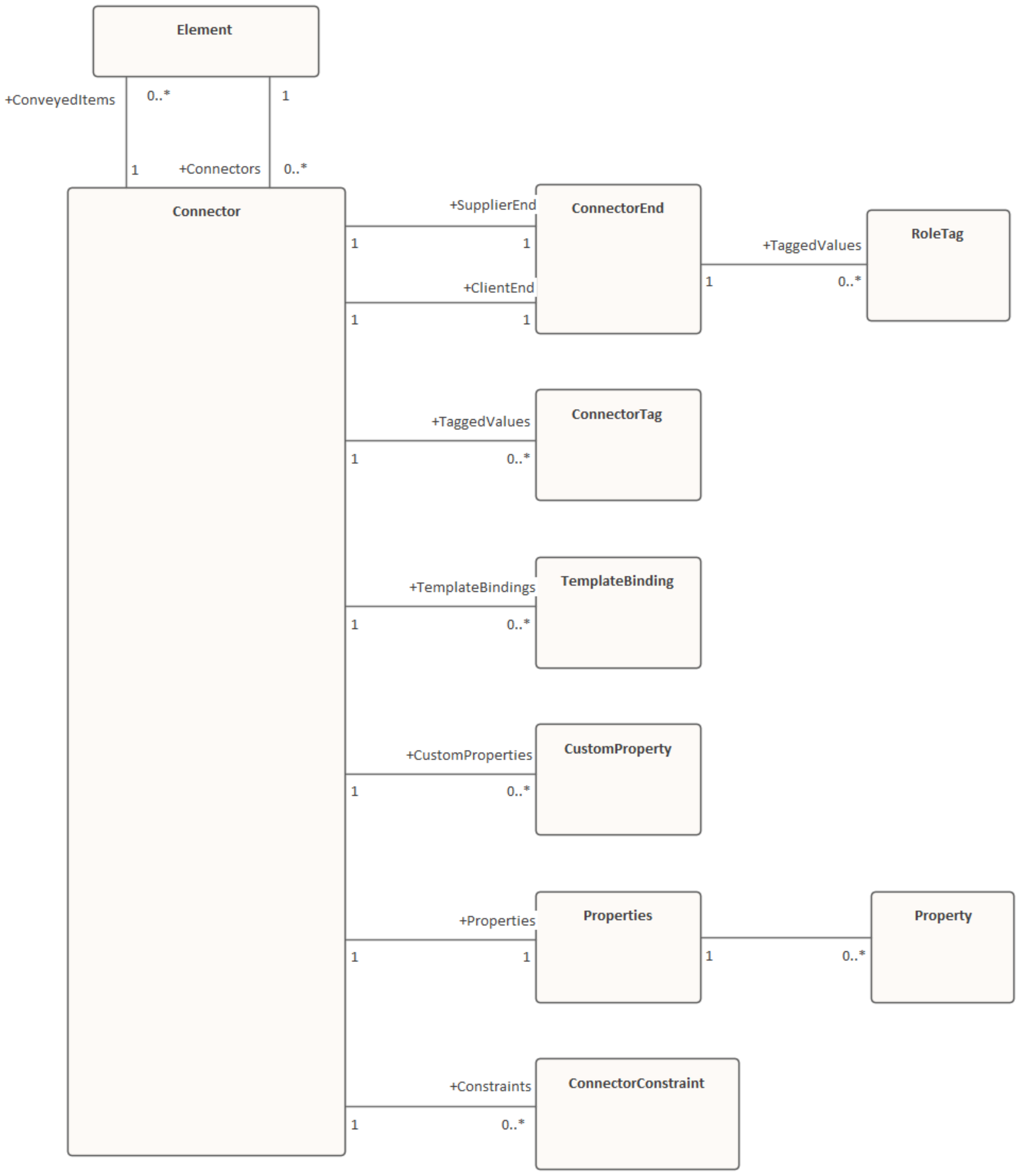
Transition Attributes

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 repository

t_connector

Connector Attributes

Attribute	Remarks
Alias	String Notes: Read/Write

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

	Notes: Read/Write The connector direction, which can be set to one of: • Unspecified • Bi-Directional • Source -> Destination or • Destination -> Source If the connector is non-navigable, set the 'sourceNavigability' and/or 'targetNavigability' attributes.
EndPointX	Long Notes: Read/Write The x-coordinate of the connector's end point. Connector end points are specified in Cartesian coordinates with the origin to the top left of the screen.
EndPointY	Long Notes: Read/Write The y-coordinate of the connector's end point. Connector end points are specified in Cartesian coordinates with the origin to the top left of the screen.
EventFlags	String Notes: Read/Write A structure to hold a variety of flags concerned with event signaling on messages.
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.
ForeignKeyInformation	String Notes: Read Only Returns the Foreign Key information.
IsLeaf	Boolean Notes: Read/Write A flag indicating that the connector is a leaf.
IsRoot	Boolean Notes: Read/Write A flag indicating that the connector is a root.
IsSpec	Boolean Notes: Read/Write A flag indicating that the connector is a specification.
MessageArguments	String Notes: Read Only

	The connector Message arguments.
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: • 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	Long Notes: Read/Write The y-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.
StateFlags	String Notes: Read/Write A structure to hold a variety of flags concerned with State signaling on messages; the list is delimited by semi-colons.
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 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	String Notes: Read/Write A possible subtype to refine the meaning of the connector.
SupplierEnd	ConnectorEnd Notes: Read Only A pointer to the ConnectorEnd object representing the target end of the relationship.
SupplierID	Long Notes: Read/Write The ElementID of the element at the target end of this connector.
TaggedValues	Collection of type ConnectorTag Notes: Read Only The collection of ConnectorTag objects.
TemplateBindings	Collection of type TemplateBinding Notes: Read Only A collection of TemplateBinding objects.

TransitionAction	String Notes: Read/Write See the *Transition* topic for appropriate values.
TransitionEvent	String Notes: Read/Write See the *Transition* topic for appropriate values.
TransitionGuard	String Notes: Read/Write See the *Transition* 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.
TypeInfoProperties	Notes: Read only Returns an interface pointer of TypeInfoProperties.
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.
GetTXAlias (string Code, long Flag)	String Notes: Returns the Alias of the element for a given language. Parameters • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long - 0 = Get the currently-stored translated Alias - 1 = Get the currently-stored translated Alias, and auto translate if the original Alias has changed

	- 2 = Always fetch the translated Alias from online
GetTXName (string Code, long Flag)	String Notes: Returns the name of the element for a given language. Parameters • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long - 0 = Get the currently-stored translated name - 1 = Get the currently-stored translated name, and auto translate if the original name has changed - 2 = Always fetch the translated name from online
GetTXNote (string Code, long Flag)	String Returns the Notes of the element for a given language. Parameters • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long - 0 = Get the currently-stored translated Notes - 1 = Get the currently-stored translated Notes, and auto translate if the original Notes have changed - 2 = Always fetch the translated Notes from online
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.
SetTXAlias (string Code, string Translation)	String Notes - Set the translated Alias of the element for a given language. • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated Alias
SetTXName (string Code, string Translation)	String Notes - Set the translated name of the element for a given language. • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated name
SetTXNote (string Code, string Translation)	String Notes - Set the translated Notes of the element for a given language. • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated Notes
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 repository

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 repository

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

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

	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 repository

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.

	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. When reading existing Tagged Values, if 'Value' = "<memo>" then the developer should read the actual body of text from the 'Notes' attribute.

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 repository

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.

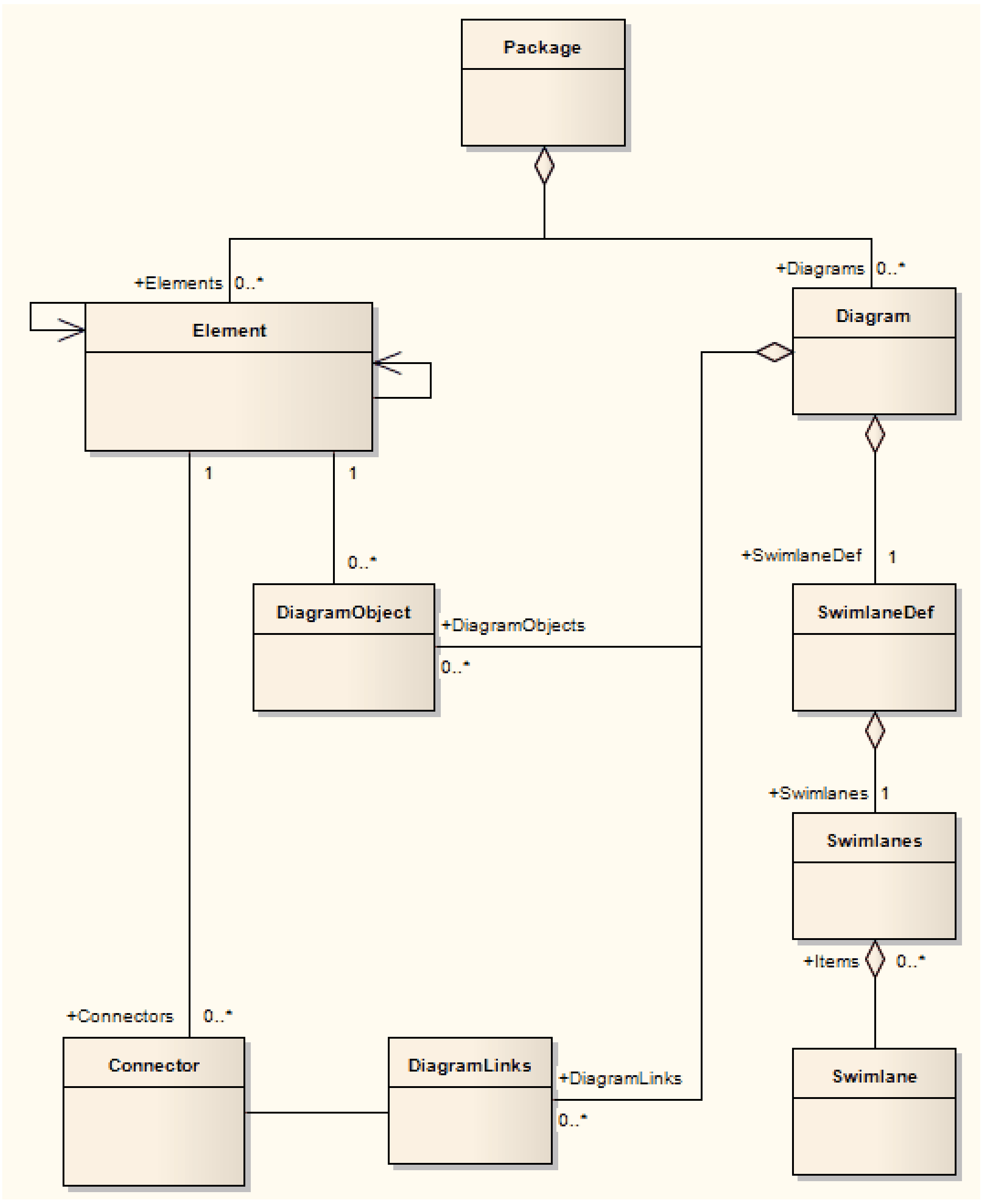


Diagram Class

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 repository

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

	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)	Notes: Applies a group lock to this diagram 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. Parameter: • aGroupName: String - the name of the user group for which to set the group lock
ApplyUserLock ()	Boolean Notes: Applies a user lock to this diagram 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.
FindElementInDiagram (long ElementID)	Boolean 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 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. Parameter • ElementID: Long - the element ID of the diagram object to locate
GetDiagramObjectByID (long ID, string DUID)	DiagramObject Notes: Returns the DiagramObject object, if it exists on the diagram. Parameters: • ID: Long - the ElementID of the diagram object • DUID: String - the optional Diagram Unique ID of the diagram object
GetElementByGrid (string GridX, string GridY)	Element Notes: Uses the Excel type of format to specify the column and row of a grid at which an element should be found: A 5, CB 1. Returns null if no element is at the specified position. Parameters: • GridX: string - Column A to Z • GridY: string - Number of row
GetElementByName (string Name)	Element Notes: Locates an element with the specified name. Returns null if no element is found with that name. Parameters: • name: string - Name of the element to find
GetObjectByGrid (string GridX, string GridY)	DiagramObject Notes: Uses the Excel type of format to specify the column and row of a grid at which an object should be found: A 5, CB 1. Returns null if no element is at the specified position.

