

Common Automation Reference

Release PADS VX.2.6

Document Revision 4

© 2011-2019 Mentor Graphics Corporation
All rights reserved.

This document contains information that is proprietary to Mentor Graphics Corporation. The original recipient of this document may duplicate this document in whole or in part for internal business purposes only, provided that this entire notice appears in all copies. In duplicating any part of this document, the recipient agrees to make every reasonable effort to prevent the unauthorized use and distribution of the proprietary information.

This document is for information and instruction purposes. Mentor Graphics reserves the right to make changes in specifications and other information contained in this publication without prior notice, and the reader should, in all cases, consult Mentor Graphics to determine whether any changes have been made.

The terms and conditions governing the sale and licensing of Mentor Graphics products are set forth in written agreements between Mentor Graphics and its customers. No representation or other affirmation of fact contained in this publication shall be deemed to be a warranty or give rise to any liability of Mentor Graphics whatsoever.

MENTOR GRAPHICS MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

MENTOR GRAPHICS SHALL NOT BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS PUBLICATION OR THE INFORMATION CONTAINED IN IT, EVEN IF MENTOR GRAPHICS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

U.S. GOVERNMENT LICENSE RIGHTS: The software and documentation were developed entirely at private expense and are commercial computer software and commercial computer software documentation within the meaning of the applicable acquisition regulations. Accordingly, pursuant to FAR 48 CFR 12.212 and DFARS 48 CFR 227.7202, use, duplication and disclosure by or for the U.S. Government or a U.S. Government subcontractor is subject solely to the terms and conditions set forth in the license agreement provided with the software, except for provisions which are contrary to applicable mandatory federal laws.

TRADEMARKS: The trademarks, logos and service marks ("Marks") used herein are the property of Mentor Graphics Corporation or other parties. No one is permitted to use these Marks without the prior written consent of Mentor Graphics or the owner of the Mark, as applicable. The use herein of a third-party Mark is not an attempt to indicate Mentor Graphics as a source of a product, but is intended to indicate a product from, or associated with, a particular third party. A current list of Mentor Graphics' trademarks may be viewed at: mentor.com/trademarks.

The registered trademark Linux[®] is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.

End-User License Agreement: You can print a copy of the End-User License Agreement from: mentor.com/eula.

Mentor Graphics Corporation
8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777
Telephone: 503.685.7000
Toll-Free Telephone: 800.592.2210
Website: mentor.com
Support Center: support.mentor.com

Send Feedback on Documentation: support.mentor.com/doc_feedback_form

Revision History ISO-26262

Revision	Changes	Status/Date
4	Modifications to title page to reflect the latest product version supported. Approved by Regis Krug. All technical enhancements, changes, and fixes listed in the <i>Personal Automated Design System Release Notes</i> for this product are reflected in this document. Approved by Mike Bare.	Released September 2019
3	Modifications to title page to reflect the latest product version supported. Approved by Regis Krug. All technical enhancements, changes, and fixes listed in the <i>Personal Automated Design System Release Notes</i> for this product are reflected in this document. Approved by Mike Bare.	Released March 2019
2	Modifications to title page to reflect the latest product version supported. Approved by Regis Krug. All technical enhancements, changes, and fixes listed in the <i>Personal Automated Design System Release Notes</i> for this product are reflected in this document. Approved by Mike Bare.	Released September 2018
1	Modifications to improve the readability and comprehension of the content. Approved by Regis Krug. All technical enhancements, changes, and fixes listed in the <i>Personal Automated Design System Release Notes</i> for this product are reflected in this document. Approved by Mike Bare.	Released February 2018

Author: In-house procedures and working practices require multiple authors for documents. All associated authors for each topic within this document are tracked within the Mentor Graphics Technical Publication's source. For specific topic authors, contact Mentor Graphics Technical Publication department.

Revision History: Released documents include a revision history of up to four revisions. For earlier revision history, refer to earlier releases of documentation on Support Center.

Table of Contents

Revision History ISO-26262

Chapter 1

Introduction.....	9
-------------------	---

Chapter 2

Scripting With Multiple Installs	11
Identifying the COM Version Number of an Install	11
COM Version Script Examples	13
ReleaseEnvServer Object	17
GetInstalledReleases Method (ReleaseEnvServer Object).....	18
SetEnvironment Method (ReleaseEnvServer Object)	20
ProgIDVersion Property (ReleaseEnvServer Object).....	21
sddHome Property (ReleaseEnvServer Object)	22
sddPlatform Property (ReleaseEnvServer Object)	23
sddVersion Property (ReleaseEnvServer Object).....	24
Release Environment Server Enumerated Types.....	25
EReleaseEnvErrCode Enum	26

Chapter 3

Scripting Server	29
Scripting Object Data Model	29
Program IDs for Applications and Automation Engines	30
Scripting Object	32
AddTypeLibrary Method (Scripting Object)	33
AttachEvents Method (Scripting Object)	34
CreateObject Method (Scripting Object)	35
DetachEvents Method (Scripting Object).....	36
ExpandEnvironmentStrings Method (Scripting Object).....	37
GetEnvVariable Method (Scripting Object).....	38
SetEnvVariable Method (Scripting Object)	39
Sleep Method (Scripting Object)	40
DontExit Property (Scripting Object).....	41
Globals Property (Scripting Object)	42
IsUnix Property (Scripting Object).....	43
OSName Property (Scripting Object).....	44
OSVersion Property (Scripting Object)	45
Struct Object	46
Count Property (Struct Object).....	47
Data Property (Struct Object)	48
Item Property (Struct Object)	49
Keys Property (Struct Object).....	50

Chapter 4**CommandBar Server 51**

CommandBarServer Data Model.	52
CommandBar Object.	53
Controls Property (CommandBar Object)	54
CommandBarButton Object.	55
Delete Method (CommandBarButton Object)	57
BitmapFile Property (CommandBarButton Object)	58
BitmapResource Property (CommandBarButton Object)	59
Caption Property (CommandBarButton Object)	60
DescriptionText Property (CommandBarButton Object)	61
Enabled Property (CommandBarButton Object)	62
ExecuteMethod Property (CommandBarButton Object)	63
Id Property (CommandBarButton Object)	64
OnAction Property (CommandBarButton Object)	66
ResourceDLL Property (CommandBarButton Object)	67
Target Property (CommandBarButton Object)	68
TooltipText Property (CommandBarButton Object)	69
UpdateMethod Property (CommandBarButton Object)	70
CommandBarControl Object.	71
Delete Method (CommandBarControl Object)	72
Caption Property (CommandBarControl Object)	73
CommandBarPopup Object.	74
Delete Method (CommandBarPopup Object)	75
Caption Property (CommandBarPopup Object)	76
Controls Property (CommandBarPopup Object)	77
CommandBarControls Collection.	78
Add Method (CommandBarControls Collection)	79
Count Property (CommandBarControls Collection)	80
Item Property (CommandBarControls Collection)	81
CommandBars Collection.	82
Item Property (CommandBars Collection)	83
Command Bar Enumerated Types.	84
CmdControlType Enum.	85

Chapter 5**KeyBind Server 87**

KeyBindServer Data Model.	87
KeyBinding Object.	89
Command Property (KeyBinding Object)	90
ExecuteMethod Property (KeyBinding Object)	91
Id Property (KeyBinding Object)	92
Key Property (KeyBinding Object)	93
KeyType Property (KeyBinding Object)	94
Target Property (KeyBinding Object)	95
Type Property (KeyBinding Object)	96
VirtualKey Property (KeyBinding Object)	97
BindingTables Collection.	98

Table of Contents

AddKeyBinding Method (BindingTables Collection)	99
AddStrokeBinding Method (BindingTables Collection)	100
Bindings Property (BindingTables Collection)	102
Item Property (BindingTables Collection)	103
KeyBindings Collection	104
Remove Method (KeyBindings Collection)	105
Count Property (KeyBindings Collection)	106
Item Property (KeyBindings Collection)	107
Key Binding Enumerated Types	108
BindKeyType Enum	109
BindType Enum	110
 Chapter 6	
Addin Controls	111
Addin Object	112
Active Property (Addin Object)	113
Control Property (Addin Object)	114
ClassID Property (Addin Object)	115
Description Property (Addin Object)	116
DisplayName Property (Addin Object)	117
Group Property (Addin Object)	118
Name Property (Addin Object)	119
Placement Property (Addin Object)	120
ProgID Property (Addin Object)	121
ShortCutKey Property (Addin Object)	122
Visible Property (Addin Object)	123
Width Property (Addin Object)	124
MGCAAddins Collection	125
Add Method (MGCAAddins Collection)	126
Remove Method (MGCAAddins Collection)	127
Count Property (MGCAAddins Collection)	128
Item Property (MGCAAddins Collection)	129
Addins Enumerated Types	130
PlacementLocation Enum	131
 Chapter 7	
JScriptHelper	133
JScriptHelper Object	134
ToScriptArray Method (JScriptHelper Object)	135
Nothing Property (JScriptHelper Object)	136
JScriptHelper Example	136
 Third-Party Information	
 End-User License Agreement	
with EDA Software Supplemental Terms	

Chapter 1

Introduction

This document provides reference information about servers that are common to multiple Mentor Graphics applications. This information is useful for anyone seeking to develop automation scripts for use with PADS Designer or other Mentor Graphics tools.

The servers described in this document are:

- The `ReleaseEnvServer` object, which enables you to manage scripts on machines that have multiple PCB software installs. See [Scripting With Multiple Installs](#).
- [Scripting Object](#) server, which is available to all applications. That is, you do not need any special calls in your script to include this server.
- The [CommandBar Server](#), which allows you to customize the toolbars and menus of applications.
- The [KeyBind Server](#), which allows you to define custom shortcut or key accelerators for applications.
- The [Addin Controls](#), which allows you to include and run add-ins in the applications.
- The [JScriptHelper](#), which is available to all applications. That is, you do not need any special calls in your script to include this server.

Chapter 2

Scripting With Multiple Installs

You can develop and run scripts on a machine with multiple Mentor PCB releases installed. To determine which install the script calls, you include a COM version number when you create application instances in your scripts. With more complex scripts, you can query the installed releases on a machine and perform different actions based on the environment.

Note



If you do not include COM version numbers in your scripts, the version defaults to the last software install tree.

Topic	Description
Identifying the COM Version Number of an Install	To use Mentor automation in a design environment where multiple software installs/releases exist on the same machine, you must know the COM version number for each of the installs.
COM Version Script Examples	VBScript examples show you how to manage scripts on systems that have multiple Mentor PCB software installs.
ReleaseEnvServer Object	The ReleaseEnvServer object provides access to the PCB software install environment. The COM version of this object does not change.

