Enterprise Architect

User Guide Series

Scripting

Author: Sparx Systems

Date: 7/08/2019

Version: 1.0

CREATED WITH ENTERPRISE ARCHITECT

Table of Contents

Scripting

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

Script Engine Support

- Mozilla SpiderMonkey [version 1.8]
- Microsoft Scripting Engine

Script Languages

- JavaScript
- JScript
- VBScript

Benefits

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

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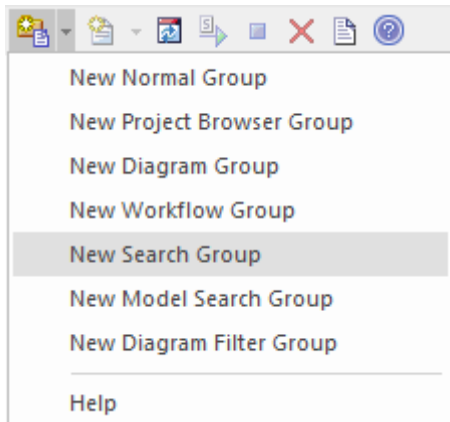
Scripts are managed and contained in groups. Each group has an attribute called 'Type'. This attribute is used to help Enterprise Architect decide how and where the script can be used and from which features it should be made available. The properties of a script group can be viewed from its shortcut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

Script Storage

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

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

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

Using Scripts

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

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

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

Notes

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

Scripts Tab

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

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

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

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

Access

Ribbon	Specialize > Tools > Scripting > Scripts

Script Toolbar

Icon	Action
	Create a new script group; this option displays a short menu of the types of script group you can create, namely: - Normal Group () - Browser window Group () - Diagram Group () - Workflow Group () - Search Group () - Model Search Group The new group is added to the end of the list in the Scripting window, with the 'New group' text highlighted so that you can type in the group name.
	Create a new script file in the selected script group; displays a short menu of the types of script you can create, namely: - VBScript ()

	• JScript ( ) • JavaScript ( ) The new script is added to the end of the list in the selected group, with the 'New script' text highlighted so that you can type in the script name.
	Refresh the script tree in the Scripting window; this icon also reloads any changes made to a workflow script.
	Compile and execute the selected script. The output from the script is written to the 'Script' tab of the System Output window, which you display using the View Script Output button.
	Stop an executing script; the icon is disabled if no script is executing.
	Delete a script from the model; you cannot use this icon to delete a script group (see the earlier 'Context Menu' item), scripts in the 'Local Scripts' group, or a script that is executing. The system prompts you to confirm the deletion only if the 'Confirm Deletes' checkbox is selected in the 'Project Browser' panel of the 'General' page of the 'Preferences' dialog; if this option is not selected, no prompt is displayed. Script deletion is permanent - scripts cannot be recovered.
	Display the System Output window with the results of the most recently executed script displayed in the 'Script' tab.

Notes

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

Console Tab

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

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

You can input two types of command:

- Console commands
- Script commands

Access

Ribbon	Specialize > Tools > Scripting > Console

Console Commands

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

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

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

Script Commands

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

Examples:

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

>!VB
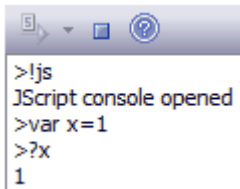
>!i MyGroup.MyScript

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

>!JS

>var x = 1

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

Console Tab Toolbar

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

- Open Console ( ) - click on the down-arrow and select to open a VBScript console, JScript console or JavaScript console

- Stop Script ( ) - click to stop an executing script and close the current console

Notes

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

Script Group Properties

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

Access

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

Define the Script Group Properties

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

<table>
<tr><td></td><td>

- Normal - (icon) General model scripts

- Browser window - (icon) Scripts that are listed in and can be executed from the Browser window 'Scripts' context menu option

- Workflow - (icon) Scripts executed by Enterprise Architect's workflow engine; you can create only VB scripts of this type

- Search - (icon) Scripts that can be executed as model searches; these scripts are listed in the 'Search' field of the Model Search window, in the last category in the list

- Diagram - (icon) Scripts that can be executed from the 'Scripts' submenu of the diagram context menu

- Find in Project - (icon) Scripts that can be executed from the 'Scripts' submenu of a context menu within the Model Search view, on the results of a successfully-executed SQL search that includes CLASSGUID and CLASSTYPE, or a Query-built search

- Element - Scripts that can be executed from the 'Scripts' submenu of element context menus; accessible from the Browser window, Diagram, Model Search, Element List, Package Browser and Gantt views

- Package - Scripts that can be executed from the 'Scripts' submenu of Package context menus; accessible from the Browser window

- Diagram - Scripts that can be executed from the 'Scripts' context menu option for diagrams; accessible from the Browser window and diagrams