	Parameters: • GridX: string - Column A to Z • GridY: string - Number of row
GetLastError ()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
ReadStyle (string StyleName)	String Notes: Returns the current value of the named diagram style. Use GetLastError() to retrieve error information. Parameters: • StyleName: String - the name of the diagram style whose value is to be retrieved; valid StyleNames are: - Show Element Property String - Show Connector Property String - Show Feature Property String
ReleaseUserLock ()	Boolean Notes: Releases a group lock or user lock on this diagram object. Returns True if the operation is successful; returns False if the operation is unsuccessful. Use GetLastError() to retrieve error information.
ReorderMessages ()	Void Notes: Resets the display order of Sequence and Collaboration messages. This is typically used after inserting or deleting messages in the diagram.
SaveAsPDF (string FileName)	Boolean Notes: Exports the diagram to a PDF document. Returns True on success. Parameters: • FileName: String - full path to file location
SaveImagePage(long x, long y, long sizeX, long sizeY, string filename, long flags)	Boolean Notes: Saves a page of the diagram to disk. Returns True if the operation is successful; returns False if the operation is unsuccessful. Use GetLastError() to retrieve error information. Parameters: • 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 The image type is determined by the extension of the filename. Currently only .emf, .bmp and .png formats are supported.

ShowAsElementList (bool ShowAsList, bool Persist)	Boolean Notes: Toggles the diagram display between diagram format and Diagram List depending on the value of ShowAsList. 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. Parameters: • ShowAsList: Boolean - indicates diagram or Diagram List • Persist: Boolean - indicates set (maintain ShowAsList value) or not (revert to default)
Update ()	Boolean Notes: Updates this diagram object after modification or appending a new item. If False is returned, use GetLastError() to retrieve error information.