Identifying the COM Version Number of an Install

To use Mentor automation in a design environment where multiple software installs/releases exist on the same machine, you must know the COM version number for each of the installs.

You can also automate the process of identifying the COM Version number using methods in the [ReleaseEnvServer Object](#).

Procedure

Use any of the following methods to determine the COM Version Number:

If you want to...	Do the following...
View the <i>COMVersion.xml</i> file	<ol style="list-style-type: none">1. In a text editor, open <code><install>/SDD_HOME/standard/COMVersion.xml</code>2. Search for “PROG_ID_VER” The <code><data></code> element contains the COM version number. Example:<pre><var> <name>PROG_ID_VER</name> <data>3</data> <append>replace</append> </var></pre> <ol style="list-style-type: none">1. Repeat these steps for each install.
Run ReleaseReader command	<ol style="list-style-type: none">1. Do one of the following, based on your operating system.<ul style="list-style-type: none">• Windows 7—Choose Start > <i>release_name</i> > Administrative Tools > MGC PCB Command Window VX1. Note: You can also type mgcmd in the Start menu’s search box.• Windows 8—In the Apps view, under <i>release_name</i> > Administrative Tools, click MGC PCB Command Window VX.1. Note: You can also type mgcmd in the Start screen to invoke the search app.2. In the command window, type ReleaseReader. The output lists each COM_VERSION variable on the machine. Example:<pre>[HKLM32_MGC_4] SDD_VERSION: PADSPProVX.1 SDD_HOME: C:\MentorGraphics\PADSPProVX.1\SDD_HOME SDD_PLATFORM: win32 COM_VERSION: 3 [HKCU_MGC_2] SDD_VERSION: EEVX.1 SDD_HOME: C:\MentorGraphics\EEVX.1\SDD_HOME SDD_PLATFORM: win32 COM_VERSION: 9</pre>

Related Topics

[COM Version Script Examples](#)

ReleaseEnvServer Object

COM Version Script Examples

VBScript examples show you how to manage scripts on systems that have multiple Mentor PCB software installs.

Note



If you are running a 32-bit install on a 64-bit machine, ensure that you use 32-bit script hosts in SysWoW64.

To start a 32-bit command prompt, do one of the following, depending on your operating system:

- **Windows 7**—Choose **Start > Run**, type `%windir%\SysWoW64\cmd.exe`, then click **OK**.
- **Windows 8**—Press the **<Windows>-R** key combination, type `%windir%\SysWoW64\cmd.exe`, then click **OK**.

Example - Creating an instance of the last installed layout application

```
Dim app

' Create an instance of the Layout application (last install).
Set app = CreateObject("PowerPCB.Application")
```

Example - Creating an instance of the PADS VX.2.5 layout application

```
Dim app

' Create an instance of the Layout application (for PADS VX.2.5 install).
Set app = CreateObject("PowerPCB.Application.78")
```

Example - Using ReleaseEnvServer methods to create an instance of the layout application

```
' Get the environment of a specific SDD_HOME
Dim dllApp
Set dllApp =
    CreateObject("MGCPBReleaseEnvironmentlib.MGCPBReleaseEnvServer")
Call dllApp.SetEnvironment("c:\\MentorGraphics\\PADSVX.2.5\\SDD_HOME")

' Launch the specific version of the Layout Application
Dim app
Set app = CreateObject("PowerPCB.Application" & "." &
    dllApp.ProgIDVersion)
app.Visible = True

' Open a message box
MsgBox "Click OK to close PADS Layout " & dllApp.sddVersion, vbOKOnly,
    "Layout Automation"

' Exit Layout
app.Quit
```

Example - ReleaseEnvServer Object

```
Option Explicit
On Error Resume Next
Dim textFile
Dim strHome : strHome=""
dim strPlatform : strPlatform = ""

textFile = "TestOutput.txt"
' Get the application object
Dim dllApp
Set dllApp = CreateObject("MGPCPCBReleaseEnvironmentLib.MGPCPCBReleaseEnvServer")
Dim installedReleases

'Default environment -- no argument, write the parameters to a text file
dllApp.SetEnvironment("")
Call WriteTextToFile(textFile, "Default Environment:" & vbcrLf)
Call WriteTextToFile(textFile, "-----" & vbcrLf)
Call WriteTextToFile(textFile, "PROG_ID_VER=" & dllApp.ProgIDVersion & vbcrLf)
Call WriteTextToFile(textFile, "SDD_VERSION=" & dllApp.sddVersion & vbcrLf)
Call WriteTextToFile(textFile, "SDD_HOME=" & dllApp.sddHome & vbcrLf)
Call WriteTextToFile(textFile, "SDD_PLATFORM=" & dllApp.sddPlatform & vbcrLf)
Call WriteTextToFile(textFile, vbcrLf)

' Specific environment -- set to PADS VX.2.5, write the parameters
' to a text file
Call dllApp.SetEnvironment("c:\\MentorGraphics\\\\PADSVX.2.5\\\\SDD_HOME")
Call WriteTextToFile(textFile, "Specific Environment:" & vbcrLf)
Call WriteTextToFile(textFile, "-----" & vbcrLf)
Call WriteTextToFile(textFile, "PROG_ID_VER=" & dllApp.ProgIDVersion & vbcrLf)
Call WriteTextToFile(textFile, "SDD_VERSION=" & dllApp.sddVersion & vbcrLf)
Call WriteTextToFile(textFile, "SDD_HOME=" & dllApp.sddHome & vbcrLf)
Call WriteTextToFile(textFile, "SDD_PLATFORM=" & dllApp.sddPlatform & vbcrLf)
Call WriteTextToFile(textFile, vbcrLf)

If (Err) Then
MsgBox Err.Description
Err.Clear
End If

' Get all PCB releases on this machine and output them to a text file.
installedReleases = dllApp.GetInstalledReleases
Call WriteTextToFile(textFile, "List of installed releases:" & vbcrLf)
Call WriteTextToFile(textFile, "-----" & vbcrLf)
WriteReleaseInformationFile(installedReleases)
Call WriteTextToFile(textFile, vbcrLf)

If (Err) Then
MsgBox Err.Description
Err.Clear
End If
```

```
Function WriteTextToFile(fileName, text)
    Dim FSO
    Set FSO = CreateObject("Scripting.FileSystemObject")
    Dim file
    Set file = FSO.OpenTextFile(fileName, 8,true)
    Call file.Write(text)
    Call file.Close()
End Function

Sub WriteReleaseInformationFile(arrResults)
    Dim str: str = ""
    Dim i,j

    For i = 0 To UBound(arrResults,1)
        For j = 0 To UBound(arrResults,2)
            'wrap token in "" and add a ,
            str = str & """" & arrResults(i, j) & """" & ","
        Next
        str = str & vbCrLf 'add newline after each release/row
    Next
    Call WriteTextToFile(textFile, str)
End Sub
```

Related Topics

[Identifying the COM Version Number of an Install](#)

[ReleaseEnvServer Object](#)

ReleaseEnvServer Object

The ReleaseEnvServer object provides access to the PCB software install environment. The COM version of this object does not change.

Table 2-1. ReleaseEnvServer Object Methods and Properties

Method or Property	Description
GetInstalledReleases Method (ReleaseEnvServer Object)	Returns a two-dimensional variant array of releases on the current machine.
SetEnvironment Method (ReleaseEnvServer Object)	Sets the environment. If you do not include the SDD_HOME argument, the default (last installed) environment is set.
ProgIDVersion Property (ReleaseEnvServer Object)	Returns the COM version of the current environment.
sddHome Property (ReleaseEnvServer Object)	Returns the SDD_HOME path of the current environment.
sddPlatform Property (ReleaseEnvServer Object)	Returns the SDD_PLATFORM of the current environment.
sddVersion Property (ReleaseEnvServer Object)	Returns the SDD_VERSION of the current environment.

GetInstalledReleases Method (ReleaseEnvServer Object)

Object: [ReleaseEnvServer Object](#)

Returns a two-dimensional variant array of releases on the current machine.

Usage

ReleaseEnvServer.GetInstalledReleases()

Arguments

None.

Return Values

Variant array contains a header row of four strings. Each subsequent row lists string values for each installed release. For example, if two releases are installed on the machine the variant array is:

COM_VERSION	SDD_HOME	SDD_PLATFORM	SDD_VERSION
9	C:\MentorGraphics\EEVX.1\ SDD_HOME	win32	EEVX.1
3	C:\MentorGraphics\ PADSProVX.1\SDD_HOME	win32	PADSProVX.1

Examples

```
Dim installedReleases
installedReleases = ReleaseEnvServer.GetInstalledReleases
WriteReleaseInformationFile(installedReleases)
...
Sub WriteReleaseInformationFile(arrResults)
' This function wraps each array element in quotes.
' Each string token is followed by a comma (,).
' Each "row" adds a newline.
' The str is output to a text file.
,
    Dim str: str = ""
    Dim i,j

    For i = 0 To UBound(arrResults,1)
        For j = 0 To UBound(arrResults,2)
            str = str & "\"" & arrResults(i, j) & "\"" & ","
        Next
        str = str & vbCrLf
    Next

    Call WriteTextToFile(textFile, str)
End Sub
```

The resulting textfile:

```
"COM_VERSION","SDD_HOME","SDD_PLATFORM","SDD_VERSION",  
"9","C:\MentorGraphics\EEVX.1\SDD_HOME","win32","EEVX.1",  
"3","C:\MentorGraphics\PADSProVX.1\SDD_HOME","win32","PADSProVX.1",
```

SetEnvironment Method (ReleaseEnvServer Object)

Object: [ReleaseEnvServer Object](#)

Sets the environment. If you do not include the SDD_HOME argument, the default (last installed) environment is set.

Usage

ReleaseEnvServer.SetEnvironment ([ByVal strSDD_HOME As Nothing]) As Nothing

Arguments

- strSDD_HOME
(Optional) A string that include the SDD_HOME path.

Return Values

Variant array contains a header row of four strings. Each subsequent row lists string values for each installed release. For example, if two releases are installed on the machine the variant array is:

COM_VERSION	SDD_HOME	SDD_PLATFORM	SDD_VERSION
9	C:\MentorGraphics\EEVX.1\ SDD_HOME	win32	EEVX.1
3	C:\MentorGraphics\ PADSProVX.1\SDD_HOME	win32	PADSProVX.1

Examples

```
' Set the default environment -- no parameters
dllApp.SetEnvironment("")

' Set the X-ENTP VX.1 environment.
dllApp.SetEnvironment("c:\\MentorGraphics\\EEVX.1\\SDD_HOME")
```

ProgIDVersion Property (ReleaseEnvServer Object)

Object: [ReleaseEnvServer Object](#)

Access: Read-Only

Returns the COM version of the current environment.