- Link - Scripts that can be executed from the 'Scripts' context menu option for connectors; accessible from diagrams

</td></tr>
<tr><td>Notes</td><td>Type in any comments you need regarding this script group.</td></tr>
</table>

Script Editor

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

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

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

Access

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

Facilities

Facility	Detail
Scripting Objects	Enterprise Architect adds to the available functionality and features of the editor script language by providing inbuilt objects; these are either Type Libraries providing Intelli-sense for editing purposes, or Runtime objects providing access to objects of the types described in the Type Libraries.
	The available Intelli-sense scripting objects are:
	- EA
	- MathLib
	- System
	The runtime scripting objects are:
	- Repository (Type: IDualRepository, an instance of EA.Repository, the Enterprise Architect Automation Interface)
	- Maths (Type: IMath, an instance of MathLib; this exposes functions from the Cephes mathematical library for use in scripts)
	- Session (Type: ISession, an instance of System)
Script Editing Intelli-sense (Required Syntax)	Intelli-sense is available not only in the 'Script Editor', but also in the 'Script Console'; Intelli-sense at its most basic is presented for the inbuilt functionality of the script engine.
	For Intelli-sense on the additional Enterprise Architect scripting objects (as listed) you must declare variables according to syntax that specifies a type; it is not necessary to use this syntax to execute a script properly, it is only present so that the correct Intelli-sense can be displayed for an item.

	The syntax can be seen in, for example:
	
	   Dim e as EA.Element
	
	Then when you type, in this case, e., the editor displays a list of member functions and properties of e's type.
	
	You select one of these to complete the line of script; you might, therefore, type:
	
	   VBTrace(e.
	
	As you type the period, the editor presents the appropriate list and you might double-click on, for example, Abstract; this is inserted in the line, and you continue to type or select the rest of the statement, in this case adding the end space and parenthesis:
	
	   VBTrace(e.Abstract)

Keystrokes	In the Script Editor or Console, Intelli-sense is presented on these keystrokes. • Press . (period) after an item to list any members for that item's type • Press Ctrl+Space on a word to list any Intelli-sense items with a name starting with the string at the point the keys were pressed • Press Ctrl+Space when not on a word to display any available top level Intelli-sense items - these are the Intelli-sense objects already described plus any built-in methods and properties of the current scripting language

Include script libraries	An *Include* statement (!INC) allows a script to reference constants, functions and variables defined by another script accessible within the Scripting Window. Include statements are typically used at the beginning of a script.
	
	To include a script library, use this syntax:
	
	   !INC [Script Group Name].[Script Name]
	
	For example:
	
	   !INC Local Scripts.EAConstants-VBScript

Using Inbuilt Math Functions	Various mathematical functions are available within the Script Editor, through the use of the inbuilt Maths object.
	
	You can access the Maths object within the Script Editor by typing 'Maths' followed by a period. The Intelli-sense feature displays a list of the available mathematical functions provided by the Cephes Mathematical Library. For example:
	
	   Session.Output "The square root of 9 is " & Maths.sqrt(9)
	
	   Session.Output "2^10 = " & Maths.pow(2,10)
	
	The Maths object is available in the Unified and Ultimate editions of Enterprise Architect.

Using COM / ActiveX Objects	VBScript, JScript and JavaScript can each create and work with ActiveX / COM objects. This can help you to work with external libraries, or to interact with other applications external to Enterprise Architect. For example, the Scripting.FileSystemObject Class can be used to read and write files on the local machine. The syntax for creating a new object varies slightly for each language, as illustrated by these examples:
	
	VBScript:
	
	   set fsObject = CreateObject("Scripting.FileSystemObject")
	
	JScript:
	
	   fsObject = new ActiveXObject("Scripting.FileSystemObject");
	
	JavaScript:

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

Notes

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

Session Object

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

Properties

Properties	Detail
Attributes	• UserName - Returns the current windows username (read only) • Version - Returns the version of this object (read only)
Methods	• Input(string Prompt) - displays a dialog box prompting the user to input a value; returns the string value that was entered by the user • Output(string Output) - writes text to the current default output location; during: - Normal script execution, output is written to the 'Script' tab of the System Output window - Script Debugging, output is written to the Debug window - Use of the Script Console, output is written to the Console • Prompt(string Prompt, long PromptType) - displays a modal dialog containing the specified prompt text and button types; returns the 'PromptResult' value corresponding to the button that the user clicked
PromptType values	• promptOK = 1 • promptYESNO = 2 • promptYESNOCANCEL = 3 • promptOKCANCEL = 4
PromptResult values	• resultOK = 1 • resultCancel = 2 • resultYes = 3 • resultNo = 4
Session.Prompt Example	(VBScript) If (Session.Prompt("Continue?", promptYESNO) = resultYes) Then...

Script Debugging

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

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

Access

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

Begin debugging a model script

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

Notes

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