VirtualizeConnector (int ConnectorID, int Action, int X, int Y)	Boolean
	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.
	If called again on the same connector, removes the virtual element. However, the waypoint remains in place.
	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.
	Returns True if the operation is successful; returns False if the operation is unsuccessful.
	Parameters:
	• 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 center point will be aligned with
	For example, to virtualize the source element of the selected connector:
	`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();
	```
WriteStyle (string StyleName, string StyleValue)	Void Notes: Sets the value of the named diagram style. Use GetLastError() to retrieve error information. Parameters: • StyleName: String - the name of the diagram style whose value is to be retrieved; valid StyleNames are: - Show Element Property String - Show Connector Property String - Show Feature Property String • StyleValue: String - the value to be set in the named diagram style; valid values for the StyleNames listed are **0** and **1**

DiagramLink 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 repository

t_diagramlinks

DiagramLink 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	Long Notes: Read/Write Returns the index of the line segment that has been suppressed. Returns 0 when no 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.

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

DiagramObject Class

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

Associated Table in Repository

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 the inherited requirements within the Requirements compartment 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 the requirements compartment for this object on the diagram.
ShowRunstates	Boolean Notes: Show Runstates for this object on the diagram.
ShowStructuredCompartments	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. See *Setting the Style*.
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.
MoveElementToGridPosition (GridX, GridY)	Notes: Currently not implemented.
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: • Parameter: String - the name of the style parameter to modify; for example: "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: • Title: String - The title text that 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
Delete(object Index)	Void Notes: Deletes a selected swimlane. If the string matches more than one entry, only the first entry is deleted. Parameter: • 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: • Index: Long - The zero-based index of the existing Swimlane before which this

	new entry is inserted • Title: String - The title text that 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)	Swimlane collection Notes: Accesses an individual swimlane. If the string matches more than one swimlane title, the first matching swimlane is returned. Parameter: • 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.

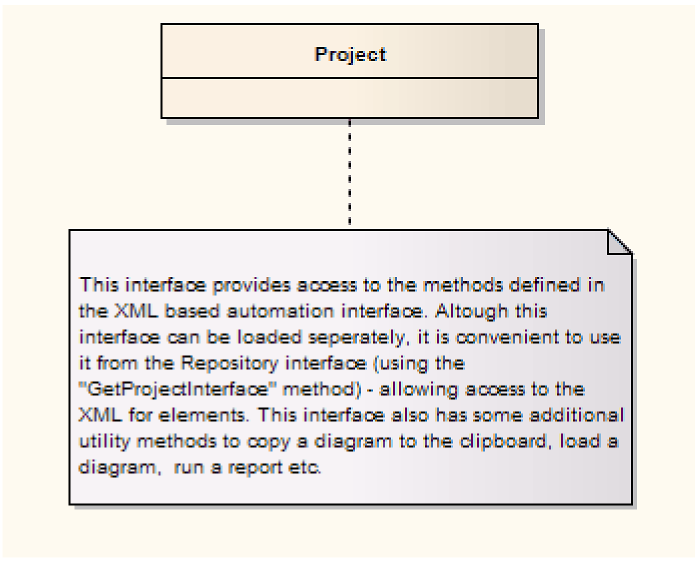
Swimlane Attributes

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
This interface provides access to the methods defined in the XML based automation interface. Altough this interface can be loaded seperately, it is convenient to use it from the Repository interface (using the "GetProjectInterface" method) - allowing access to the XML for elements. This interface also has some additional utility methods to copy a diagram to the clipboard, load a diagram, run a report etc.

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
BuildExecutableStatemachine (string ElementGUID, string ExtraOptions)	Boolean Notes: Builds Executable StateMachine code for an <<executable statemachine>> Artifact element. Parameters: • 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)
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: • 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: • PackageGUID: String - the GUID (in XML format) of the Package to be

	Baselined • 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
CreateSnapshot (string ItemGUID, string Notes, string ExtraOptions)	Boolean Notes: Creates Snapshot of a specified Package, Element or Diagram. Parameters: • ItemGUID: String - GUID (in XML format) of the Package/Element/Diagram whose Snapshot is to be created • Notes: String - any notes concerning the snapshot • ExtraOptions: String - enables extra options to be given to the command: - IncludeFullObjectFeatures=1/0 - applicable when creating Element/Diagram and specifies whether to include Element features (like Attributes and Operations) in the snapshot
DefineRule (string CategoryID, EA.EnumMVErrorType ErrorType, string ErrorMessage)	String Notes: Defines the individual rules that can be performed during model validation. It must be called once for each rule from the EA_OnInitializeUserRules broadcast handler. 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. Parameters: • 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: - 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)	String 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. The return value is a CategoryId that must to be passed to the DefineRule method. See the *Model Validation Example* for a detailed example of the use of this method. Parameters: • CategoryName: String - a text string that is visible in the 'Model Validation Configuration' dialog
DeleteBaseline (string BaselineGUID)	Boolean Notes: Deletes a Baseline, identified by the BaselineGUID, from the repository. If the repository is configured to store Baselines in a Reusable Asset Service Registry, then it is not possible to delete the Baseline and a False value is returned.

	Parameters: • BaselineGUID: String - the GUID (in XML format) of the Baseline to delete
DeleteSnapshot (string ItemGUID)	Boolean Notes: Deletes a Snapshot, identified by the SnapshotGUID, from the repository. Parameters: SnapshotGUID: String - GUID (in XML format) of the Snapshot to delete
DoBaselineCompare (string PackageGUID, string Baseline, string ConnectString)	String Notes: Performs a Baseline comparison using the supplied Package GUID and Baseline GUID (obtained in the result list from GetBaselines). Optionally you can include the connection string required to find the Baseline if it exists in a different model file. 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. Parameters: • 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 - not currently used
DoBaselineMerge (string PackageGUID, string Baseline, string MergeInstructions, string ConnectString)	String 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 located in another model. 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. <Merge> <MergeItem guid="{XXXXXX}" /> <MergeItem guid="{XXXXXX}" /> </Merge> Alternatively, you can supply a single Mergeitem with a GUID of RestoreAll. In this case, Enterprise Architect batch-processes ALL differences. <Merge> <MergeItem guid="RestoreAll" changed="true" baselineOnly="true" modelOnly="true" moved="true" fullRestore="false" /> </Merge> Parameters: • 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 - not currently used

DoFileCompare (string PackageGUID, string FileName, string ConnectString)	String Notes: Performs a Snapshot comparison using the supplied Package/Element/Diagram GUID and Snapshot GUID (obtained in the result list from GetSnapshots). Optionally you can include the connection string required to find the Snapshot if it exists in a different model file. This method returns a log file of the status of all elements found and compared in the difference procedure. Parameters: • ItemGUID: String - GUID (in XML format) of the Package/Element/Diagram to run the comparison on • SnapshotGUID: String - the GUID (in XML format) of the Snapshot to run the comparison on • ConnectString: String - not currently used
DoSnapshotCompare (string ItemGUID, string SnapshotGUID, string ConnectString)	String Notes: Performs a Snapshot comparison using the supplied Package/Element/Diagram GUID and Snapshot GUID (obtained in the result list from GetSnapshots). Optionally you can include the connection string required to find the Snapshot if it exists in a different model file. This method returns a log file of the status of all elements found and compared in the difference procedure. Parameters: • ItemGUID: String - GUID (in XML format) of the Package/Element/Diagram to run the comparison on • SnapshotGUID: String - the GUID (in XML format) of the Snapshot to run the comparison on • ConnectString: String - not currently used
DoTAMCompare (string ItemGUID, string Options, string ConnectString)	String Notes: Performs comparison of a Package/Element, identified by the supplied ItemGUID, against its TAM ancestor or clone. Optionally you can include the connection string required to find the Package/Element if it exists in a different model file. Parameters: • ItemGUID: String - GUID (in XML format) of the Package/Element to be compared • Options: String - specifies whether to compare Package/Element with its TAM ancestor or clone; accepted values are : 1 - compare with ancestor 2 - compare with clone <Clone GUID> - when a Package/Element contains more than one immediate clone, pass in GUID of one of these clones to compare with • ConnectString: String - not currently
EnumDiagramElements (string DiagramGUID)	protected abstract: String Notes: Gets an XML list of all elements in a diagram. Parameters: • DiagramGUID: String - the GUID (in XML format) of the diagram to get

	elements for
EnumDiagrams (string PackageGUID)	protected abstract: String Notes: Gets an XML list of all diagrams in a specified Package. Parameters: • PackageGUID: String - the GUID (in XML format) of the Package to list diagrams for
EnumElements (string PackageGUID)	protected abstract: String Notes: Gets an XML list of elements in a specified Package. Parameters: • PackageGUID: String - the GUID (in XML format) of the Package to get a list of elements for
EnumLinks (string ElementGUID)	protected abstract: String Notes: Gets an XML list of connectors for a specified element. Parameters: • 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: • 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.
EnumViewEx (string ProjectGUID)	protected abstract: String Notes: Gets a list of Views in the current project. Parameters: • ProjectGUID: String - the GUID (in XML format) of the project to get views for
EnumViews ()	protected abstract: String Notes: Enumerates the Views for a project. Returned as an XML document.
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, enumXMIType XMIType, long DiagramXML, long DiagramImage, long FormatXML, long	protected abstract: String Notes: Exports XMI for a specified Package. Parameters: • PackageGUID: String - the GUID (in XML format) of the Package to be exported

UseDTD, string FileName)	• XMIType: EnumXMIType - specifies the XMI type and version information; see *XMIType Enum* 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 alternative 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
ExportPackageXMIEx (string PackageGUID, enumXMIType XMIType, long DiagramXML, long DiagramImage, long FormatXML, long UseDTD, string FileName, ea.ExportPackageXMIFlag Flags)	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. Parameters: • PackageGUID: String - the GUID (in XML format) of the Package to be exported • XMIType: EnumXMIType - specifies the XMI type and version information; see *XMIType Enum* 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 alternative 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
ExportProjectXML (string DirectoryPath)	Boolean Notes: Exports the entire current project to Native XML files in the specified directory. The contents of the directory will be deleted prior to exporting the project data Parameters: • DirectoryPath: String - directory path to save the exported Native XML files
ExportReferenceData	

(string FileName, string Tables)	Boolean Notes: Exports Reference Data. Parameters: • FileName: String - the name of the file to output the reference data to • Tables: String - the list of reference data tables to be output; the data table delimeter is ";" If the string is empty, Enterprise Architect will prompt with a dialog to select the tables to output
GenerateBuildRunExecutableStateMachine (string ElementGUID, string ExtraOptions)	Boolean Notes: Generates, builds and runs Executable StateMachine code for an <<executable statemachine>> Artifact element, which will start simulation of the StateMachine. Parameters: • 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)
GenerateClass (string ElementGUID, string ExtraOptions)	Boolean Notes: Generates the code for a single Class. Parameters: • 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)	Boolean Notes: Generates various diagrams from the scenario specification of an element. Parameters: • ElementGUID: String - the GUID (in XML format) of the element containing the scenario specification • DiagramType: EnumScenarioDiagramType - the type of diagram to generate; see ScenarioDiagramType Enum 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)	Boolean Notes: Generates DDL for an element using the options that are currently set on the Generate DDL screen.
GenerateExecutableStatemachine (string ElementGUID, string ExtraOptions)	Boolean Notes: Generates Executable StateMachine code for an <<executable statemachine>> Artifact element. Parameters:

	• 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, pass an empty string to ensure current behavior in future versions)
GeneratePackage (string PackageGUID, string ExtraOptions)	Boolean Notes: Generates the code for all Classes within a Package. For example: recurse=1;overwrite=1;dir=C:\ Parameters: • 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: - 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)	Boolean Notes: Generates DDL for all elements in a Package using the options that are currently set on the Generate DDL screen.
GenerateTestFromScenario (string ElementGUID, EnumScenarioTestType TestType)	Boolean Notes: Generates a Vertical Test Suite, a Horizontal Test Suite, an Internal test or an External test from the scenario specification of an element. Parameters: • ElementGUID: String - the GUID (in XML format) of the element containing the scenario specification • TestType: EnumScenarioTestType - the type of test to generate; see *ScenarioTestType Enum* for accepted values
GenerateWSDL(string WSDLComponentGUID, string Filename, string Encoding, string ExtraOptions)	Boolean Notes: Generates WSDL for the specified WSDL stereotyped Component. Parameters: • 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)	Boolean Notes: Creates an XML schema for a Package, specified by its GUID. Returns True on success. Parameters: • 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: - 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 when referencing external Package, provided the schemaLocation tag is empty on the referenced Packages; for example: UseRelativePath=1
GetAllDiagramImagesAndMap (string Directory)	Boolean Notes : Saves the image and image-map for every diagram in the model, in the specified directory location. The image files will be saved in PNG format and each will have the diagram GUID as the image name. The image-map files will be saved as .txt files and each will have the diagram GUID as the image map name. The 'Auto Create Diagram Image and Image Map' option must be selected in the model options for this function to save the images and image-maps. Parameters: • Directory – the location of the directory into which the images and image-maps are to be saved