Usage

ReleaseEnvServer.ProgIDVersion = String

Arguments

None.

Return Values

String that contains the COM version of the install environment.

sddHome Property (ReleaseEnvServer Object)

Object: [ReleaseEnvServer Object](#)

Access: Read-Only

Returns the SDD_HOME path of the current environment.

Usage

ReleaseEnvServer.sddHome = String

Arguments

None.

Return Values

String that contains the current SDD_HOME path.

sddPlatform Property (ReleaseEnvServer Object)

Object: [ReleaseEnvServer Object](#)

Access: Read-Only

Returns the SDD_PLATFORM of the current environment.

Usage

ReleaseEnvServer.sddPlatform = String

Arguments

None.

Return Values

String that contains current platform.

sddVersion Property (ReleaseEnvServer Object)

Object: [ReleaseEnvServer Object](#)

Access: Read-Only

Returns the SDD_VERSION of the current environment.

Usage

ReleaseEnvServer.sddVersion = String

Arguments

None.

Return Values

String that contains SDD Version number.

Release Environment Server Enumerated Types

This section contains the alphabetical listing of the Release Environment Server enumerated types. Refer to each type to determine the constants available.

Table 2-2. Release Environment Server Enumerated Types

Enumerated Type	Description
EReleaseEnvErrCode Enum	Release Environment Server error codes.

EReleaseEnvErrCode Enum

Prerequisites: None.

Release Environment Server error codes.

Usage

EReleaseEnvErrCode.Constant

Arguments

Constant	Value	Description
eReleaseEnvErrCodeEnvLibraryNotFound	-2147220989 (&H80040203)	The MGC PCB environment library for this <32/64 bit>application is not found in the selected SDD_HOME tree. Select a valid <32/64 bit> installation.
eReleaseEnvErrCodeEnvWrongLibVersion	-2147220988 (&H80040204)	The <32/64 bit> MGC PCB environment library could not be found.
eReleaseEnvErrCodeIncorrectPLATFORM	-2147220990 (&H80040202)	The default MGC PCB environment is <win32/win64>. This application cannot configure a <win32/win64> environment. Select a valid <32/64 bit> installation, or switch to a <32/64 bit> application.
eReleaseEnvErrCodeIncorrectSDD_HOME	-2147220991 (&H80040201)	Invalid SDD_HOME specified.
eReleaseEnvErrCodeInvalidParameter	-2147220992 (&H80040200)	Invalid parameter.
eReleaseEnvErrCodeNoDefaultRelease	-2147220985 (&H80040207)	There is no default MGC PCB release registered. Register a release, or provide a valid SDD_HOME.
eReleaseEnvErrCodeRelLibraryNotFound	-2147220987 (&H80040205)	This version of the MGC PCB Release Reader library is not able to get the list of registered installs. Use a more recent installation, or upgrade to a newer version of the Release Reader library.

Constant	Value	Description
eReleaseEnvErrCodeRelWrongLibVersion	-2147220986 (&H80040206)	The <32/64 bit> MGC PCB Release Reader library could not be found.

Chapter 3

Scripting Server

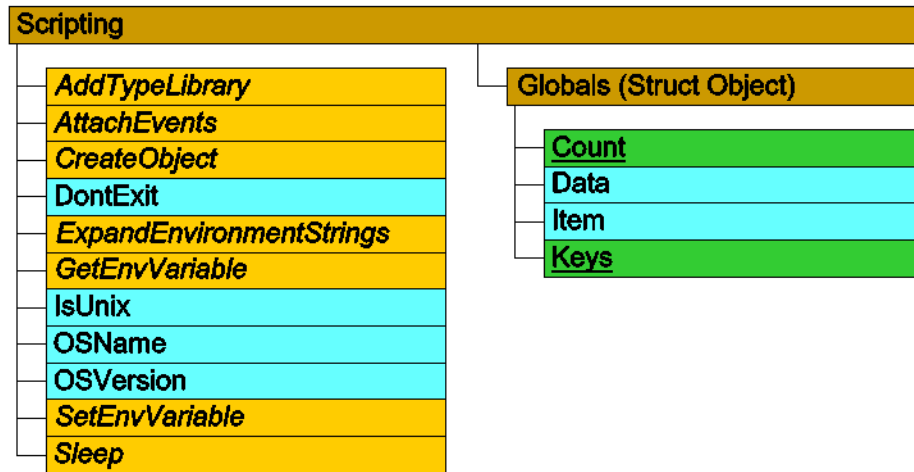
Scripting Object server is available to all applications. That is, you do not need any special calls in your script to include this server.

Scripting Object Data Model	29
Program IDs for Applications and Automation Engines	30
Scripting Object	32
AddTypeLibrary Method (Scripting Object)	33
AttachEvents Method (Scripting Object)	34
CreateObject Method (Scripting Object)	35
DetachEvents Method (Scripting Object)	36
ExpandEnvironmentStrings Method (Scripting Object)	37
GetEnvVariable Method (Scripting Object)	38
SetEnvVariable Method (Scripting Object)	39
Sleep Method (Scripting Object)	40
DontExit Property (Scripting Object)	41
Globals Property (Scripting Object)	42
IsUnix Property (Scripting Object)	43
OSName Property (Scripting Object)	44
OSVersion Property (Scripting Object)	45
Struct Object	46
Count Property (Struct Object)	47
Data Property (Struct Object)	48
Item Property (Struct Object)	49
Keys Property (Struct Object)	50

Scripting Object Data Model

The following figure shows the Scripting Data model.

Figure 3-1. Scripting Object Data Model



Program IDs for Applications and Automation Engines

The following table provides a summary of all automation applications and their corresponding programs IDs.

Table 3-1. Program IDs for Applications and Automation Engines

Application/Engine	Program ID
Automation Licensing	MGCPBAutomationLicensing.Application
Board Station RE	MGCPB.BoardStationREApplication
Board Station XE	MGCPB.BoardStationXEApplication
Cell Editor	CellEditorAddin.CellEditorDlg
DFFDRC	MGCPBEngines.DFFDRC
DRC	MGCPBEngines.DRC
DXFExport	MGCPBEngines.DXFExport
EDM Library Tools	LibraryManager.Application
ExtendedPrint	MGCPBEngines.ExtendedPrint
GeneralInterfaceEngine	MGCPBEngines.GeneralInterfaceEngine
Gerber	MGCPBEngines.Gerber
ICX Pro Verify	MGCPB.ICXProVerifyApplication
IDFExport	MGCPBEngines.IDFExport

Table 3-1. Program IDs for Applications and Automation Engines (cont.)

ManufacturingValidationOutput	MGCPBEngines.ManufacturingOutputValidation
NCDrill	MGCPBEngines.NCDrill
NeutralFileExport	MGCPBEngines.NeutralFileExport
ODBPPOutputEngine	MGCPBEngines.ODBPPOutputEngine
PADS Designer	ViewDraw.Application
PADS Layout	PowerPCB.Application
PADS Logic	PowerLogic.Application
PADS Router	BlazeRouter.Application
Padstack Editor	MGCPBLibraries.PadstackEditorDlg
Part Editor	MGCPBLibraries.PartsEditorDlg
PDFOutput	MGCPBEngines.PDFOutput
SFX RE	MGCPB.SFXREApplication
SilkscreenGen	MGCPBEngines.SilkscreenGen
Xpedition Designer	ViewDraw.Application
Xpedition FabLink	MGCPB.FablinkXEApplication
Xpedition FabLink Drawing Editor	MGCPB.DrawingEditorApplication
Xpedition FabLink Drawing Wizard	MGCPBEngines.DrawingFileWizard
Xpedition Layout	MGCPB.ExpeditionPCBApplication
Xpedition Layout Planner	MGCPB.PlannerPCBApplication
Xpedition Layout Viewer	MGCPB.ViewerPCBApplication

Scripting Object

The Scripting object provides a means of attaching events to dynamically created COM objects. It also provides other useful properties and methods when running script with **mgscript** command. The Scripting object is automatically created and added to the currently running script as a variable named *Scripting*.

Table 3-2. Scripting Object Methods and Properties

Method or Property	Description
AddTypeLibrary Method (Scripting Object)	Binds constants (enums) that the specified Type Library defines, into the currently running script.
AttachEvents Method (Scripting Object)	Attaches an object's event sources to script functions with a given prefix.
CreateObject Method (Scripting Object)	Create and return a reference to an Automation Object. If the new object has an associated type library, the method automatically adds the type library to the script.
DetachEvents Method (Scripting Object)	Detaches an object's event sources from script functions with a given prefix.
ExpandEnvironmentStrings Method (Scripting Object)	Translates an environment variable string.
GetEnvVariable Method (Scripting Object)	Translate Environment variable.
SetEnvVariable Method (Scripting Object)	Set Environment variable. This will only set the value for the current process.
Sleep Method (Scripting Object)	Suspend script execution.
DontExit Property (Scripting Object)	Set to true to keep the script from exiting after the last statement executes.
Globals Property (Scripting Object)	Returns global name/key - value pairs. Use this property for script-to-script communications.
IsUnix Property (Scripting Object)	Indicates whether or not the script is running on a Unix based system.
OSName Property (Scripting Object)	Sets or returns the Operating System name string.
OSVersion Property (Scripting Object)	Sets or returns the Operating System version string.

AddTypeLibrary Method (Scripting Object)

Object: [Scripting Object](#)

Binds constants (enums) that the specified Type Library defines, into the currently running script.

Usage

```
Scripting.AddTypeLibrary (ByVal TypeGUIDorProgID As String,  
    [ByVal MajorVersion As Variant],  
    [ByVal MinorVersion As Variant]) As Boolean
```

Arguments

- **TypeGUIDorProgID**
String specifying either a Type Library GUID in the form {8digits-4digits-4digits-4digits-12digits} or a ProgID. For more information about the program IDs, see [Program IDs for Applications and Automation Engines](#).
- **MajorVersion**
(Optional) Parameter specifying the Major version as found in the type library. The default value is 1.
- **MinorVersion**
(Optional) Parameter specifying the Minor version as found in the type library. The default value is 0.

Return Values

Boolean indicating of the method succeeded (True) or failed (False).

Examples

```
' Use any constants defined in MGCPB Automation layer  
Call Scripting.AddTypeLibrary("MGCPB.Application")
```

AttachEvents Method (Scripting Object)

Object: [Scripting Object](#)

Attaches an object's event sources to script functions with a given prefix.

Usage

Scripting.AttachEvents (ByVal Object As Object, ByVal ObjectName As String) As Boolean

Arguments

- **Object**
The object that you want to attach to.
- **ObjectName**
Prefix string used to map the object events for this instance. When the object fires an event named Foo, the script engine calls a subroutine named EventPrefix_Foo.