GetBaselines (string PackageGUID, string ConnectString)	String Notes: Returns a list (in XML format) of Baselines associated with the supplied Package GUID. Parameters: • PackageGUID: String - the GUID (in XML format) of the Package to get Baselines for • ConnectString: String - not currently used
GetDiagram (string DiagramGUID)	protected abstract: String Notes: Gets the diagram details, in XML format. Parameters: • DiagramGUID: String - the GUID (in XML format) of the diagram to get details for
GetDiagramImageAndMap (string DiagramGUID, string Directory)	Boolean Notes: Saves the image and image-map for the diagram with the specified GUID, in the specified directory location. The image will be saved in PNG format and will have the DiagramGUID as the image name. The image-map will be saved as a .txt file and will have the DiagramGUID as the image-map name. The 'Auto Create Diagram Image and Image Map' option must be selected in the model-specific options for this function to save the image and image-map. Parameters: • DiagramGUID – the GUID of the diagram for which the image and image-map are to be saved • Directory – the directory into which the image and image-map are to be saved
GetElement (string ElementGUID)	protected abstract: String Notes: Gets XML for the specified element.

	Parameters: • ElementGUID: String - the GUID (in XML format) of the element to retrieve XML for
GetElementConstraints (string ElementGUID)	protected abstract: String Notes: Gets constraints for an element, in XML format. Parameters: • ElementGUID: String - the GUID (in XML format) of the element
GetElementEffort (string ElementGUID)	protected abstract: String Notes: Gets efforts for an element, in XML format. Parameters: • ElementGUID: String - the GUID (in XML format) of the element
GetElementFiles (string ElementGUID)	protected abstract: String Notes: Gets metrics for an element, in XML format. Parameters: • ElementGUID: String - the GUID (in XML format) of the element
GetElementMetrics (string ElementGUID)	protected abstract: String Notes: Gets files for an element, in XML format. Parameters: • ElementGUID: String - the GUID (in XML format) of the element
GetElementProblems (string ElementGUID)	protected abstract: String Notes: Gets a list of issues (problems) associated with an element, in XML format. Parameters: • ElementGUID: String - the GUID (in XML format) of the element
GetElementProperties (string ElementGUID)	protected abstract: String Notes: Gets Tagged Values for an element, in XML format. Parameters: • ElementGUID: String - the GUID (in XML format) of the element
GetElementRequirements (string ElementGUID)	protected abstract: String Notes: Gets a list of requirements for an element, in XML format. Parameters: • ElementGUID: String -the GUID (in XML format) of the element
GetElementResources (string ElementGUID)	protected abstract: String Notes: Gets a list of resources for an element, in XML format. Parameters: • ElementGUID: String - the GUID (in XML format) of the element
GetElementRisks (string ElementGUID)	protected abstract: String Notes: Gets a list of risks associated with an element, in XML format. Parameters:

	• ElementGUID: String - the GUID (in XML format) of the element
GetElementScenarios (string ElementGUID)	protected abstract: String Notes: Gets a list of scenarios for an element, in XML format. Parameters: • ElementGUID: String - the GUID (in XML format) of the element
GetElementTests (string ElementGUID)	protected abstract: String Notes: Gets a list of tests for an element, in XML format. Parameters: • ElementGUID: String - the GUID (in XML format) of the element
GetFileNameDialog (string Filename, string FilterString, long FilterIndex, long Flags, string InitialDirectory, long OpenOrSave)	String 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. For example: 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) In this example, the 'Save As' dialog will prompt for a CSV file. Parameters: • 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 ()	protected abstract: String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
GetLink (string LinkGUID)	protected abstract: String Notes: Gets connector details, in XML format. Parameters: • LinkGUID: String - the GUID (in XML format) of the connector to get details of
GetSnapshots (string ItemGUID, string	String Notes: Returns a list (in XML format) of snapshots associated with the supplied

ConnectString)	Package, Element or Diagram GUID. Parameters: • ItemGUID: String - GUID (in XML format) of the Package/Element/Diagram to get snapshots for • ConnectString: String - not currently used
GUIDtoXML (string GUID)	String Notes: Changes an internal GUID to the form used in XML. Parameters: • GUID: String - the Enterprise Architect style GUID to convert to XML format
ImportDirectory (string PackageGUID, string Language, string DirectoryPath, string ExtraOptions)	Boolean Notes: Imports a source code directory into the model. Parameters: • 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)	Boolean Notes: Imports an individual file or binary module into the model, in a Package per namespace style import. Parameters: • 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)	String Notes: Imports an XMI file at a point in the tree. Returns an empty string if successful, or returns an error message on failure. Parameters: • 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 - 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
ImportReferenceData (string FileName, string DataSets)	Boolean Notes: Imports Reference Data. Parameters: • FileName: String - the name of the reference data file to import from • DataSets: String - the list of reference data sets to import from; the data set delimeter is ";" If the string is empty, Enterprise Architect displays a dialog prompt to select the data sets to import
ImportVisualStudioSolution (string PackageGUID, string SolutionPath)	Boolean Notes: Imports a Visual Studio Solution into the model. Parameters: • PackageGUID: string - the GUID (in XML format) of the Package to reverse engineer the solution into • SolutionPath: string - specifies the path of the Visual Studio Solution file on the computer
LayoutDiagram (string DiagramGUID, long LayoutStyle)	Boolean Notes: Deprecated. Use **LayoutDiagramEx**. Calls the function to automatically layout a diagram in hierarchical fashion. It is only recommended for Class and Object diagrams. Parameters: • 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)	Boolean Notes: Calls the function to automatically layout a diagram in hierarchical fashion. It is only recommended for Class and Object diagrams. LayoutStyle accepts these options • Default Options: - lsDiagramDefault - lsProgramDefault • Cycle Removal Options: - lsCycleRemoveGreedy - lsCycleRemoveDFS • Layering Options: - lsLayeringLongestPathSink - lsLayeringLongestPathSource - lsLayeringOptimalLinkLength • Initialize Options: - IsInitializeNaive - IsInitializeDFSOut

	- IsInitializeDFSIn • Crossing Reduction Option: - lsCrossReduceAggressive • Layout Options - Direction - lsLayoutDirectionUp - lsLayoutDirectionDown - lsLayoutDirectionLeft - lsLayoutDirectionRight Parameters: • 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)	String Notes: Loads a Package that has been marked and configured as controlled. The filename details are stored in the Package control data. Parameters: • PackageGUID: String - the GUID (in XML format) of the Package to load
LoadDiagram (string DiagramGUID)	protected abstract: Boolean Notes: Loads a diagram by its GUID. Parameter: • 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)	protected abstract: Boolean Notes: Loads an Enterprise Architect project file. Do not use this method if you have accessed the Project interface from the Repository, which has already loaded a file. Parameters: • FileName: String - the name of the project file to load
Migrate (string GUID, string SourceType, string DestinationType)	Void Notes: Migrates a model (or part of a model) from one BPMN, ArchiMate, UPDM or SysML format to an upgraded format. Parameters: • 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: - BPMN - BPMN1.1

	- UPDM - SysML1.1 - SysML1.2 - SysML1.3 - ArchiMate - ArchiMate2 - UPDM2 • DestinationType: String - the type of model to upgrade to; accepted values: - BPMN1.1 - BPMN1.1::BPEL - BPMN2.0 - UPDM2 - SysML1.2 - SysML1.3 - SysML1.4 - ArchiMate2 - ArchiMate3 - UAF
MigrateToBPMN11 (string GUID, string Type)	Void Notes: Migrates every BPMN 1.0 construct in a Package or an element (including elements, attributes, diagrams and connectors) to BPMN 1.1. Parameters • 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: - BPMN = migrate to BPMN 1.1 - BPEL = migrate to BPMN 1.1 and update: - any diagram with stereotype BPMN to BPEL - any element with stereotype BusinessProcess to BPELProcess Migrating to BPEL is possible in the Ultimate and Unified Editions of Enterprise Architect.
ProjectTransfer (string SourceFilePath, string TargetFilePath, string LogFilePath)	Boolean Notes: Transfers the project from a source .eap file or DBMS to a target .eap file, .eapx file, .feap file, .qea file or .qeax file. Parameters: • SourceFilePath: String - the path of the source file to transfer • TargetFilePath: String - the path of the target file, including the file type extension; Enterprise Architect creates a new Base project in this location (using the TargetFilePath as its name) and then transfers the content of the source project into that file • LogFilePath: String - the path of the log file where the status of the transfer process is updated In automation, the target file must not previously exist. Enterprise Architect creates a new, empty Base.* file using the specified target name and extension, and transfers the source project into it.
PublishResult (string CategoryID, EA.EnumMVErrorType ErrorType, string ErrorMessage)	String 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. 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. Parameters: • 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: - mvErrorCritical - mvError - mvWarning, or - mvInformation • ErrorMessage: String - contains an error string
PutDiagramImageOnClipboard (string DiagramGUID, long Type)	protected abstract: Boolean Notes: Copies an image of the specified diagram to the clipboard. Parameters: • DiagramGUID: String - the GUID (in XML format) of the diagram to copy • Type: Long - the file type - 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)	protected abstract: Boolean Notes: Saves an image of the specified diagram to file. Parameters: • 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 - 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 ()	protected abstract: Boolean Notes: Reloads the current project. This is a convenient method to refresh the current loaded project (in case of outside changes to the .eap file).
RunExecutableStatemachine (string ElementGUID, string ExtraOptions)	Boolean Notes: Runs Executable StateMachine code for an <<executable statemachine>> Artifact element, which will start simulation of the StateMachine Parameters: • 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)
RunModelSearch (string Search, string SearchTerm, bool ShowInEA)	Void Notes: Invokes the Model Search component. Parameters: • 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)	protected abstract: Void Notes: Runs a named document report. Parameters: • 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 PackageGUID, string ExportPath, string ImageFormat, string Style, string Extension)	String Notes: Runs an HTML report (as for 'Documentation | Publish as HTML' when you click on a Package in the Browser window and on the ☰ icon). Parameters: • 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)	String Notes: Saves a Package that has been configured as a controlled Package, to Native/XMI format. Only the Package GUID is required, Enterprise Architect picks the rest up from the Package control information. Parameter: • PackageGUID: String - the GUID (in XML format) of the Package to save
SaveDiagramImageToFile (string Filename)	protected abstract: String Notes: Saves a diagram image of the current diagram to file. Parameters: • FileName: String - the filename of the image to save
ShowWindow (long Show)	protected abstract: Void Notes: Shows or hides the Enterprise Architect User Interface. Parameters: • Show: Long
SynchronizeClass (string ElementGUID, string ExtraOptions)	Boolean Notes: Synchronizes a Class with the latest source code. Parameters: • 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)	Boolean Notes: Synchronizes each Class in a Package with the latest source code. Parameters: • 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, string ExtraOptions)	Boolean Notes: Transforms an element into a Package. Parameters: • TransformName: String - specifies the transformation that should be executed • 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: - 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)	Boolean Notes: Runs a transformation on the contents of a Package. Parameters: • 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: - GenCode=True/False - articulate code generation from the transformed elements; this option supercedes the current model setting - SubPackages=True/False - specify if the child Packages are to be included whilst transforming a Package
ValidateDiagram (string DiagramGUID)	Boolean Notes: Invokes the Enterprise Architect Model Validation component, then validates the diagram (for correctness) and the elements and connectors within the diagram. Output can be viewed through 'Start > Application > Design > System Output > Model Validation'. Returns a Boolean value to indicate the success or failure of the process, regardless of the results of the validation. Parameters: • DiagramGUID: String - the GUID of the Diagram Class object

ValidateElement (string ElementGUID)	Boolean Notes: Invokes the Enterprise Architect Model Validation component, then validates the element and all child elements, diagrams, connectors, attributes and operations. Output can be viewed through 'Start > Application > Design > System Output > Model Validation'. Returns a Boolean value to indicate the success or failure of the process, regardless of the results of the validation. Parameters: • ElementGUID: String - the GUID of the Element Class object
ValidatePackage (string PackageGUID)	Boolean Notes: Invokes the Enterprise Architect Model Validation component, then validates the Package and all sub-Packages, elements, connectors and diagrams within it. Output can be viewed through 'Start > Application > Design > System Output > Model Validation'. Returns a Boolean value to indicate the success or failure of the process, regardless of the results of the validation. Parameters: • PackageGUID: String - the GUID of the Package Class object
XMLtoGUID (string GUID)	String Notes: Changes a GUID in XML format to the form used inside Enterprise Architect. Parameters: • GUID: String - the XML style GUID to convert to Enterprise Architect internal format

Notes

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

Chart Package

The Chart interface can be used to dynamically construct any of the supported Chart types, using the functions provided in the Chart Package. The interface is obtained using the GetChart method on a Dynamic Chart element. A Dynamic Chart element can be created from the 'Charts' page of the Diagram Toolbox, and is typically used on a Dashboard diagram.

Chart Enumerations

These enumerations, used specifically by methods in the Chart interface, are described in the topics of this section. Click on the enumeration name in the list to the left of this text.

ChartAxisCrossType

Enum Values

Enum	Value
Auto	value: 0
MaximumAxisValue	value: 1
MinimumAxisValue	value: 2