Return Values

Boolean indicating if the method succeeded (True) or failed (False).

Examples

```
' Attach application object to its event handlers with prefix appEvents
Dim app
Set app = GetObject(, "MGCPCB.Application")
Call Scripting.AttachEvents(app, "appEvents")
```

CreateObject Method (Scripting Object)

Object: [Scripting Object](#)

Create and return a reference to an Automation Object. If the new object has an associated type library, the method automatically adds the type library to the script.

Usage

Scripting.CreateObject (ByVal Class As String, [ByVal EventPrefix As Variant]) As Object

Arguments

- **Class**
ProgID in the form Servername.TypeName that specifies the object to create. For more information about the program IDs, see [Program IDs for Applications and Automation Engines](#).
- **EventPrefix**
(Optional) Optional argument used to bind to the outgoing interface of the object. If specified, the script engine connects the object's outgoing interface to the script file after creating the object. When the object fires an event, the script engine calls a subroutine named EventPrefix and the event name. For example, if EventPrefix is "MyObj" and the object fires an event named "OnBegin," The scripting engine calls the "MyObj_OnBegin" subroutine located in the script. The underscore is automatically added between then EventPrefix and the Event name.

Return Values

String naming the newly created object.

Examples

```
' Create a PADS Layout application object
Dim appPCB
Set appPCB = Scripting.CreateObject("PowerPCB.Application.78")
```

Related Topics

[Scripting With Multiple Installs](#)

DetachEvents Method (Scripting Object)

Object: [Scripting Object](#)

Detaches an object's event sources from script functions with a given prefix.

Usage

Scripting.DetachEvents (ByVal Object As Object, ByVal ObjectName As String) As Boolean

Arguments

- **Object**
The object from which you want to detach.
- **ObjectName**
Prefix string used to map the object events for this instance. When the object fires an event named Foo, the script engine calls a subroutine named EventPrefix_Foo.

Return Values

Boolean indicating whether the method succeeded (True) or failed (False).

Examples

```
' Detach application object from its event handlers with prefix appEvents
Dim app
Set app = GetObject(, "MGCPCB.Application")
Call Scripting.DetachEvents(app, "appEvents")
```

ExpandEnvironmentStrings Method (Scripting Object)

Object: [Scripting Object](#)

Translates an environment variable string.

Usage

Scripting.ExpandEnvironmentStrings (ByVal InputStr As String) As String

Arguments

- InputStr
String containing %VAR%s to translate.

Return Values

Translated string or an empty string if not found.

GetEnvVariable Method (Scripting Object)

Object: [Scripting Object](#)

Translate Environment variable.

Usage

Scripting.GetEnvVariable (ByVal VariableName As String) As String

Arguments

- VariableName
The name of the Environment variable to translate.

Return Values

Translated String or an empty string if not found.

SetEnvVariable Method (Scripting Object)

Object: [Scripting Object](#)

Set Environment variable. This will only set the value for the current process.

Usage

```
Scripting.SetEnvVariable (ByVal VariableName As String, ByVal Value As String) As  
Boolean
```

Arguments

- **VariableName**
Name of the environment variable.
- **Value**
Value of the environment variable.

Return Values

Boolean indicating of the method succeeded (True) or failed (False).

Sleep Method (Scripting Object)

Object: [Scripting Object](#)

Suspend script execution.

Usage

Scripting.Sleep (ByVal nMilliseconds As Long)

Arguments

- nMilliseconds
Time in milliseconds to suspend script execution.

DontExit Property (Scripting Object)

Object: [Scripting Object](#)

Access: Read/Write

Set to true to keep the script from exiting after the last statement executes.

Usage

Scripting.DontExit = True | False

Arguments

None.

Return Values

Boolean, if True, keep the script from exiting after the last statement executes. If False, exit the script after the last statement executes.

Globals Property (Scripting Object)

Object: [Scripting Object](#)

Access: Read-Only

Returns global name/key - value pairs. Use this property for script-to-script communications.

Note



Multiple global variables are supported. The number of global variables is limited only by the available memory.

Usage

Scripting.Globals

Arguments

None

Return Values

Object. The [Scripting Object](#).

Examples

To store a global name/key - value pair:

```
Scripting.Globals("MyGlobalName") = "MyGlobalValue"
```

To read from a global name/key -value pair:

```
MsgBox Scripting.Globals("MyGlobalName")  
' Returns "MyGlobalValue"
```

To store a global object:

```
Scripting.Globals.Data("MyGlobalObjName") = myObj
```

To output the global object's name:

```
MsgBox Scripting.Globals("MyGlobalObjName").Name  
' Returns the value of myObj.Name
```

To iterate the keys:

```
For Each key In Scripting.Globals.Keys  
    MsgBox key & "=" & Scripting.Globals(key)  
Next
```

IsUnix Property (Scripting Object)

Object: [Scripting Object](#)

Access: Read-Only

Indicates whether or not the script is running on a Unix based system.

Usage

Scripting.IsUnix = True | False

Arguments

None.

Return Values

Boolean. If True, the system is UNIX-based. If False, the system is not UNIX-based.

OSName Property (Scripting Object)

Object: [Scripting Object](#)

Access: Read/Write

Sets or returns the Operating System name string.

Usage

Scripting.OSName = String

Arguments

None.

Return Values

A string that contains the name of the operating system.

OSVersion Property (Scripting Object)

Object: [Scripting Object](#)

Access: Read/Write

Sets or returns the Operating System version string.

Usage

Scripting.OSVersion = String

Arguments

None.

Return Values

A string that contains the version of the operating system.

Struct Object

The Struct object holds Key - Value pairs. This object returns from the Globals Property (Scripting Object).

The Struct object is a static object in the context of a process space. Therefore, any in-process script or form that accesses the Struct object is accessing the same object. This means that one script can write data to the object and a different script can read the same data.

The [Globals Property \(Scripting Object\)](#) works only in the current process space.

Table 3-3. Struct Object Properties

Property	Description
Count Property (Struct Object)	Returns the number of items in the struct object.
Data Property (Struct Object)	Sets or returns an object.
Item Property (Struct Object)	Sets or returns a value based on a name/key.
Keys Property (Struct Object)	Returns the collection of names/keys.

Count Property (Struct Object)

Object: [Struct Object](#)

Access: Read-Only

Returns the number of items in the struct object.

Usage

Struct.Count

Arguments

None.

Return Values

Long. The number of items.

Data Property (Struct Object)

Object: [Struct Object](#)

Access: Read/Write

Sets or returns an object.

Usage

Stuct.Data (ByVal Name As String) = Object

Arguments

- Name
A string that contains the key/name.

Return Values

Object. The object associated with key/name.

Examples

To store a global object:

```
Set Scripting.Globals("MyGlobalObjName").Data = myObj
```

To output the global object's name:

```
MsgBox Scripting.Globals("MyGlobalObjName").Name  
' Returns the value of myObj.Name
```


Item Property (Struct Object)

Object: [Struct Object](#)

Access: Read/Write

Sets or returns a value based on a name/key.

Usage

Stuct.Item(ByVal Index As Variant) = Value

Arguments

- Index
A name or key.

Return Values

The value of the name/key. The value can be any type.

Keys Property (Struct Object)

Object: [Struct Object](#)

Access: Read-Only

Returns the collection of names/keys.

Usage

Struct.Keys

Arguments

None.

Return Values

The collection of keys.

Examples

To iterate the keys:

```
For Each key In Scripting.Globals.Keys  
    MsgBox key & "=" & Scripting.Globals(key)  
Next
```

Chapter 4

CommandBar Server

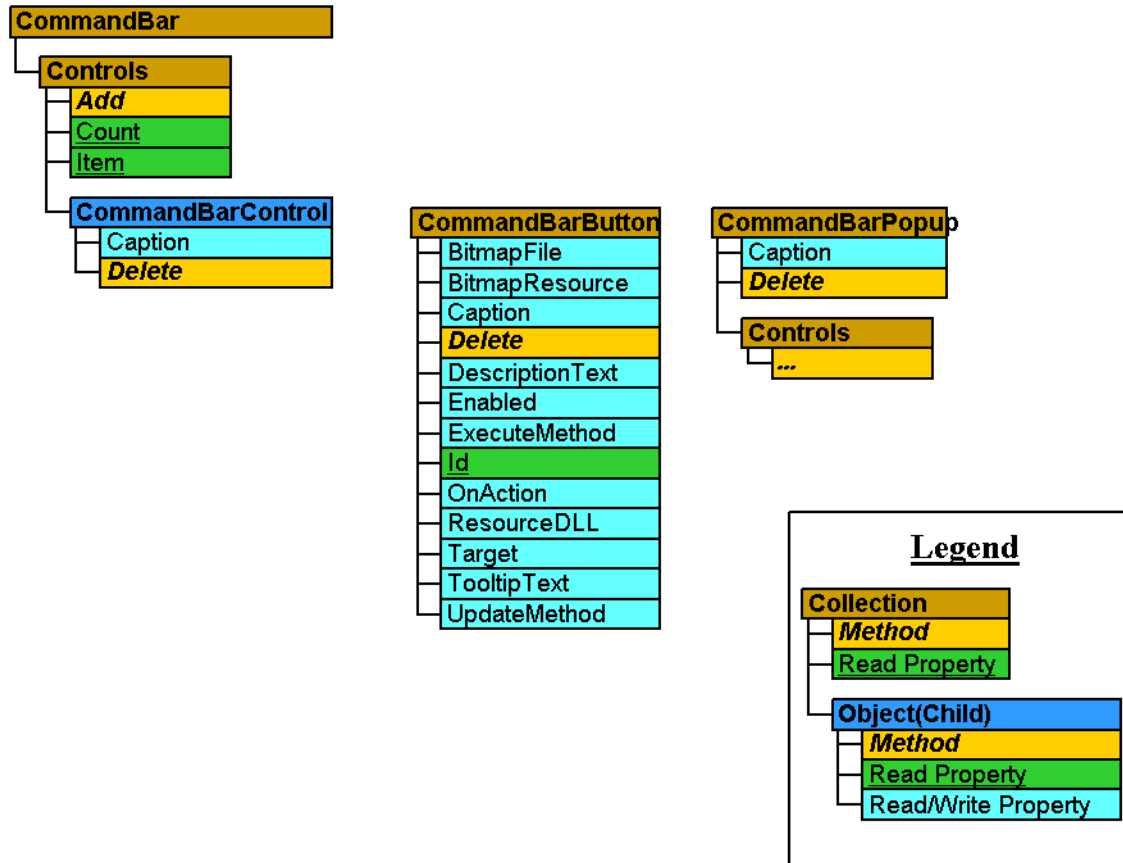
The CommandBar Server lets you to customize the toolbars and menus of applications.

CommandBarServer Data Model	52
CommandBar Object	53
Controls Property (CommandBar Object)	54
CommandBarButton Object	55
Delete Method (CommandBarButton Object)	57
BitmapFile Property (CommandBarButton Object)	58
BitmapResource Property (CommandBarButton Object)	59
Caption Property (CommandBarButton Object)	60
DescriptionText Property (CommandBarButton Object)	61
Enabled Property (CommandBarButton Object)	62
ExecuteMethod Property (CommandBarButton Object)	63
Id Property (CommandBarButton Object)	64
OnAction Property (CommandBarButton Object)	66
ResourceDLL Property (CommandBarButton Object)	67
Target Property (CommandBarButton Object)	68
TooltipText Property (CommandBarButton Object)	69
UpdateMethod Property (CommandBarButton Object)	70
CommandBarControl Object	71
Delete Method (CommandBarControl Object)	72
Caption Property (CommandBarControl Object)	73
CommandBarPopup Object	74
Delete Method (CommandBarPopup Object)	75
Caption Property (CommandBarPopup Object)	76
Controls Property (CommandBarPopup Object)	77
CommandBarControls Collection	78
Add Method (CommandBarControls Collection)	79
Count Property (CommandBarControls Collection)	80
Item Property (CommandBarControls Collection)	81
CommandBars Collection	82
Item Property (CommandBars Collection)	83
Command Bar Enumerated Types	84
CmdControlType Enum	85

CommandBarServer Data Model

The following figure shows the CommandBarServer Data model.

Figure 4-1. CommandBarServer Data Model



CommandBar Object

Lists the CommandBarControl properties.

Table 4-1. CommandBar Object Properties

Property	Description
Controls Property (CommandBar Object)	Returns the collection of command bar controls of the command bar object.

Controls Property (CommandBar Object)

Object: [CommandBar Object](#)

Access: Read-Only

Returns the collection of command bar controls of the command bar object.

Usage

CommandBar.Controls

Arguments

None.

Return Values

The collection of command bar controls ([CommandBarControls](#)) of the command bar.

CommandBarButton Object

Lists the CommandBarButton methods and properties.

Table 4-2. CommandBarButton Object Methods and Properties

Method or Property	Description
Delete Method (CommandBarButton Object)	Delete the command bar button in the corresponding command bar.
BitmapFile Property (CommandBarButton Object)	Sets or returns the bitmap file used by the commandbar button. The bitmap should be 16 x 16.
BitmapResource Property (CommandBarButton Object)	Sets or returns the id of the bitmap resource to load. It is loaded from the ResourceDLL.
Caption Property (CommandBarButton Object)	Sets or returns the caption text of the commandbar button. The caption text is used as the commandbar button's ScreenTip.
DescriptionText Property (CommandBarButton Object)	Sets or returns the description text of the commandbar button. The text is displayed in the status bar.
Enabled Property (CommandBarButton Object)	Enable or disable the commandbar button. Only user-defined command can be enabled / disabled by this property.
ExecuteMethod Property (CommandBarButton Object)	Sets or returns the procedure associated with the commandbar button.
Id Property (CommandBarButton Object)	Returns the ID of the command associated with the commandbar button.
OnAction Property (CommandBarButton Object)	Sets or returns the action to be taken when the commandbar button is clicked.
ResourceDLL Property (CommandBarButton Object)	Sets or returns the .dll from which the bitmap resource for the button is loaded.
Target Property (CommandBarButton Object)	Sets or returns the target COM object associated with the commandbar button.

Table 4-2. CommandBarButton Object Methods and Properties (cont.)

Method or Property	Description
TooltipText Property (CommandBarButton Object)	Sets or returns the tooltip text of the commandbar button.
UpdateMethod Property (CommandBarButton Object)	Sets or returns the method when to enable or disable the commandbar button.

Delete Method (CommandBarButton Object)

Object: [CommandBarButton Object](#)

Delete the command bar button in the corresponding command bar.

Usage

CommandBarButton.Delete ()

Arguments

None.

BitmapFile Property (CommandBarButton Object)

Object: [CommandBarButton Object](#)

Access: Read/Write

Sets or returns the bitmap file used by the commandbar button. The bitmap should be 16 x 16.

Usage

CommandBarButton.BitmapFile = String

Arguments

None.

Return Values

A string that represents the full path of the bitmap file used by the commandbar button.

BitmapResource Property (CommandBarButton Object)

Object: [CommandBarButton Object](#)

Access: Read/Write

Sets or returns the id of the bitmap resource to load. It is loaded from the ResourceDLL.

Usage

CommandBarButton.BitmapResource = Long

Arguments

None.

Return Values

A long that contains the bitmap resource.

Description

If the ResourceDLL is not set, the bitmap is not loaded. For more information on resources and C++ clients for Automation see the MSDN site.

Caption Property (CommandBarButton Object)

Object: [CommandBarButton Object](#)

Access: Read/Write

Sets or returns the caption text of the commandbar button. The caption text is used as the commandbar button's ScreenTip.

Usage

CommandBarButton.Caption = String

Arguments

None.

Return Values

A string that represents the caption text of the commandbar button.

DescriptionText Property (CommandBarButton Object)

Object: [CommandBarButton Object](#)

Access: Read/Write

Sets or returns the description text of the commandbar button. The text is displayed in the status bar.

Usage

CommandBarButton.DescriptionText = String

Arguments

None.

Return Values

A string that represents the description text of the command bar button.

Enabled Property (CommandBarButton Object)

Object: [CommandBarButton Object](#)

Access: Read/Write

Enable or disable the commandbar button. Only user-defined command can be enabled / disabled by this property.

Usage

CommandBarButton.Enabled = True | False

Arguments

None.

Return Values

Boolean. If True, the command bar button is enabled. If False, the command bar button is disabled.

ExecuteMethod Property (CommandBarButton Object)

Object: [CommandBarButton Object](#)

Access: Read/Write

Sets or returns the procedure associated with the commandbar button.

Usage

CommandBarButton.ExecuteMethod = String

Arguments

None.

Return Values

A string that represents the procedure to be run when the button is clicked.

Examples

```
'  
' This example add a menu entry to execute a method defined in the script  
'  
' With the menu controls collection  
Set button = myMenuCtrls.Add  
button.Caption = "Zoom Selected"  
button.Target = ScriptEngine  
button.ExecuteMethod = "myZoomSelectedMethod"  
' call function below with this name  
button.DescriptionText = "Zoom around selected objects"  
  
Sub myZoomSelectedMethod( )  
    ' zoom around selected objects  
    doc.ActiveViewEx.SetExtentsToSelection  
End Sub
```

Id Property (CommandBarButton Object)

Object: [CommandBarButton Object](#)

Access: Read-Only

Returns the ID of the command associated with the commandbar button.

Usage

CommandBarButton.Id

Arguments

None.

Return Values

A long that represents the ID of command associated with the commandbar button.

Examples

```
'
' This example walks through the menus of a PCB application and
' write the menu name and its command ID out to a output file.
'
' Create an output file
Set filesys = CreateObject("Scripting.FileSystemObject")
file = "c:\temp\output.txt"
Set filetxt = filesys.CreateTextFile(file, True)

' Get the document menu bar object
Dim docMenuBar
Set docMenuBar = Gui.CommandBars("Document Menu Bar")

' Walk through all menu in the menu bar
' and write out its name and command id
' to a file
xTab = vbTab
For i = 1 To docMenuBar.Controls.Count
    Set menu = docMenuBar.Controls.Item(i)
    filetxt.WriteLine "+" & menu.Caption
    Call WriteMenuIDs(menu)
    filetxt.WriteLine
Next
' Subroutine to write out menu item name
' and its command ID
Sub WriteMenuIDs(menu)
    Set menuCtrls = menu.Controls
    For j = 1 To menuCtrls.Count
        cmdName = menuCtrls.Item(j).Caption
        On Error Resume Next
        id = menuCtrls.Item(j).Id
        If Err Then
            ' CommandBarPopup doesn't support Id property
            Err.Clear
            filetxt.WriteLine xTab & cmdName
            saveTab = xTab
            xTab = xTab & vbTab
            Call WriteMenuIDs(menuCtrls.Item(j))
            xTab = saveTab
        ElseIf id <> 0 Then
            ' Don't write out separator whose command id is 0
            filetxt.WriteLine xTab & cmdName & " : " & id
        End If
    Next
End Sub
MsgBox "fini"
```

OnAction Property (CommandBarButton Object)

Object: [CommandBarButton Object](#)

Access: Read/Write

Sets or returns the action to be taken when the commandbar button is clicked.

Usage

CommandBarButton.OnAction = String

Arguments

None.

Return Values

A string that represents the action to be taken when the commandbar button is clicked.

Description

This property is used for creating a new menu entry or toolbar button associated with a PCB application built-in command, or key-in command.

Examples

```
'  
' This example adds a menu entry using built-in command, Highlight  
'  
' With the menu object (CommandBar) defined  
Set cntrls = menu.Controls  
Set button = cntrls.Add  
button.Caption = "&Highlight"  
button.OnAction = "32867"      ' 32867 is the ID of the built-in command,  
Highlight
```

ResourceDLL Property (CommandBarButton Object)

Object: [CommandBarButton Object](#)

Access: Read/Write

Sets or returns the *.dll* from which the bitmap resource for the button is loaded.

Usage

CommandBarButton.ResourceDLL = String

Arguments

None.

Return Values

A string that contains the name of the *.dll*.

Description

This property is used in conjunction with the [BitmapResource](#) property.

Target Property (CommandBarButton Object)

Object: [CommandBarButton Object](#)

Access: Read/Write

Sets or returns the target COM object associated with the commandbar button.

Usage

CommandBarButton.Target = COM Object

Arguments

- COM Object.

The name of the COM object associated with the commandbar button.

Return Values

Examples

```
'  
' This example add a menu entry to execute a method defined in the script  
'  
' With the menu controls collection  
Set button = myMenuCtrls.Add  
button.Caption = "Zoom Selected"  
'The target COM object is this scripting engine  
button.Target = ScriptEngine  
button.ExecuteMethod = "myZoomSelectedMethod"  
button.DescriptionText = "Zoom around selected objects"  
  
Sub myZoomSelectedMethod( )  
    ' zoom around selected objects  
    doc.ActiveViewEx.SetExtentsToSelection  
End Sub
```

TooltipText Property (CommandBarButton Object)

Object: [CommandBarButton Object](#)

Access: Read/Write

Sets or returns the tooltip text of the commandbar button.

Usage

CommandBarButton.TooltipText = String

Arguments

None.

Return Values

A string that represents the tooltip text of the command bar button.