AxisValue	value : 3
Ignore	value: 4
FixedDefaultPos	value: 5

ChartAxisIndex

Enum Values

Enum	Value
Unknown	value: -1
X	value: 0
Y	value: 1
Z	value: 2

ChartAxisLabelType

Enum Values

Enum	Value
NoLabels	value: 0
NextToAxis	value: 1
High	value: 2
Low	value: 3

ChartAxisTickMarkType

Enum Values

Enum	Value
NoTicks	value: 0
Inside	value: 1
Outside	value: 2
Cross	value: 3

ChartAxisType

A set of constants that refer to the various axes used in charts.

Enum Values

Enum	Value
CHART_Y_PRIMARY_AXIS	value: 0
CHART_Y_SECONDARY_AXIS	value: 1
CHART_X_PRIMARY_AXIS	value: 2
CHART_X_SECONDARY_AXIS	value: 3
CHART_Z_PRIMARY_AXIS	value: 4
CHART_Z_SECONDARY_AXIS	value: 5
CHART_Y_POLAR_AXIS	value: 6
CHART_X_POLAR_AXIS	value: 7
CHART_A_TERNARY_AXIS	value: 8
CHART_B_TERNARY_AXIS	value: 9
CHART_C_TERNARY_AXIS	value: 10

ChartBarShape

Enum Values

Enum	Value
Box	value: 0
Pyramid	value: 1
PyramidPartial	value: 2

ChartCategory

Enum Values

Enum	Value
chartDefault	value: 0
chartLine	value: 1
chartPie	value: 2
chartPie3D	value: 3
chartPyramid	value: 4
chartPyramid3D	value: 5
chartFunnel	value: 6
chartFunnel3D	value: 7
chartColumn	value: 8
chartBar	value: 9
chartHistogram	value: 10
chartArea	value: 11
chartStock	value: 12
chartBubble	value: 13
chartLongData	value: 14
chartHistoricalLine	value: 15
chartPolar	value: 16
chartDoughnut	value: 17
chartDoughnut3D	value: 18
chartTorus3D	value: 19
chartTernary	value: 20
chartColumn3D	value: 21

chartBar3D	value: 22
chartLine3D	value: 23
chartArea3D	value: 24
chartSurface3D	value: 25
chartDoughnutNested	value: 26
chartBoxPlot	value: 27
chartBarSmart	value: 28
chartBar3DSmart	value: 29

ChartColorMode

Enum Values

Enum	Values
Single	value: 0
Multiple	value: 1
Palette	value: 2
Custom	value: 3
Series	value: 4

ChartCurveType

Enum Values

Enum	Value
NoLine	value: 0
Line	value: 1
Spline	value: 2
SplineHermite	value: 3
Step	value: 4
ReversedStep	value: 5

ChartDashStyle

Enum Values

Enum	Value
Solid	value: 0
Dash	value: 1
Dot	value: 2
DashDot	value: 3
DashDotDot	value: 4
Custom	value: 5

ChartFrameStyle

Enum Values

Enum	Value
None	value: 0
Mesh	value: 1
Contour	value: 2
ContourMesh	value: 3

ChartGradientType

Enum Values

Enum	Value
None	value: 0
Horizontal	value: 1
Vertical	value: 2
DiagonalLeft	value: 3
DiagonalRight	value: 4
CenterHorizontal	value: 5
CenterVertical	value: 6
RadialTop	value: 7
RadialCenter	value: 8
RadialBottom	value: 9
RadialLeft	value: 10
RadialRight	value: 11
RadialTopLeft	value: 12
RadialTopRight	value: 13
RadialBottomLeft	value: 14
RadialBottomRight	value: 15
Bevel	value: 16
PipeVertical	value: 17
PipeHorizontal	value : 18

ChartMarkerShape

Enum Values

Enum	Value
Circle	value: 0
Triangle	value: 1
Rectangle	value: 2
Rhombus	value: 3

ChartStockSeriesType

Enum Values

Enum	Value
Bar	value: 0
Candle	value: 1
LineOpen	value: 2
LineHigh	value: 3
LineLow	value: 4
LineClose	value: 5
LineCustom	value: 6

ChartType

Enum Values

Enum	Value
chartTypeDEFAULT	value: 0
chartTypeSIMPLE	value: 1
chartTypeSTACKED	value: 2
chartType100STACKED	value: 3
chartTypeRANGE	value: 4

ChartWallOptions

Enum Values

Enum	Value
None	value: 0, 0x0000
FillLeftWall	value: 1, 0x0001
OutlineLeftWall	value: 2, 0x0002
FillRightWall	value: 4, 0x0004
OutlineRightWall	value: 8, 0x0008
FillFloor	value: 16, 0x0010
OutlineFloor	value: 32, 0x0020
DrawAll	value: 65535, 0xFFFF
DrawLeftWall	FillLeftWall | OutlineLeftWall
DrawRightWall	FillRightWall | OutlineRightWall
DrawFloor	FillFloor | OutlineFloor
DrawAllWalls	DrawLeftWall | DrawRightWall
OutlineAllWalls	OutlineLeftWall | OutlineRightWall
OutlineAll	OutlineAllWalls | OutlineFloor
FillAllWalls	FillLeftWall | FillRightWall
FillAll	FillAllWalls | FillFloor
Default	OutlineAll

Chart Class

The Chart Class is the primary interface for Chart elements; it is used to create a series, add datapoints to a series and configure the chart appearance.

Chart Attributes

Attribute	Description
Title	String Notes: Read/Write The title of the chart.
Category	ChartCategory Notes: Read only The chart category; provided in the SetChartType method.
Type	ChartType Notes: Read only The chart type; provided in the SetChartType method.

Chart Methods

Method	Description
AddChartDataYXZ(double Y, double X, double Z, long seriesIndex)	long Adds a datapoint to an existing series. Parameters: • Y: double, the primary Y axis value • X: double, the primary X axis value • Z: double, the primary Z axis value • seriesIndex: long, the index of the series (returned by the CreateSeries methods)
AddChartDataYY1(string category, double Y, double Y1, long seriesIndex)	long Adds a datapoint to an existing series. Parameters: • category: string - the x axis group, column or label • Y: double, the primary Y axis value • Y1: double, the secondary Y axis value • seriesIndex: long, the index of the series (returned by the CreateSeries and CreateSeriesEx methods)

CreateSeries(string name)	LDISPATCH Adds a new series to the chart. Returns an interface that can be used to add data to the series and configure its appearance. Parameters: • name: string, the displayed name of the series
CreateSeriesEx(string name, long color, ChartType type, ChartCategory category)	LDISPATCH Creates a series of a particular chart category and type and returns an IChartSeries interface. This allows charts to form multiple series in various ways. The CombinedCharts in the EAExample Model are an example of this, displaying three series for the Area, Column and Line categories respectively. Parameters: • name: string, the name of the series • color: long, RGB color value,-1 for default • type: ChartType, one of the ChartType enumerations • category: ChartCategory, one of the ChartCategory enumerations
EnableResizeAxes(boolean bEnable)	void Grants or denies the ability to resize axes. Parameters: • bEnable: boolean
GetChartAxis(ChartAxisType type)	LDISPATCH Returns an IChartAxis interface for the specified axis. Parameters: • type: ChartAxisType, one of the ChartAxisType enumerations
GetDiagram3D()	LDISPATCH Returns an IChartDiagram interface that can be used to specify the rendering engine; Software or OpenGL.
GetSeries(long index)	LDISPATCH Returns an IChartSeries interface for the given index. Indices for series begin at zero. Parameters: • index: long
GetSeriesCount()	long Returns the number of series represented by the chart.
Redraw()	void Redraws the chart.
SetChartType(ChartType type, ChartCategory category, boolean bRedraw, boolean bResizeAxis)	void This is typically the first call made on the Chart interface. It defines the style and appearance of the Chart when it is rendered. Parameters: • type: ChartType

	• category: ChartCategory • bRedraw: Boolean • bResizeAxis: Boolean
SetCurveType(ChartCurve Type type)	void Sets the interpolation method of drawing the curve. Parameters: • type: ChartCurveType, one of the ChartCurveType enumerations, such as Line, Spline or SplineHermite.
SetSeriesShadow(boolean bShow)	void Displays or hides shadows on series. Parameters: • bShow: Boolean
SetThemeOpacity(long percentage)	void Sets the opacity of the chart. Parameters: • percentage: long, chart transparency as a percentage
ShowAxis(long index)	void Shows the axis for the given index. Parameters: • index: long, one of the ChartAxisType enumerations
ShowDataLabels(boolean show, boolean border, boolean dropLineTomarker)	void Shows or hides data labels on the chart. Parameters: • show: Boolean, show or hide labels • border: Boolean, show or hide border on labels • dropLineTomarker: Boolean, changes position of label with respect to line
ShowDataMarkers(boolean show, long size, ChartmarkerShape shape)	void Shows or hides data markers on the chart. Also allows setting the appearance of markers. Parameters: • show: Boolean, show or hide markers • size: long, size of markers in pixels • shape: ChartmarkerShape, one of the ChartmarkerShape enumerations

ChartAxisIndex Class

ChartAxisIndex Attributes

Attribute	Description
Visible	Boolean Shows or hides the axis.

ChartAxisIndex Methods

Method	Description
EnableMajorUnitIntervalInterlacing(boolean binterlace)	void Turns interlacing on or off.
GetGuid()	string Returns the guid of the axis. Uniquely identifies an axis.
GetLabel()	string Returns the value of the label of the axis.
SetAxisName(string label, boolean showonaxis)	void Sets the label for the axis and whether it should be displayed on the chart. Parameters: • label: string, the text for the label • showonaxis: Boolean, a true value indicates that the label is displayed
SetCrossType(long type)	void Provides a directive or hint for use when calculating the position of labels on an axis. Parameters: • type: long, one of the ChartAxisCrossType enumerations
SetDataFormat(string format, boolean formatAsDate)	void Sets the format string for the conversion of values to strings (e.g. "%.4f"). If the datapoints represent datetime values, the formatAsDate argument should be true, and the format string set appropriately (e.g. "%H:%M") Parameters: • format: string, the format to use when converting datapoint values to string • formatAsDate: Boolean, a true value indicates the datapoint represent a datetime
SetDisplayUnits(double	

units)	void Sets the display units on the axis. Basically, the datapoint values are divided by this figure to give a major unit value. For example, if the datapoint contains meter values, a value of 1000 would result in kilometers being used as the major unit on the axis. Parameters: • units: double, the value of a single unit on the axis
SetFixedDisplayRange(double fmin, double fmax)	void Sets a fixed range for the axis. Parameters: • fmin: double, the minimum value • fmax: double, the maximum value
SetLabelType(long labelpos)	void Sets the position of labels on the axis. Parameters: • labelpos: long, one of the ChartAxisLabelType enumerations
SetTickMark(long tickmarkpos)	void Sets the position of tick marks on the axis. Parameters: • tickmarkpos: long, one of the ChartAxisTickMarkType enumerations
ShowMajorGridLines(boolean show)	void Shows or hides grid lines.

ChartDataValue Class

The ChartDataValue class provides an interface that allows values to be obtained from points in a series.

ChartDataValue Methods

Method	Description
GetValue()	double Returns the value associated with the datapoint.
IsEmpty()	Boolean True if no value exists for the datapoint.
SetEmpty(boolean empty)	void Sets a datapoint on a series to be empty. Parameters: • empty: Boolean, true if the datapoint is to be considered as empty, having no value
SetValue(double value)	void Sets the value of a datapoint. Parameters: • value: double, the value of the datapoint; setting a value makes a datapoint non-empty

ChartDiagram3D Class

ChartDiagram3D Methods

Method	Description
SetDrawWallOptions(long options, boolean redraw)	void Sets the option for how walls and floors - if any - are displayed on the 3D chart. The options parameter is a bitmask of one or more values from the ChartWallOptions enum. Parameters: • options: Long, bitmask of wall and floor display options • redraw: Boolean, redraws the chart after the function completes
SetRenderingType(long engine)	void Parameters: • engine: long, 0 for software,1 for openGL

ChartFormatSeries Class

A helper class for the ChartSeries class that allows setting appearance options.

ChartFormatSeries Methods

Method	Description
SetCurveType(ChartCurve Type type)	void Sets the graphic option for rendering lines. Parameters: • type: long, one of the ChartCurveType enumerations
SetSeriesLineWidth(long width)	void Sets the line width in pixels. Parameters: • width: long, a pixel value
SetSeriesOutlineDashStyle (ChartDashStyle dashstyle)	void Sets the dash style of the line on the chart/graph. Parameters: • dashstyle: ChartDashStyle, one of the ChartDashStyle enumerations

ChartSeries Class

ChartSeries Methods

Method	Description
AddBoxPlotData(double ave, double min, double q1, double q2, double q3, double max, double notched)	long For a chart having the BoxPlot category, adds a single datapoint to the series. Parameters: • ave: double, the mean value at this point • min: double, the minimum value at this point • q1: double, the first quartile value • q2: double, the second quartile value • q3: double, the third quartile value • max: double the maximum value at this point • notched: double, for a series with notched style, the notched value to express at this point
AddDataPoint(double Y)	long Adds a datapoint to the series. Returns the index of the point, which is the number of points -1. Parameters: • Y: double, the Y axis value
AddDataPoint2(double Y, double X)	long Adds a datapoint to the series. Returns the index of the point, which is the number of points -1. Parameters: • Y: double, the Y axis value • X: double, the X axis value
AddDataPoint3(string category, double Y)	long Adds a Y axis value for a given category on the X axis. Parameters: • category: string, the category or column name • Y: double, the value
AddStockData(double open, double high, double low, double closing, VARIANT timestamp)	void Adds data to a series for a chart of the Stock category. Parameters: • open: double, opening value • high: double, high value • low: double, low value • closing: double, closing value • timestamp: {datetime, double utcsecs} either VARIANT date value or double, in which case the value is interpreted as the number of seconds since midnight

	on January 1st, 1970, UTC time
AddSurfaceColors(VARIANT colors)	void Adds one or more colors to the series. Parameters: • colors: long, or long[], a single RGB color or an array of RGB color values
CloseShape(boolean close, boolean fill)	void Connects the first and last datapoints and fills the shape if 'fill' is true. Parameters • close: Boolean, if true closes the series • fill: Boolean, fills the shape
GetDataPointCount()	long Returns the number of datapoints in the series.
GetDataPointValue(long index)	LDISPATCH Returns a ChartDataValue interface for the datapoint with the given index. Parameters: • index: long, the index of the datapoint (typically returned by AddDataPoint functions; a value in the range 0 to n-1, where n is the number of points returned by the *GetDataPointCount* function)
GetSeriesFormat()	LDISPATCH Returns a ChartFormatSeries interface that allows the chart appearance to be changed.
SetBarShape(long barshape)	void Sets the shape for Bar charts, 0 for Box, 1 for Pyramid, 2 for PyramidPartial. Parameters: • barshape: ChartBarShape, one of the ChartBarShape enumerations
SetColorMapCount(long count)	void Sets the number of colors used when rendering the series. Typical values are 4, 8, 16 and 32 Parameters: • count: long, the number of colors to use
SetColorMode(ChartColorMode mode)	void For 3D charts, sets the interpolation method for filling shapes. Single, for example, would result in the 3D object being filled by varying the color slightly. The level of variation will depend on the number of colors used by the chart (see *SetColorMapCount*). Parameters: • mode: ChartColorMode
SetDrawFlat(boolean flat)	void Draws the shape flattened when set to true. Parameters:

	• flat: Boolean, draw flat
SetFrameColor(long color)	void Sets the color of the frame for 3D objects. Parameters: • color: long, the RGB color value for coloring the frame
SetFrameStyle(ChartFrame Style style)	void Sets the frame style for the chart - none, mesh, contour or both. Parameters: • style: ChartFrameStyle, one of the ChartFrameStyle enumerations
SetGradientType(long type)	void Sets the gradient type to use. Parameters: • type: long, one of the ChartGradientType enumerations
SetGroupID(long id)	void Groups series on a stacked chart having the same id. Must be a non-negative number. Parameters: • id: long, a non-negative number used to group the series on a chart
SetLevelRangeMode(long mode)	void Sets the mode for ranges in series. • 0 - Minimum and maximum for Series • 1 - Minimum and maximum for Y axis • 2 - Custom Parameters: • mode: long, either 0 or 1 supported
SetRelatedAxis(string axis, long index)	void Sets the related axis for a series. The related axis is created using the Split function of the ChartAxis interface. The axis is first created using Split, then a new series is created, and this function called on it to one of its axes. The axis is specified by the index parameter; the value is one of the ChartAxisIndex enumerations (0 for X, 1 for Y or 2 for Z) Parameters: • axis: string, the guid of the axis returned by a ChartAxis.Split method call; returned by the ChartAxis.GetGuid method • index: long, one of the ChartAxisIndex enumerations
SetStockSeriesType(Chart StockSeriesType type)	void For Stock charts, sets the graphic used to render the series. Parameters: • type: ChartStockSeriesType, one of the ChartStockSeriesType enumerations
SetWireFrame(boolean	void

wired)	Sets the wireframe option on or off. When set to true, the chart is no longer rendered as a solid object but is instead rendered as a frame composed of wires. Parameters: • wired: Boolean, displays as a wireframe object if true

Document Generator Interface Package

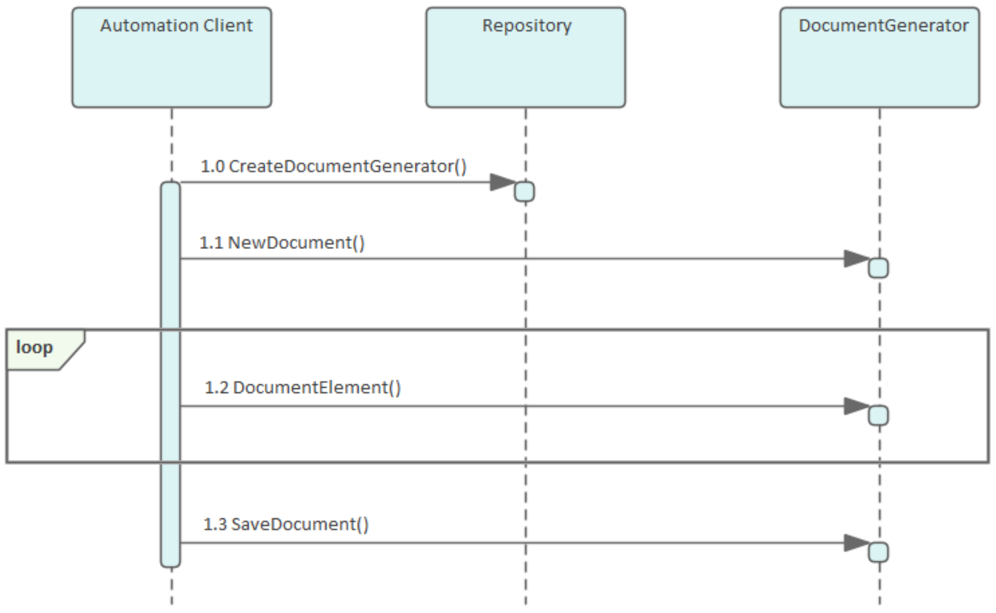
The DocumentGenerator Class provides an interface to the document and web reporting facilities, which you can use to generate reports on specific Packages, diagrams and elements in your model.

Access

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

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 ('Specialize > Tools > Script Library', 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: • connectorId: Long - the ID of the connector • nDepth: Long - the depth by which to adjust the heading level • templateName: 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: • XML: String - the XML of the data to be documented • nDepth: Long - the depth by which to adjust the heading level • templateName: 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: • diagramId: Long - the ID of the diagram • nDepth: Long - the depth by which to adjust the heading level • templateName: String - the name of a template to use when documenting diagrams; this can be blank
DocumentElement (long elementID, long nDepth,	Boolean

string templateName)	Notes: Documents an element. Parameters: • 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)	Boolean Notes: Documents a model author. Parameters: • 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)	Boolean Notes: Documents a single model client. Parameters: • 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)	Boolean Notes: Documents a single model glossary term. Parameters: • 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)	Boolean Notes: Documents a single model issue. Parameters: • 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)	Boolean Notes: Documents a single model resource. Parameters: • 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	Boolean

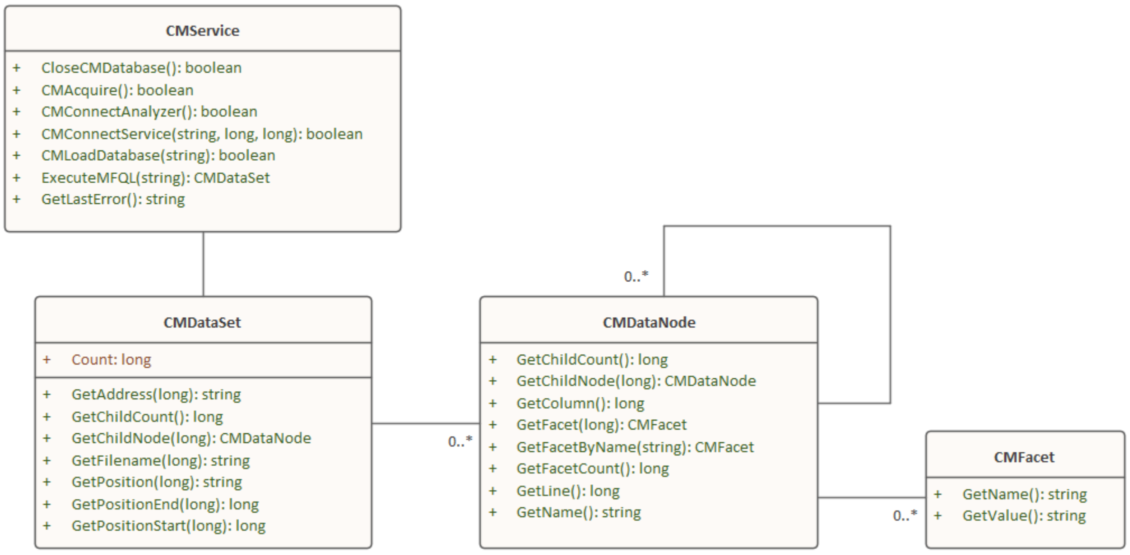
(string name, long nDepth, string templateName)	Notes: Documents a single model role. Parameters: • name: String - the name of the role • nDepth: Long - the depth by which to adjust the heading level • templateName: String - a template to use when documenting model roles; this can be blank
DocumentModelTask (long id, long nDepth, string templateName)	Boolean Notes: Documents a single model task. Parameters: • 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)	Boolean Notes: Documents a Package. Parameters: • 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()	Read Only. Returns a string value of the document in raw Rich Text Format.
GetProjectConstant (string nameVal)	String Notes: Returns the value of a Project Constant. Parameters: • nameVal: String - the name of the Project Constant for which to extract the value.
GetLastError ()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
InsertBreak (long breakType)	Boolean Notes: Inserts a break into the report at the current location. Parameters: • breakType: Long - 0 = page break, 1 = section break
InsertCoverPageDocument (string Name)	Boolean Notes: Inserts the Coverpage into the document at the current location. The style sheet is applied to the document before it is insert into the generated document. Parameters: • Name: String - the name of the Cover page document found in the Resource tree

InsertHyperlink (string Name, string URL)	Boolean Notes: Inserts a hyperlink at the current location. If you use a URL with the #BOOKMARKNAME syntax, the hyperlink will link to another part of the document. Parameters: • Name: String - the link text to insert into the report • URL: String - The URL of the website to link to
InsertLinkedDocument (string guid)	Boolean Notes: Inserts a Linked Document into the report at the current location. 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. Parameters: • guid: String - the GUID of the element that has a Linked Document
InsertTableOfContents	Boolean Notes: Inserts a Table of Contents at the current position.
InsertTeamReviewPost (string path)	Boolean Notes: Inserts a Model Library posting into the report at the current location. Parameters: • path: String - the path of the Model Library post
InsertTemplate (string templateName)	Notes: Inserts the contents of the template directly into the report. Parameters: • templateName: String - the name of the template to use
InsertText (string text, string style)	Boolean Notes: Inserts static text into the report at the current location. A carriage return is not included; if you need to use one, you can add it manually. Parameters: • 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)	Boolean Notes: Inserts the Table of Contents into the document at the current location. Note: The stylesheet is applied to the document before it is insert into the generated document. Parameters: • name: String - the name of the Table of Contents document found in the Resource tree
LoadDocument(string FileName)	Boolean Notes: Inserts an external document into the currently generated file. Parameters: • FileName: String - the filename of an external document file to insert into the document.

NewDocument (string templateName)	Boolean Notes: Starts a new document; you call this before attempting to document anything else. Parameters: • templateName: String - the name of a template to use when documenting elements; this can be blank
ReplaceField (string fieldname, string fieldvalue)	Boolean Notes: Replaces the 'Section' field identified by the fieldname parameter with the value provided in fieldvalue. For example: ReplaceField ("Element.Alias", "MyAlias") If you call this function more than once with the same fieldname, the field only has the most recent value set. Parameters: • 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)	Boolean Notes: Saves the document to disk. Parameters: • filename: String - the filename to save the file to • nDocType: Long - 0 = RTF, 1 = HTML, 2 = PDF, 3 = DOCX
SetPageOrientation (long pageOrientation)	Boolean Notes: Sets the current page orientation. Parameters: • pageOrientation: Long - 0 = Portrait, 1 = Landscape
SetProjectConstant (string newNameVal, string newValue)	Boolean Notes: Sets a Project Constant for the documentation generator; this is saved in the current model. Parameters: • newNameVal: String - the name of the Project Constant • newValue: String - the value of the Project Constant
SetStyleSheetDocument (string name)	Boolean Notes: Sets the Stylesheet to be used for TOC, Coverpage and templates used. This can be called before NewDocument. Parameters: • name: String - the name of the stylesheet found in the Resource tree
SetSuppressProfile (name)	Boolean Notes: Sets the Suppress Profile to be used during report generation. Parameters: • Name: String - The name of the Suppress Profile, as created on the 'Suppress Sections' tab of the 'Document Generation' dialog.

Code Miner Package

The Code Miner Package provides the Automation Interface to the Code Miner elements. It contains these classes:


CMService
+ CloseCMDatabase(): boolean
+ CMAcquire(): boolean
+ CMConnectAnalyzer(): boolean
+ CMConnectService(string, long, long): boolean
+ CMLoadDatabase(string): boolean
+ ExecuteMFQL(string): CMDataSet
+ GetLastError(): string
CMDataSet
+ Count: long
+ GetAddress(long): string
+ GetChildCount(): long
+ GetChildNode(long): CMDataNode
+ GetFilename(long): string
+ GetPosition(long): string
+ GetPositionEnd(long): long
+ GetPositionStart(long): long
0..*
CMDataNode
0..*
+ GetChildCount(): long
+ GetChildNode(long): CMDataNode
+ GetColumn(): long
+ GetFacet(long): CMFacet
+ GetFacetByName(string): CMFacet
+ GetFacetCount(): long
+ GetLine(): long
+ GetName(): string
0..*
CMFacet
+ GetName(): string
+ GetValue(): string

For an overview of using the Code Miner, see the Help topic *Code Miner Framework* under Software Engineering.

Interface CMService

The **CMService** is the primary interface object, used to access Code Miner data. It allows you to connect to local files as well as local or remote code miner services, and supports execution of mFQL queries against Code Miner databases.

CMService Methods

Method	Remarks
GetLastError( )	Returns a string value describing the most recent error that occurred in relation to this object, or null string if there was no error. Return type: string (read only)
CMLoadDatabase(string filename)	Load a Code Miner database from a file. Returns True on success; check GetLastError on failure. Return type: boolean (read only) Parameters: • filename: Full path to Code Miner database file.
CMConnectService(string hostname, long port, long dbID)	Connect to a Code Miner service. Returns True on success; check GetLastError on failure. Note: To query all code miner databases, specify dbID = -1. Return type: boolean (read only) Parameters: • hostname: the computer name to connect to • port: port number the service is listening to • dbID: Database ID to use.
CMConnectAnalyzer()	Connect to the default Code Miner database. Returns True on success; check GetLastError on failure. Return type: boolean (read only)
CMAcquire()	Currently not implemented. Return type: boolean
ExecuteMFQL(string Query)	Returns a CMDataSet object on success; check GetLastError on failure. Return type: CMDataSet (read only) Parameter: • query: An mFQL query string
CloseCMDatabase()	Close the connection to the Code Miner service or database file. Returns True on success; check GetLastError on failure. Return type: boolean (read only)

Interface CMDataSet

The **CMDataSet** is used to return the results of executing an mFQL query against a Code Miner library.

The results are returned as binary data in a CMDataSet, which consists of an array of CMDataNode objects. The CMDataNode objects, details of which can be found in a separate topic, contain detailed information of a **'match'** for the search query. Information related to each CMDataSet node, such as the name of the source code file, the line number and also the start and end position within the source code file, where the match occurred, can be obtained through the CMDataSet functions set out below.

All data within the CMDataSet is read-only.

CMDataSet Attributes

Attributes	Remarks
Count	The number of items in the dataset. Type: long (read only)

CMDataSet Methods

Method	Remarks
GetFilename(long index)	Returns the name of the source code file in which the matching element was found. Return type: string (read only) Parameters: • index: specifies the array position of the child node for which the Filename is being retrieved. (0 based index.)
GetPosition(long index)	Returns the position within the source code file, at which the matching element was found. The returned value has the format: "<startPosition>:<endPosition>" Return type: string (read only) Parameters: • index: specifies the array position of the child node for which the position information is being retrieved. (0 based index.)
GetAddress(long index)	A Code Miner database, consists of a list of abstract syntax tree nodes, where the primary key of each node is its address. This function operates on a list of AST nodes and returns the primary key (address) of the nth entry in that list. Return type: string (read only) Parameters: • index: specifies the array position of the child node for which the address information is being retrieved. (0 based index.)
GetPositionStart(long index)	Returns the start position within the source code file, at which the matching element was found. Return type: long (read only)

	Parameters: • index: specifies the array position of the child node for which the start position information is being retrieved. (0 based index.)
GetPositionEnd(long index)	Returns the end position within the source code file, at which the matching element was found. Return type: long (read only) Parameters: • index: specifies the array position of the child node for which the end position information is being retrieved. (0 based index.)
GetChildCount()	Returns count of number of CMDataNode child nodes in this CMDataSet. Return type: long (read only)
GetChildNode(long index)	Returns the CMDataNode object at the specified index position. Return type: CMDataNode (read only) Parameters: • index: specifies the array position of the child node to be retrieved. (0 based index.)

Interface CMDataNode

The **CMDataNode** objects contain detailed information relating to each item matching the executed mFQL query.

Each CMDataNode contains the name of the matching source code element, along with the line number and column number within the source code file where the matching element was found. Each CMDataNode also contains its own array of CMDataNode objects as well as an array of CMFacet objects. The CMFacet objects are detailed in a separate topic.

CMDataNode Methods

Method	Remarks
GetName()	Returns the name of the CMDataNode Return type: string (read only)
GetLine()	Returns the line on which this node can be found. Return type: long (read only)
GetColumn()	Returns the column on which this node can be found. Return type: long (read only)
GetChildCount()	Returns the number of CMDataNode child nodes belonging to the current CMDataNode. Return type: long (read only)
GetChildNode(long index)	Returns the CMDataNode object found at the nth position in the array of CMDataNode child nodes, or NULL if not found. Return type: CMDataNode (read only) Parameters: • index: specifies the array position of the child node to be retrieved. (0 based index)
GetFacetCount()	Returns the number of CMFacet objects that are part of the current node. Return type: long (read only)
GetFacet(long index)	Returns the CMFacet object found at the nth position in the array of CMFacet child nodes, or NULL if not found. Return type: CMFacet (read only) Parameters: • index: specifies the array position of the facet node to be retrieved. (0 based index)
GetFacetByName(string name)	Returns the named CMFacet object, or NULL if not found. Return type: CMFacet (read only) Parameters: • name: specifies the name of facet to find.

Interface CMFacet

Each **CMDataNode** holds of array of **CMFacet** objects. The CMFacet objects are Name/Value data pairs.

The Name data and Value data that is present, varies depending on the particular CMDataNode being accessed, and of course on the original mFQL query that was executed.

CMFacet Methods

Method	Remarks
GetName()	Returns the name of the current CMFacet node. Return type: string (read only)
GetValue()	Returns the value of the current CMFacet node. Return type: string (read only)

Code Miner Example Script

The following example script is designed to demonstrate the use all of the methods available for all of the Code Miner API interface objects.

Example Script