UpdateMethod Property (CommandBarButton Object)

Object: [CommandBarButton Object](#)

Access: Read/Write

Sets or returns the method when to enable or disable the commandbar button.

Usage

CommandBarButton.UpdateMethod = String

Arguments

None.

Return Values

A string that represents the method when to enable or disable the commandbar button.

Examples

```
'
' This example add a menu entry to execute a key-in command, "pr *" with a
' user-defined update method to enable/disable the menu entry
'
' With the menu controls collection object (CommandBarControls) defined
Set button = myMenuCtrls.Add
button.Caption = "Place Unplaced Parts"
button.OnAction = "pr *"
button.Target = ScriptEngine
button.UpdateMethod = "onUpdatePlaceUnplaced"
' call update method below with this name

Function onUpdatePlaceUnplaced(nID)
    ' Disable the command if no unplaced parts
    If doc.components(epcbSelectUnplaced).count > 0 Then
        onUpdatePlaceUnplaced = true
    Else
        onUpdatePlaceUnplaced = false
    End If
End Function
```

CommandBarControl Object

Lists the CommandBarControl methods and properties.

Table 4-3. CommandBarControl Object Methods and Properties

Method or Property	Description
Delete Method (CommandBarControl Object)	Deletes the command bar control in the corresponding command bar.
Caption Property (CommandBarControl Object)	Sets or returns the caption text of the command bar control.

Delete Method (CommandBarControl Object)

Object: [CommandBarControl Object](#)

Deletes the command bar control in the corresponding command bar.

Usage

CommandBarControl.Delete ()

Arguments

None.

Caption Property (CommandBarControl Object)

Object: [CommandBarControl Object](#)

Access: Read/Write

Sets or returns the caption text of the command bar control.

Usage

CommandBarControl.Caption = String

Arguments

None.

Return Values

A string that represents the caption text of the command bar control.

CommandBarPopup Object

Lists the CommandBarPopup methods and properties.

Table 4-4. CommandBarPopup Object Methods and Properties

Method or Property	Description
Delete Method (CommandBarPopup Object)	Delete the command bar popup in the corresponding command bar.
Caption Property (CommandBarPopup Object)	Sets or returns the caption text of the commandbar popup.
Controls Property (CommandBarPopup Object)	Returns the commandbar controls collection of the commandbar popup.

Delete Method (CommandBarPopup Object)

Object: [CommandBarPopup Object](#)

Delete the command bar popup in the corresponding command bar.

Usage

CommandBarPopup.Delete ()

Arguments

None.

Caption Property (CommandBarPopup Object)

Object: [CommandBarPopup Object](#)

Access: Read/Write

Sets or returns the caption text of the commandbar popup.

Usage

CommandBarPopup.Caption = String

Arguments

None.

Return Values

A string that represents the caption text of the commandbar popup.

Controls Property (CommandBarPopup Object)

Object: [CommandBarPopup Object](#)

Access: Read-Only

Returns the commandbar controls collection of the commandbar popup.

Usage

CommandBarPopup.Controls

Arguments

None.

Return Values

CommandBarControls. The collection of commandbar controls ([CommandBarControls](#)) of the commandbar popup.

CommandBarControls Collection

Lists the CommandBarControls methods and properties.

Table 4-5. CommandBarControls Collection Methods and Properties

Method or Property	Description
Add Method (CommandBarControls Collection)	Adds a commandbar control object to the collection.
Count Property (CommandBarControls Collection)	Returns the number of commandbar control objects contained in the commandbar controls collection.
Item Property (CommandBarControls Collection)	Returns a commandbar control object contained in the commandbar controls collection.

Add Method (CommandBarControls Collection)

Object: [CommandBarControls Collection](#)

Adds a commandbar control object to the collection.

Usage

```
CommandBarControls.Add (ByVal Type As Variant,  
    ByVal Id As Variant,  
    ByVal Parameter As Variant,  
    ByVal Before As Variant) As CommandBarControl
```

Arguments

- Type
The control type ([CmdControlType](#)) of the commandbar control.
- Id
(Optional) The ID of the command associated with the commandbar control. Omitted the parameter will have the commandbar server assign an ID for the control.
- Parameter
(Optional) Reserved for future use.
- Before
The position before which the new control is added.

Return Values

CommandBarControl. The object ([CommandBarControl](#)) to add to the collection.

Count Property (CommandBarControls Collection)

Object: [CommandBarControls Collection](#)

Access: Read/Write

Returns the number of commandbar control objects contained in the commandbar controls collection.

Usage

CommandBarControls.Count = Long

Arguments

None.

Return Values

Long. The number of commandbar control objects the collection contains.

Item Property (CommandBarControls Collection)

Object: [CommandBarControls Collection](#)

Access: Read-Only

Returns a commandbar control object contained in the commandbar controls collection.

Usage

CommandBarControls.Item (ByVal Index As Variant)

Arguments

- Index
Index of the object to retrieve from the collection

Return Values

CommandBarControl. The object ([CommandBarControl](#)) located at position Index.

CommandBars Collection

Lists the CommandBars properties.

Table 4-6. CommandBars Collection Properties

Property	Description
Item Property (CommandBars Collection)	Returns a commandbar object contained in the commandbars collection.

Item Property (CommandBars Collection)

Object: [CommandBars Collection](#)

Access: Read-Only

Returns a commandbar object contained in the commandbars collection.

Usage

CommandBars.Item (ByVal Index As Variant)

Arguments

- Index

Index of the CommandBar object to retrieve from the collection.

Return Values

The object ([CommandBar](#)) located at position Index.

Command Bar Enumerated Types

Lists the CommandBar enumerated types.

Table 4-7. CommandBar Enumerated Types

Enumerated Type	Description
CmdControlType Enum	Constants for CommandBarControls Add method.

CmdControlType Enum

Constants for CommandBarControls Add method.

Usage

CmdControlType.Constant

Arguments

- Constant

You can specify three types of constants:

cmdControlButton

The value for this constant is 0. Button control.

cmdControlButtonSeparator

The value for this constant is 2. Button Separator.

cmdControlPopup

The value for this constant is 1. Popup menu control.

Chapter 5

KeyBind Server

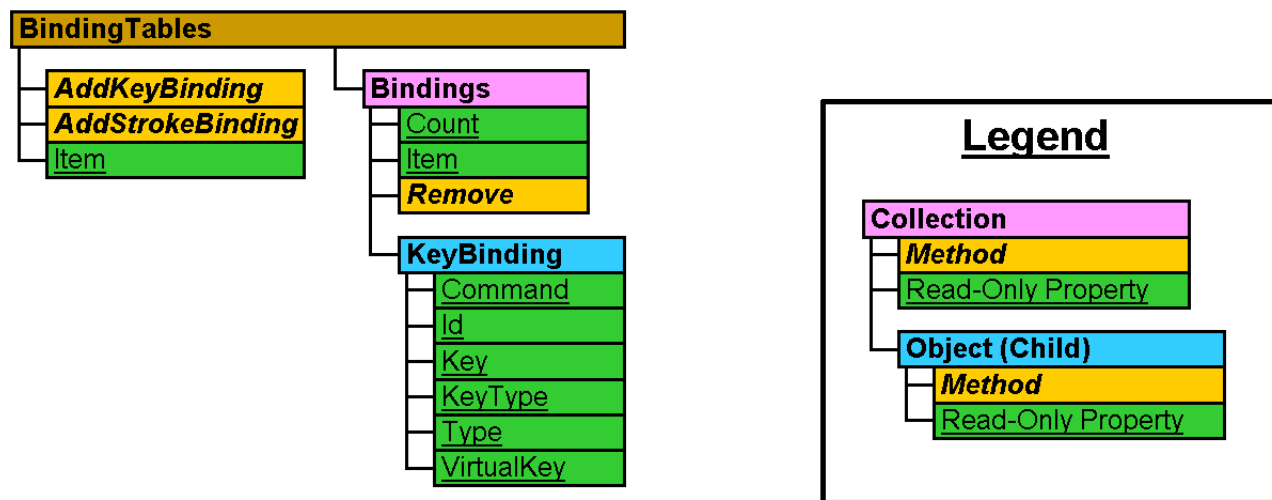
The KeyBind Server lets you to define custom shortcuts or key accelerators for applications.

KeyBindServer Data Model	87
KeyBinding Object	89
Command Property (KeyBinding Object)	90
ExecuteMethod Property (KeyBinding Object)	91
Id Property (KeyBinding Object)	92
Key Property (KeyBinding Object)	93
KeyType Property (KeyBinding Object)	94
Target Property (KeyBinding Object)	95
Type Property (KeyBinding Object)	96
VirtualKey Property (KeyBinding Object)	97
BindingTables Collection	98
AddKeyBinding Method (BindingTables Collection)	99
AddStrokeBinding Method (BindingTables Collection)	100
Bindings Property (BindingTables Collection)	102
Item Property (BindingTables Collection)	103
KeyBindings Collection	104
Remove Method (KeyBindings Collection)	105
Count Property (KeyBindings Collection)	106
Item Property (KeyBindings Collection)	107
Key Binding Enumerated Types	108
BindKeyType Enum	109
BindType Enum	110

KeyBindServer Data Model

The following figure shows the KeyBindServer Data model.

Figure 5-1. KeyBindServer Data Model



KeyBinding Object

Lists the KeyBinding properties.

Table 5-1. KeyBinding Object Properties

Property	Description
Command Property (KeyBinding Object)	Returns the menu command or key-in command associated with the keybinding object.
ExecuteMethod Property (KeyBinding Object)	Sets or returns the procedure associated with the keybinding object.
Id Property (KeyBinding Object)	Returns the ID of command associated with the keybinding object. (key-in command returns -1)
Key Property (KeyBinding Object)	Returns the key combination associated with the keybinding object.
KeyType Property (KeyBinding Object)	Returns the key type of the keybinding object.
Target Property (KeyBinding Object)	Sets or returns the target COM object associated with the key binding object.
Type Property (KeyBinding Object)	Returns the type of the keybinding object.
VirtualKey Property (KeyBinding Object)	Returns the ANSI number of the key associated with the keybinding object.

Command Property (KeyBinding Object)

Object: [KeyBinding Object](#)

Access: Read-Only

Returns the menu command or key-in command associated with the keybinding object.

Usage

KeyBinding.Command

Arguments

None.

Return Values

A string that represents the menu command or key-in command.

Examples

```
' With BindingTables object defined
' Assign shortcut keys Alt+F1 to Help->Contents->Automation
docBindingTables.AddKeyBinding "Alt+F1","Help->Contents->Automation",0,1

' With KeyBindings object defined
' Command property will return "Help->Contents->Automation"
MsgBox docKeyBindings.Item(1).Command
```

ExecuteMethod Property (KeyBinding Object)

Object: [KeyBinding Object](#)

Access: Read/Write

Sets or returns the procedure associated with the keybinding object.