```
!INC Local Scripts.EAConstants-JScript
/*
* Script Name:            CodeMiner API Test
* Author:                                 Sparx Systems
* Purpose:                     Exercise all of the functions available within the Code Miner API
* Date:                                      May 2024
*/
var q1 = "{\
                byItem(\"OPERATION\",\"NAME\",\"LoadBinary\"),\
                byItem(\"OPERATION\",\"NAMESPACE\",\"C2DShapeEditDlg\")\
}";
function PrintFacetInfo(indent, facet, k)
{
                indent = indent + "    ";
                // print out the Name/Value data pair, for the specified CMFacet object
                Session.Output(indent + "---  Facet " + k + " Info  ---");
                Session.Output(indent + "Facet name: " + facet.GetName() );
                Session.Output(indent + "Facet value: " + facet.GetValue() );
                Session.Output("   ");
}
function PrintDataNode(indent, dataNode, n, level)
{
                 indent = indent + "=   ";
                 level++;
//              print out the details for the specified CMDataNode object.
                 Session.Output("  ");
                 Session.Output(indent + " DataNode " + level + " - " + n + "  ----------");
                 Session.Output(indent + "DataNode name: " + dataNode.GetName() );
                 Session.Output(indent + "Line: " + dataNode.GetLine() );
                 Session.Output(indent + "Column: " + dataNode.GetColumn() );
                 Session.Output(indent + "- - - - - - -");
                 Session.Output(indent + "Facet count: " + dataNode.GetFacetCount() );
//               iterate through the array of CMFacet objects and print out their contents
                 var facet as EA.CMFacet;
                 var facetCount;
```

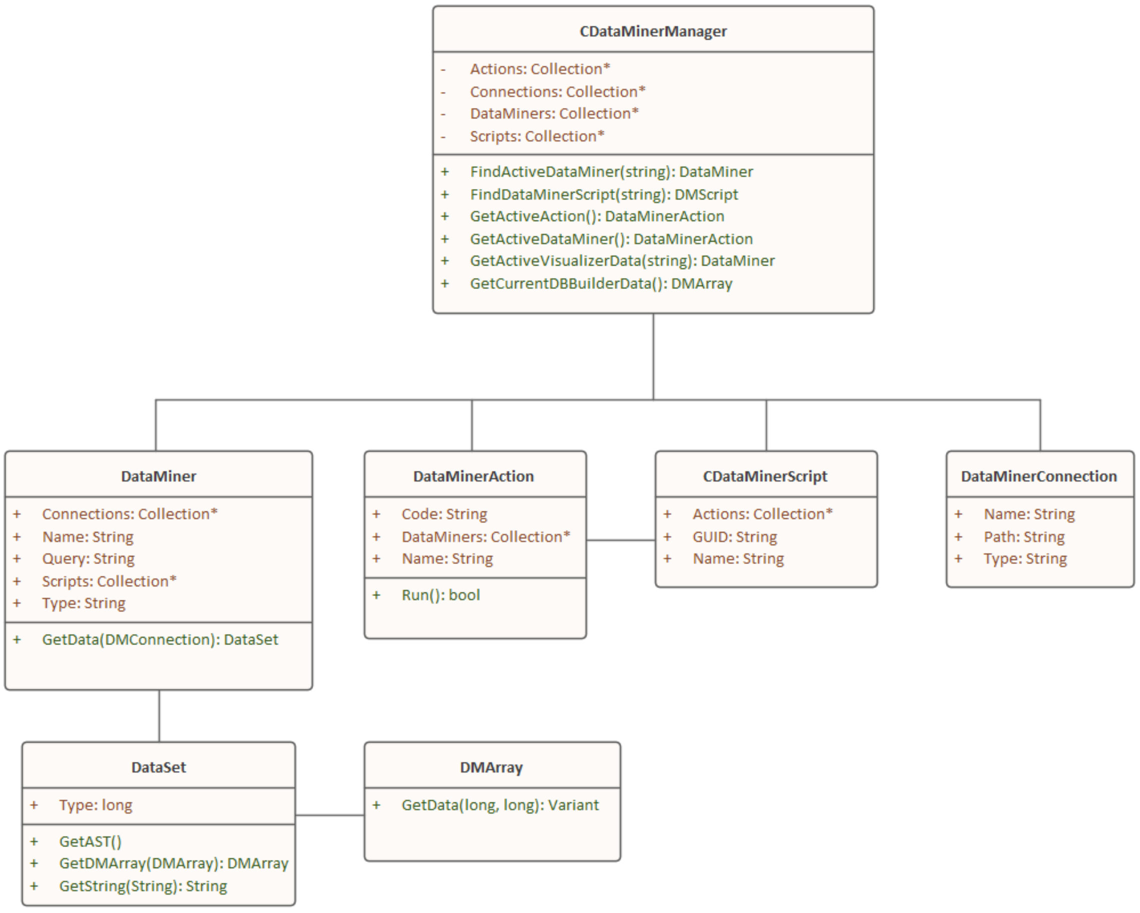
```
                        facetCount = dataNode.GetFacetCount();
                        var k;
                        for(k=0; k < facetCount; k++)
                        {
                                                facet = dataNode.GetFacet(k);
                                                PrintFacetInfo(indent, facet, k+1);
                        }
//                      iterate through the array of CMDataNode child objects and print out their contents
                        var nextLevel = level + 1;
                        Session.Output("    ");
                        Session.Output(indent + "Level " + nextLevel + " DataNode count: " + dataNode.GetChildCount() );
                        var childDataNodeCount;
                        childDataNodeCount = dataNode.GetChildCount();
                        var l;
                        for(l=0; l < childDataNodeCount; l++)
                        {
                                                childDataNode = dataNode.GetChildNode(l);
                                                PrintDataNode(indent, childDataNode, l+1, level);
                        }
                        Session.Output(indent + "---------------------");
}
function PrintResultSetElement(indent, resultset, i)
{
//                      print out the details of the specified result set element
                        indent = indent + "     ";
                        var elementNumber = i + 1;
                        Session.Output(indent + "ResultSet - Element(" + elementNumber + ")");
                        Session.Output(indent + "------------------------");
                        Session.Output(indent + "  Filename : " + resultset.GetFilename(i) );
                        Session.Output(indent + "  Address : " + resultset.GetAddress(i) );
                        Session.Output(indent + "  Position : " + resultset.GetPosition(i) );
                        Session.Output(indent + "  Start : " + resultset.GetPositionStart(i) );
                        Session.Output(indent + "  End : " + resultset.GetPositionEnd(i) );
                        Session.Output(indent + "  Top Level DataNode count: " + resultset.GetChildCount()
);
                        var childCount;
                        childCount = resultset.GetChildCount();
                        var level = 0;
            var dataNode as EA.CMDataNode;
                        var j;
                        for(j=0; j < childCount; j++)
                        {
```