Usage

KeyBinding.ExecuteMethod = String

Arguments

None.

Return Values

A string that represents the procedure to be run when the assigned accelerator key (or key combination) is pressed.

Examples

```
'  
' This example adds a key binding to execute a method defined in the  
' script  
'  
' Assign shortcut keys Alt+F1 to myZoomSelectedMethod  
Set bindObj = docBindingTables.AddKeyBinding ("Alt+F1", _  
    "myZoomSelectedMethod", BindFunction, BindAccelerator)  
  
' Associate the current script engine with the key binding  
bindObj.Target = ScriptEngine  
' Call method below with this name  
bindObj.ExecuteMethod = "myZoomSelectedMethod"  
' Keep this script running so that the handler can be executed  
Scripting.DontExit = True  
  
Sub myZoomSelectedMethod( )  
    ' Zoom around selected objects  
    doc.ActiveViewEx.SetExtentsToSelection  
End Sub
```

Id Property (KeyBinding Object)

Object: [KeyBinding Object](#)

Access: Read-Only

Returns the ID of command associated with the keybinding object. (key-in command returns -1)

Usage

KeyBinding.Id

Arguments

None.

Return Values

A long that represents the ID of the command associated with the keybinding object.

Examples

```
' With BindingTables object defined
' Assign shortcut keys Alt+F1 to Help->Contents->Automation
docBindingTables.AddKeyBinding "Alt+F1","Help->Contents->Automation",0,1

' With KeyBindings object defined
' Id property will return 33818, which is the internal command id.
MsgBox docKeyBindings.Item(1).Id
```

Key Property (KeyBinding Object)

Object: [KeyBinding Object](#)

Access: Read-Only

Returns the key combination associated with the keybinding object.

Usage

KeyBinding.Key

Arguments

None.

Return Values

A string that represents the shortcut keys combination associated with the keybinding object.

Examples

```
' With BindingTables object defined
' Assign shortcut keys Alt+F1 to Help->Contents->Automation
docBindingTables.AddKeyBinding "Alt+F1", "Help->Contents->Automation", 0, 1

' With KeyBindings object defined
' Key property will return Alt+F1
MsgBox docKeyBindings.Item(1).Key
```

KeyType Property (KeyBinding Object)

Object: [KeyBinding Object](#)

Access: Read-Only

Returns the key type of the keybinding object.

Usage

KeyBinding.KeyType

Arguments

None.

Return Values

A long that represents the key type ([BindKeyType](#)) of the keybinding object.

Examples

```
' With BindingTables object defined
' Assign shortcut keys Alt+F1 to Help->Contents->Automation
docBindingTables.AddKeyBinding "Alt+F1","Help->Contents->Automation",0,1

' With KeyBindings object defined
' KeyType property will return 1 which is BindAccelerator constant
MsgBox docKeyBindings.Item(1).KeyType
```

Target Property (KeyBinding Object)

Object: [KeyBinding Object](#)

Access: Read/Write

Sets or returns the target COM object associated with the key binding object.

Usage

KeyBinding.Target = COM Object

Arguments

- COM Object

The name of the COM object associated with the key binding.

Examples

```
'
' This example adds a key binding to execute a method defined in the
' script
'
' Assign shortcut keys Alt+F1 to myZoomSelectedMethod
Set bindObj = docBindingTables.AddKeyBinding ("Alt+F1", _
    "myZoomSelectedMethod", BindFunction, BindAccelerator)

' Associate the current script engine with the key binding
bindObj.Target = ScriptEngine
' Call method below with this name
bindObj.ExecuteMethod = "myZoomSelectedMethod"
' Keep this script running so that the handler can be executed
Scripting.DontExit = True

Sub myZoomSelectedMethod( )
    ' Zoom around selected objects
    doc.ActiveViewEx.SetExtentsToSelection
End Sub
```

Type Property (KeyBinding Object)

Object: [KeyBinding Object](#)

Access: Read-Only

Returns the type of the keybinding object.

Usage

KeyBinding.Type

Arguments

None.

Return Values

A long that represents the type ([BindType](#)) of the keybinding object.

Examples

```
' With BindingTables object defined
' Assign shortcut keys Alt+F1 to Help->Contents->Automation
docBindingTables.AddKeyBinding "Alt+F1","Help->Contents->Automation", 0,1
' With KeyBindings object defined
' Type property will return 0 which is BindMenu constant
MsgBox docKeyBindings.Item(1).Type
```


VirtualKey Property (KeyBinding Object)

Object: [KeyBinding Object](#)

Access: Read-Only

Returns the ANSI number of the key associated with the keybinding object.

Usage

KeyBinding.VirtualKey

Arguments

None.

Return Values

A long that represents the ANSI number of the key associated with the keybinding object.

Examples

```
' With BindingTables object defined
' Assign shortcut keys Alt+2 to View->FitAll command
docBindingTables.AddKeyBinding "Alt+2", "View->FitAll", 0, 1

' With KeyBindings object defined
' VirtualKey property will return 50 which is the ANSI number of key "2"
MsgBox docKeyBindings.Item(1).VirtualKey
```

BindingTables Collection

Lists the BindingTables methods and properties.

Table 5-2. BindingTables Collection Methods and Properties

Method or Property	Description
AddKeyBinding Method (BindingTables Collection)	Adds a shortcut key assignment to the keybindings collection.
AddStrokeBinding Method (BindingTables Collection)	Adds a Stroke to the KeyBindings collection.
Bindings Property (BindingTables Collection)	Returns the collection of keybinding objects of the BindingTables.
Item Property (BindingTables Collection)	Returns a keybinding object contained in the keybindings collection.

AddKeyBinding Method (BindingTables Collection)

Object: [BindingTables Collection](#)

Adds a shortcut key assignment to the keybindings collection.

Usage

```
BindingTables.AddKeyBinding (ByVal KeySequence As String,  
    ByVal Command As String,  
    ByVal BindType As Variant,  
    ByVal KeyType As Variant,  
    ByVal UserData As Variant) As IKeyBinding
```

Arguments

- **KeySequence**
A string that represents the shortcut keys combination.
- **Command**
A string that represents the menu command or key-in command assigned to the shortcut keys.
- **BindType**
A [BindType](#) constant that defines what command type the shortcut key is assigned to.
- **KeyType**
[BindKeyType](#) constant that defines the binding type of the shortcut keys.
- **UserData**
(Optional) Reserved for future use.

Return Values

IKeyBinding. The new [KeyBinding Object](#).

Examples

```
' Get the document BindingTables object  
Set docBindingTables = Gui.Bindings("Document")  
  
' Assign shortcut keys Alt+F1 to Help->Contents->Automation  
docBindingTables.AddKeyBinding "Alt+F1", "Help->Contents->Automation",  
BindMenu, BindAccelerator
```

AddStrokeBinding Method (BindingTables Collection)

Object: [BindingTables Collection](#)

Adds a Stroke to the KeyBindings collection.

Usage

```
BindingTables.AddStrokeBinding (ByVal Stroke As String,  
    ByVal Command As String  
    ByVal BindType As Variant,  
    ByVal UserData As Variant) As IKeyBinding
```

Description

You can overwrite existing Stokes in the application. Refer to the Strokes topic in the application help for more information on Strokes.

Arguments

- String
A string that Stroke pattern.
- Command
A string that contains the menu command, script function, keyin command or key to assign to the Stroke.
- BindType
A [BindType](#) constant that defines the command type—key press, menu command, script function, or keyin command.
- UserData
(Optional) Reserved for future use.

Return Values

IKeyBinding. The new [KeyBinding Object](#).

Examples

```
' Get the document BindingTables object
Set docBindingTables = Gui.Bindings("Document")

' Assign Stroke to the "2" key
docBindingTables.AddStrokeBinding "14789", "2", BindKey

' Assign Stroke to menu item "FitBoard"
docBindingTables.AddStrokeBinding "159", "View->FitBoard", BindMenu

' Assign Stroke to a script function
Set bindObj = docBindingTables.AddStrokeBinding( "258", "" ,BindFunction)
bindObj.Target = ScriptEngine
bindObj.ExecuteMethod = "myLaunchDirMethod"

' Assign Stroke to a keyin command
docBindingTables.AddStrokeBinding "951", "run c:\temp\NewUserLayer.vbs",
    BindCommand
```

Bindings Property (BindingTables Collection)

Object: [BindingTables Collection](#)

Access: Read-Only

Returns the collection of keybinding objects of the BindingTables.

Usage

BindingTables.Bindings

Arguments

None.

Return Values

KeyBindings. The collection of KeyBinding objects ([KeyBindings](#)) of the BindingTables.

Examples

```
' Get the document BindingTables object
Set docBindingTables = Gui.Bindings("Document")

' Assign shortcut keys Alt+F1 to Help->Contents->Automation
docBindingTables.AddKeyBinding "Alt+F1", "Help->Contents->Automation",
BindMenu, BindAccelerator
```

Item Property (BindingTables Collection)

Object: [BindingTables Collection](#)

Access: Read-Only

Returns a keybinding object contained in the keybindings collection.

Usage

BindingTables.Item (ByVal Index As Variant)

Arguments

- Index
Index of the object to retrieve from the collection.

Return Values

KeyBinding. The object ([KeyBinding](#)) located at position Index.

KeyBindings Collection

Lists the KeyBindings methods and properties.

Table 5-3. KeyBindings Collection Methods and Properties

Method or Property	Description
Remove Method (KeyBindings Collection)	Removes a keybinding object from the keybindings collection.
Count Property (KeyBindings Collection)	Returns the number of keybinding objects contained in the keybindings collection.
Item Property (KeyBindings Collection)	Returns a keybinding object contained in the keybindings collection.

Remove Method (KeyBindings Collection)

Object: [KeyBindings Collection](#)

Removes a keybinding object from the keybindings collection.

Usage

KeyBindings.Remove (ByVal Index As Long)

Arguments

- Index
The index of the object ([KeyBinding](#)) to remove from the collection.

Count Property (KeyBindings Collection)

Object: [KeyBindings Collection](#)

Access: Read-Only

Returns the number of keybinding objects contained in the keybindings collection.

Usage

KeyBindings.Count

Arguments

None.

Return Values

The long number of keybinding objects the collection contains.

Item Property (KeyBindings Collection)

Object: [KeyBindings Collection](#)

Access: Read-Only

Returns a keybinding object contained in the keybindings collection.

Usage

KeyBindings.Item (ByVal Index As Variant)

Arguments

- Index
Index of the keybinding object to retrieve from the collection.

Return Values

KeyBinding. The object ([KeyBinding](#)) located at position Index.

Key Binding Enumerated Types

Lists the KeyBindSvr constants.