```
                                        dataNode = resultset.GetChildNode(0);
                                        PrintDataNode(indent, dataNode, j+1, level);
                    }
                      Session.Output(indent + "---------------------");
}
function main()
{
                   var minerService as EA.CMService;
                   minerService = Repository.CodeMinerService;
//               Connect to a Code Miner database
                   if( minerService.CMLoadDatabase("C:\\CodeMiner Example\\Project1.cdb") )
                   {
                                var resultset as EA.CMDataSet;
         //                     Execute a query against the Code Miner database
                                Session.Output("Query: " + q1);
                                resultset = minerService.ExecuteMFQL(q1);
                                Session.Output("Results Count: " + resultset.Count);
                                Session.Output("Child count : " + resultset.GetChildCount() );
                                var indent;
                                indent = "  ";
         //                     Iterate through the result set, printing out the details of each 'match' returned by the query.
                                var i;
                                for(i=0; i < resultset.Count; i++)
                                {
                                                       PrintResultSetElement(indent, resultset, i);
                                 }

                                /* Out of Range Test */
                                var x = resultset.GetFilename(8);
                                Session.Output(indent + "Node Filename: " + x );
                 }
                 else
                                Session.Output(indent + "Failed to Open Database: " + minerService.GetLastError() );
}
main();
```

Data Miner Package

The Data Miner Package provides the Automation Interface to the Data Miner elements. It contains these Classes:


CDataMinerManager
- Actions: Collection*
- Connections: Collection*
- DataMiners: Collection*
- Scripts: Collection*
+ FindActiveDataMiner(string): DataMiner
+ FindDataMinerScript(string): DMScript
+ GetActiveAction(): DataMinerAction
+ GetActiveDataMiner(): DataMinerAction
+ GetActiveVisualizerData(string): DataMiner
+ GetCurrentDBBuilderData(): DMArray
DataMiner
+ Connections: Collection*
+ Name: String
+ Query: String
+ Scripts: Collection*
+ Type: String
+ GetData(DMConnection): DataSet
DataMinerAction
+ Code: String
+ DataMiners: Collection*
+ Name: String
+ Run(): bool
CDataMinerScript
+ Actions: Collection*
+ GUID: String
+ Name: String
DataMinerConnection
+ Name: String
+ Path: String
+ Type: String
DataSet
+ Type: long
+ GetAST()
+ GetDMArray(DMArray): DMArray
+ GetString(String): String
DMArray
+ GetData(long, long): Variant

For an overview of using the Data Miner see the *Data Miner* Help topic under the *Model Exchange* group of topics.

Notes

- The Data Miner is available in the Unified and Ultimate Editions

DataMinerManager Class

DataMinerManager Attributes

Attribute	Remarks
Actions	Collection Notes: Returns a pointer to the EA.DMAction objects.
Connections	Collection Notes: Returns a Collection of EA.DMConnection objects.
DataMiners	Collection Notes: Returns a Collection of EA.DataMiner objects
Scripts	Collection Notes: Returns a Collection of EA.DMScript objects.

DataMinerManager Methods

Method	Remarks
FindActiveDataMiner (string guid)	DataMiner Object Loads the DataMiner object from the model specified by its GUID. Returns an EA.DataMiner object or NULL if the current selected object isn't a DataMiner object. Parameters: • GUID: string - GUID of the Data Miner object to look up
FindDataMinerScript (string guid)	DMScript object Returns an EA.DMScript object in the model. Parameters: • GUID: string - GUID of DMScript object.
GetActiveAction ()	DMAction Object When you run an Action (operation), from a diagram, this returns the Action's EA.DMAction object. Note: This is generally used for an Action to work out what DataMiner and DMConnections it is linked to.
GetActiveDataMiner ()	DataMiner Object Returns a pointer to an EA.DataMiner object, or NULL if the currently selected object is not a DataMiner object.

GetActiveVisualizerData (string name)	DataSet Object Get the EA.DataSet of the currently open Visualizer. Parameters: • Name: string - Name of Open Visualizer Note: Passing in a blank name will return the first Visualizer tab.
GetCurrentDBBuilderData ()	DMArray Object Get the current data from the Database Builder's latest SQL query. Returns the current output of the SQL scratch window. Accessible via: Ribbon: Develop > Data Modeling > Database Builder > SQL Scratch Pad. Return Type: DMArray Returns a pointer to an EA.DMArray object, or NULL if there is not a current Database Builder window with returned data. See The Database Builder Help topic for more information on how to get data into this window.

DataMiner Class

DataMiner Attributes

Attribute	Remarks
Connections	Collection A collection of EA.DMConnection's, Notes: Read Only
Name	String Name of the Script object. Notes: Read Only
Query	String Query of the Data miner object Notes: Read Only
Scripts	Collection A collection of EA.DMScript's, Notes: Read Only
Type	String Type of the Data miner object Notes: Read Only

DataMiner Methods

Method	Remarks
GetData (DMCconnection Connection)	DataSet Returns an EA.DataSet object that represents the query on the connection. Parameters: • connection: DMConnection - A DMConnection object

DataSet Class

DataSet Attributes

Attribute	Remarks
Type	long Type of data contained in this data set. 1. Safe Array 2. Abstract Data type 3. JSon 4. Text Notes: Read Only

DataSet Methods

Method	Remarks
GetAST ()	Currently not supported
GetDMArray ()	DMArray Returns an EA.DMArray object Note: Only supported when Type = 1
GetString ()	String Returns a string of the data. NOTE: Only supported when Type = 3 or 4.

DMArray Class

DMArray Attributes

Attribute	Remarks
ColumnCount	long Notes: Read Only Number of Columns returned in this dataset
RowCount	long Notes: Read Only Number of rows returned in this dataset

DMArray Methods

GetData (long row, long column)	Variant Notes: When the database returns a NULL value, this will return an empty string. Return: Variant. Parameters: • row: Row number of data • column: Column number of data

DMAction Class

DMAction Attributes

Attribute	Remarks
Code	String The code on the Action Notes: Read Only
DataMiners	Collection A Collection of DMDataminer objects Notes: Read Only
Name	String Name of the Action. Notes: Read Only

DMAction Methods

Run ()	Boolean Returns TRUE if the script was run successfully.

DMScript Class

DMScript Attributes

Attribute	Remarks
Actions	Collection Returns a Collection of EA.DMAction's
GUID	String Guid of the Script object. Notes: Read Only
Name	String Name of the Script object. Notes: Read Only

DMConnection Class

DMConnection Attributes

String

Sets the type that the connect object is.

Notes: Read Only

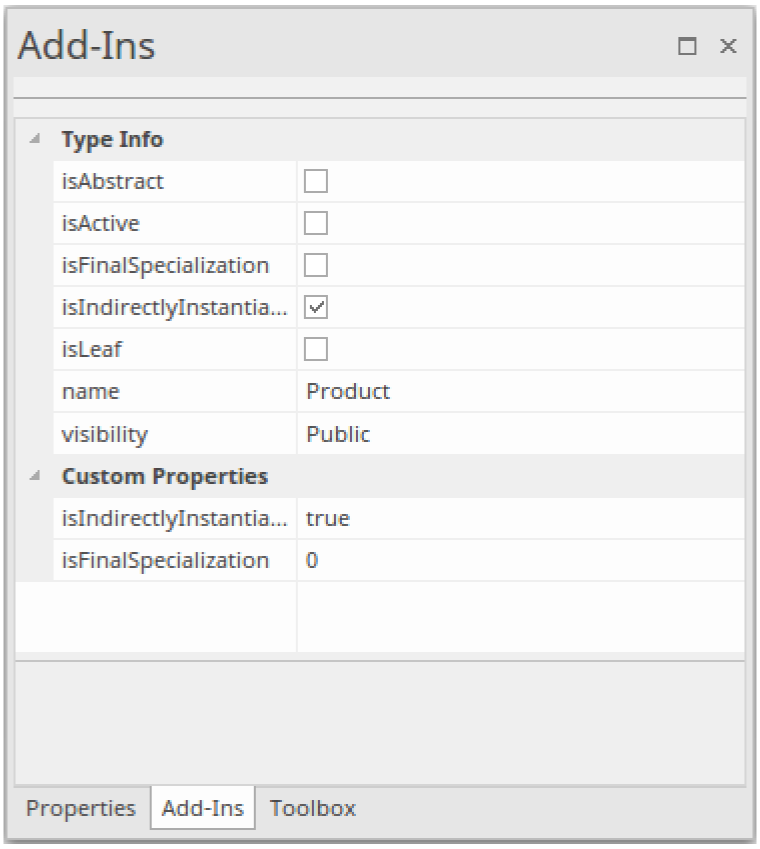
Attribute	Remarks
Name	Type: String Notes: Read Only Name of the Connection object.
Path	Type: String Path to the data we are connecting to. Notes: Read Only
Type	Type: String Notes: Read Only Type of Connection. Options: • ODBC • EA Repository • File • URL

TypeInfoProperties Package

The TypeInfoProperties Package provides an interface to the properties of an object from the perspective of the technology rather than the Enterprise Architect database, allowing read and write access to those properties. It effectively shows the properties contained in the technology-specific and custom categories of the Properties window for the object (and omits the Enterprise Architect specific properties such as the General and Project properties). The interface hides the origin of the properties - whether they are from the base object directly, a Tagged Value, or are MOF properties.

You can see this interface in action in the EA.Example model ('Start > Help > Help > Open the Example Model'). When you open this model:

1. Select the 'Specialize > Manage Addin' ribbon option.
2. Select the checkbox against 'Type Info' and click on the OK button. An icon for 'Type Info' displays on the right of the Add-Ins panel.
3. Click on the drop-down arrow and select the 'Show Type Info' option. The Add-Ins window displays, showing the type information (properties) for the currently-selected object.
4. If you also want to display custom properties in the Add-Ins window, click on the 'Type-Info' icon again and select the 'Include Custom Properties option'. The window resembles this illustration, which is for a UML Component element.



Type Info	
isAbstract	☐
isActive	☐
isFinalSpecialization	☐
isIndirectlyInstantia...	✓
isLeaf	☐
name	Product
visibility	Public
Custom Properties	
isIndirectlyInstantia...	true
isFinalSpecialization	0

5. Browse the EA.Example model, clicking on different types of object. You will see a different list of properties for, say, an Action than for a Class. Then you can both read and write to those properties. Also compare the list with the Properties window for the same objects.

TypeInfoProperties Class

TypeInfoProperties Attributes

Attribute	Remarks
Count	long Returns the number of TypeInfo Properties.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

TypeInfoProperties Methods

Method	Remarks
GetLastError ()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
GetProperty (String PropName)	Returns the property value as a string. Parameters: • PropName : String - Name of the property
HasProperty (String PropName)	Returns True if the object has the property. Parameters: • PropName : String - Name of the property
Items (object Index)	TypeInfoProperty collection Notes: Accesses an individual TypeInfoProperty. Parameters: • Index: Object - Either a string representing the title text or an integer representing the zero-based index of the TypeInfoProperty to get
SetProperty (String PropName, String Value)	Returns True if the property was set. Parameters: • PropName : String - Name of property • Value : String - Value of property

TypeInfoProperty Class

TypeInfoProperty Attributes

Attribute	Remarks
Name	String Notes: Readonly. Name of the property.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Value	String Get/Sets the Property value.

TypeInfoProperty Methods

<None.>

Method	Remarks

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: • 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: • 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: • 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: • 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: • 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: • 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: • 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:

	• UserName: String - The name of a user defined in the current model
GetWebHyperlink (string URL, string LinkText)	String Notes: Returns a string containing a hyperlink to the URL specified by the input parameter URL. Parameters: • 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)	Boolean Notes: Creates and sends a new mail message using the values specified in the input parameters. Parameters: • 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

Search Window Package

The Search Window Package contains:

- The EAContext Class, which provides a description of a single selected item
- The EASelection Class, which provides optimized functions to access information about the current selection
- The SearchWindow Class, which provides a method for displaying the results of your operation using the Search Window

EAContext Class

The EAContext Class provides a description of a single selected item. The fields with values depend on the location of the selected item.

EAContext Attributes

Atttribute	Remarks
Alias	String Notes: Read only The Alias of the context item.
BaseType	String Notes: Read only Returns the base UML type of the context item.
ContextType	ContextType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
ElementGUID	String Notes: Read only The GUID of the current context object; empty if an object isn't selected. That is: • ElementGUID if an element has context • AttributeGUID if an attribute has context • MethodGUID if an operation has context. • DiagramGUID if a diagram has context • PackageGUID if a Package has context
ElementID	Long Notes: Read only The ID of the current context object; 0 if an object isn't selected. That is: • ElementID if an element has context • AttributeID if an attribute has context • MethodID if an operation has context. • DiagramID if a diagram has context • PackageID if a Package has context
Locked	Boolean Notes: Read only Indicates if the context item is locked.
MetaType	String Notes: Read only

	Returns the metatype of the context item.
Name	String Notes: Read only The name of the context item.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

EAContext Methods

Method	Remarks
HasStereotype (String stereo)	Boolean Returns: True if the stereotype is applied to an object. Parameters • stereo: String - the stereotype to check against the context object, to see if has been applied

EASelection Class

The EASelection Class provides optimized functions to access information on the current selection. It should be used when building Add-In menus and setting the menu state, as almost all properties can be used without any database queries being made.

EASelection Attributes

Attribute	Remarks
Context	EAContext Notes: Describes the currently focused element without requiring any database calls.
ElementSet	Collection Notes: When the selection consists of one or more objects of type otElement, this provides a collection giving optimized access to all of those elements.
List	Collection Notes: For any window where multiple selection is supported, this provides a list describing the type of every selected element without requiring any database calls.
Location	String Notes: Provides the type of window that contains the current selection. Possible values are: • Calendar • Diagram • Dialog • Element List • Gantt • Model View • Browser window • Project View • Relationship Matrix • Reviews • Search • Specification Manager Further values could be added to this list in the future.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

EASelection Methods

None.

SearchWindow Class

The SearchWindow Class provides a method for displaying the results of your operation using the Search Window.

SearchWindow Attributes

Attribute	Remarks
FieldChooserVisible	Boolean Shows or hides the search Field Chooser.
FiltersVisible	Boolean Shows or hides the search filters.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

SearchWindow Methods

Method	Remarks
AddColumn (string Name, long Width)	Adds the column into the current Search window. Returns the column number, or -1 on error. Parameters: • Name: String - Name of the column • Width: Long - Width of the column
AddRow (ObjectType ot, String ElementGUID, Long ElementID, String ClassType, VARIANT Values)	Returns the row inserted into the search. Parameters: • ot: ObjectType - the Object Type • ElementGUID: String - GUID of the element • ElementID: long - Object ID of the element • ClassType: String - the type of object • Values: an array of values
ClearGrouping ()	Clear all groupings in the search. Returns FALSE on error.
ClearSorting ()	Clear all column sorting in the search. Returns FALSE on error.
EnsureVisible ()	Make the Search window visible.

	Returns FALSE, if the Search window isn't open.
GetCell (long Row, long Column)	Returns the value of the cell. Parameters: • Row: long - Row number • Column: long - Column number
GroupByColumn (long Column)	Sets the group order by column. Returns FALSE if it cannot group by the specified column. Parameters: • Column: Long - Column number
LoadLayout (string LayoutGUID)	Set the layout of the Search window. Returns FALSE if the layout cannot be set. Parameters: • LayoutGUID: String - Layout GUID
NewLayout (string LayoutGUID)	Saves the layout of the Search window. Parameters: • LayoutGUID: String - Layout GUID
SetCellString (long Row, long Column, String Data)	Sets a value in a cell. Parameters: • Row: long - Row number • Column: long - Column number • Data: String - Value to set the cell to
SetCellVariant (long Row, long Column, VARIANT Data)	Sets an alternative value in a cell. Parameters: • Row : long - Row number • Column : long - Column number • Data: Value to set the cell to
SortByColumn (long Column)	Sets the column to sort by. Returns FALSE if it cannot sort by the specified column. Parameters: • Column: Long - Column number

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: • sSignalName: String - the name of the signal OR the GUID of the Signal element • sParameters: 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 in *model,* defined for this property in the Schema Composer. Combine with SchemaChoices attribute to obtain all available choices.
SchemaChoices	SchemaTypeEnum Returns an iterator allowing navigation of choice elements in *schema,* defined for this property in the Schema Composer. Combine with Choices attribute to obtain all available choices.
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.

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)	void Notes: Use this function to add entries that are offered by the Schema Composer when the user clicks on the Generate button. Parameters: • description: describes the export format provided by the Add-In
SetCapability(string name,boolean enabled)	void Notes: Use this function to enable/disable capabilities. Parameters: • name: name of the capability • enabled: True or False Capabilities: 'allowCardinality' - allows/denies restrictions to cardinality 'allowRootElement' - allows/denies setting root element 'allowPropByRef' - allows/denies By Reference restriction 'allowRedefine' - allows/denies ability to redefine an element
SetProperty(string name, string value)	void Notes: Sets properties displayed in the Schema Composer. Parameters: • name: property name • value: property value Properties: 'Namespace' - Target namespace for XML schema 'Namespace Prefix' - Namespace prefix for XML schema 'Qualifier' - string qualifier that prepends schema type names

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.
Namespaces	SchemaNamespaceEnum Notes: Read only Enumerator for the namespaces referenced by schema

SchemaComposer Methods

Method	Description
FindInSchema(long	SchemaType

typeID)	Notes: Obtains an interface to a Class as represented in the schema for a given model Class ID. Parameters: • typeID: the model Class ID
FindInModel(long typeID)	ModelType Notes: Obtains an interface to a Class as represented in the UML model for a given model Class ID Parameters: • typeID: the model Class ID
FindSchemaTypeByName(string typename)	SchemaType Notes: Returns an interface to the schema type that matches the type specified or null if no type exists. Parameters: • name : the name of the type
GetNamespacePrefixForType(long typeID)	String Notes: Returns the schema namespace prefix for a given type Parameters: • typeID: the model Class ID
GetNamespaceForPrefix(string prefix)	String Notes: Returns the URI for a given schema namespace prefix Parameters: • name: the namespace prefix

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.

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)	ModelTypeEnum Notes: Enumerator Returns an enumerator that can be used to traverse the Class ancestry. Parameters: • searchtype: the type of traversal to use, breadth first or depth first
GetSubClassEnum(SearchType searchType)	ModelTypeEnum Notes: Enumerator Returns an enumerator that can be used to iterate over any descendents of the Class. Parameters: • searchtype: the type of traversal to use, breadth first or depth first
IsEnumeration	True where type represents an enumeration element

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.

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.
IsEnumeration()	True if the type represents an enumeration element

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. Could be null depending on composition method.

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.

SchemaNamespace Class

An interface presenting namespace information

SchemaNamespace Attributes

Name	string Notes: Read only The namespace prefix.
URI	string Notes: Read only The URI of the namespace.

SchemaNamespaceEnum Class

An enumerator interface for namespaces referenced by schema.

SchemaNamespaceEnum Methods

GetFirst()	SchemaNamespace Returns the first namespace interface in a collection of namespaces.
GetNext()	SchemaNamespace Returns the next namespace interface in a collection of namespaces

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
```