Table 5-4. KeyBinding Enumerated Types

Enumerated Type	Description
BindKeyType Enum	KeyBindSvr Binding Type constants.
BindType Enum	KeyBindSvr Binding Key Type constants.

BindKeyType Enum

KeyBindSvr Binding Type constants.

Usage

BindKeyType.Constant

Arguments

- Constant

You can specify two types of constants:

BindAccelerator

The value for this constant is 1. Assign the shortcut keys in the accelerator table.

BindKey

The value for this constant is 2. Reserved for future use.

BindType Enum

KeyBindSvr Binding Key Type constants.

Usage

BindType.Constant

Arguments

- Constant

You can specify two types of constants:

BindCommand

The value for this constant is 1. Assign the shortcut key or stroke to a key-in command.

BindFunction

The value for this constant is 2. Assign the shortcut key or stroke to a end-user function.

BindKeyPress

The value for this constant is 3. Assign the shortcut key or stroke to a key.

BindMenu

The value for this constant is 0. Assign the shortcut key or stroke to a menu command.

Chapter 6

Addin Controls

The Addin Controls let you include and run add-ins in the applications.

Addin Object	112
Active Property (Addin Object)	113
Control Property (Addin Object)	114
ClassID Property (Addin Object)	115
Description Property (Addin Object)	116
DisplayName Property (Addin Object)	117
Group Property (Addin Object)	118
Name Property (Addin Object)	119
Placement Property (Addin Object)	120
ProgID Property (Addin Object)	121
ShortCutKey Property (Addin Object)	122
Visible Property (Addin Object)	123
Width Property (Addin Object)	124
MGCAddins Collection	125
Add Method (MGCAddins Collection)	126
Remove Method (MGCAddins Collection)	127
Count Property (MGCAddins Collection)	128
Item Property (MGCAddins Collection)	129
Addins Enumerated Types	130
PlacementLocation Enum	131

Addin Object

Lists the Addin object properties.

Table 6-1. Addin Object Properties

Property	Description
Active Property (Addin Object)	Enable or disable the add-in control.
Control Property (Addin Object)	Returns the control interface of the add-in.
ClassID Property (Addin Object)	Returns the class identifier (CLSID) of the add-in control.
Description Property (Addin Object)	Returns the description about the add-in control.
DisplayName Property (Addin Object)	Sets or returns the display name of the add-in control.
Group Property (Addin Object)	Sets or returns the group name that the add-in control belongs to.
Name Property (Addin Object)	Sets or returns the add-in control instance name.
Placement Property (Addin Object)	Sets or returns the placement location of the add-in control.
ProgID Property (Addin Object)	Returns the programmatic identifier (ProgID) of the add-in control.
ShortCutKey Property (Addin Object)	Returns the shortcut keys assigned to the add-in control.
Visible Property (Addin Object)	Sets or returns the add-in control visible property.
Width Property (Addin Object)	Sets or returns the width of the add-in control.

Active Property (Addin Object)

Object: [Addin Object](#)

Access: Read/Write

Enable or disable the add-in control.

Usage

Addin.Active = True | False

Arguments

None.

Return Values

Boolean. If True, the Addin object is enabled. False, the Addin object is disabled.

Control Property (Addin Object)

Object: [Addin Object](#)

Access: Read-Only

Returns the control interface of the add-in.

Usage

Addin.Control

Arguments

None.

ClassID Property (Addin Object)

Object: [Addin Object](#)

Access: Read-Only

Returns the class identifier (CLSID) of the add-in control.

Usage

Addin.ClassID

Arguments

None.

Return Values

A string that represents the class identifier (CLSID) of the Addin object.

Description Property (Addin Object)

Object: [Addin Object](#)

Access: Read-Only

Returns the description about the add-in control.

Usage

Addin.Description

Arguments

None.

Return Values

A string that contains the description about the add-in control.

DisplayName Property (Addin Object)

Object: [Addin Object](#)

Access: Read/Write

Sets or returns the display name of the add-in control.

Usage

Addin.DisplayName = String

Arguments

None.

Return Values

A string that contains the display name of the add-in control.

Group Property (Addin Object)

Object: [Addin Object](#)

Access: Read/Write

Sets or returns the group name that the add-in control belongs to.

Usage

Addin.Group = String

Arguments

None.

Return Values

A string that contains the group name that the add-in control belongs to.

Name Property (Addin Object)

Object: [Addin Object](#)

Access: Read/Write

Sets or returns the add-in control instance name.

Usage

Addin.Name = String

Arguments

None.

Return Values

A string that contains the add-in control instance name.

Placement Property (Addin Object)

Object: [Addin Object](#)

Access: Read/Write

Sets or returns the placement location of the add-in control.

Usage

Addin.Placement = PlacementLocation

Arguments

None.

Return Values

PlacementLocation. A long that represents the placement location ([PlacementLocation](#)) of the add-in control.

ProgID Property (Addin Object)

Object: [Addin Object](#)

Access: Read-Only

Returns the programmatic identifier (ProgID) of the add-in control.

Usage

Addin.ProgID

Arguments

None.

Return Values

A string that represents the programmatic identifier (ProgID) of the add-in control.

ShortCutKey Property (Addin Object)

Object: [Addin Object](#)

Access: Read-Only

Returns the shortcut keys assigned to the add-in control.

Usage

Addin.ShortCutKey

Arguments

None.

Return Values

A string that represents the shortcut keys assigned to the add-in control.

Visible Property (Addin Object)

Object: [Addin Object](#)

Access: Read/Write

Sets or returns the add-in control visible property.

Usage

Addin.Visible = True | False

Arguments

None.

Return Values

Boolean. If True, the Addin object is visible. If False, the Addin object is not visible.

Description

The default value of this property is determined within the addin. So, you should specifically set this property value when the addin is loaded.

Examples

```
Sub ToggleVisibility (nID)
    'Called by Hide/Unhide Output Window menu item
    If Not outputAddin Is Nothing Then
        ' Toggle Output Window visibility
        If outputAddin.Visible Then
            outputAddin.Visible = False
        Else
            outputAddin.Visible = True
        End If
    End If
End Sub
```

Width Property (Addin Object)

Object: [Addin Object](#)

Access: Read/Write

Sets or returns the width of the add-in control.

Usage

Addin.Width

Arguments

None.

Return Values

Integer

MGCAddins Collection

Lists the MGCAddins methods and properties.

Table 6-2. MGCAddins Collection Methods and Properties

Method or Property	Description
Add Method (MGCAddins Collection)	Adds an Addin object to the MGCAddins collection.
Remove Method (MGCAddins Collection)	Removes an Addin object from the MGCAddins collection.
Count Property (MGCAddins Collection)	Returns the number of Addin objects contained in the MGCAddins collection.
Item Property (MGCAddins Collection)	Returns an Addin object contained in the MGCAddins collection.

Add Method (MGCAAddins Collection)

Object: [MGCAAddins Collection](#)

Adds an Addin object to the MGCAAddins collection.

Usage

```
MGCAAddins.Add (ByVal ProgIDorClassID As String,  
    ByVal InstanceID As String,  
    ByVal AssociatedScript As String,  
    [ByVal Location As Variant],  
    [ByVal ShortCutKey As Variant],  
    [ByVal GroupName As Variant],  
    [ByVal Quiet As Variant],  
    [ByVal TbarMenu As Variant]) As Addin
```

Arguments

- ProgIDorClassID
A string that represents the [ProgID](#) or the [ClassID](#) of the add-in program.
- InstanceID
A string that represents the name of the add-in program instance.
- AssociatedScript
A string that represents the script will be run when the Addin object is loaded.
- Location
(Optional) The placement location ([PlacementLocation](#)) of the Addin object when loaded.
- ShortCutKey
(Optional) Reserved for future use.
- GroupName
(Optional) A string that represents the group name of the Addin object.
- Quiet
(Optional) Enable or disable errors reporting when loading Addin object.
- TbarMenu
(Optional) Reserved for future use.

Return Values

Addin. The object ([Addin](#)) to add to the collection.

Remove Method (MGCAAddins Collection)

Object: [MGCAAddins Collection](#)

Removes an Addin object from the MGCAAddins collection.

Usage

MGCAAddins.Remove (ByVal Index As Long)

Arguments

- Index
The index of the object ([Addin](#)) to remove from the collection.

Count Property (MGCAAddins Collection)

Object: [MGCAAddins Collection](#)

Access: Read-Only

Returns the number of Addin objects contained in the MGCAAddins collection.

Usage

MGCAAddins.Count

Arguments

None.

Return Values

Count. The number of Addin objects the collection contains.

Item Property (MGCAAddins Collection)

Object: [MGCAAddins Collection](#)

Access: Read-Only

Returns an Addin object contained in the MGCAAddins collection.

Usage

MGCAAddins.Item (ByVal Index As Variant)

Arguments

- Index
Index of the Addin object to retrieve from the collection.

Return Values

Addin. The object ([Addin](#)) located at position Index.

Addins Enumerated Types

Lists the Addins types.

Table 6-3. Addins Enumerated Types

Enumerated Type	Description
PlacementLocation Enum	Addin Object Placement Location Type constants.

PlacementLocation Enum

Addin Object Placement Location Type constants.

Usage

PlacementLocation.Constant

Arguments

- Constant

You can specify three types of constants:

PlacementLocationBottom

The value for this constant is 3. Align the add-in control under the design window.

PlacementLocationLeft

The value for this constant is 0. Align the add-in control to the left of the design window.

PlacementLocationRight

The value for this constant is 2. Align the add-in control to the right of the design window.

PlacementLocationTop

The value for this constant is 1. Align the add-in control above the design window.

Chapter 7

JScriptHelper

The JScriptHelper object provides additional methods for using JScript to access automation in Mentor Graphics applications.

JScriptHelper Object	134
ToScriptArray Method (JScriptHelper Object)	135
Nothing Property (JScriptHelper Object)	136
JScriptHelper Example	136

JScriptHelper Object

Lists the JScriptHelper methods and properties.

Table 7-1. JScriptHelper Object Methods and Properties

Method or Property	Description
ToScriptArray Method (JScriptHelper Object)	Converts a JScriptArray to a VBArray.
Nothing Property (JScriptHelper Object)	Returns a null object in JScripts.

ToScriptArray Method (JScriptHelper Object)

Object: [JScriptHelper](#)

Prerequisites: None.

Converts a JScriptArray to a VBAArray.

Usage

ScriptHelper.ToScriptArray(JScriptArray) As Object

Arguments

None.

Return Values

Object. Returns a VBAArray that has been converted from a JScriptArray.

Description

In order to pass a points array (often used to create new design objects defined by a geometry), you must convert any JScriptArrays to VBAArrays.

Nothing Property (JScriptHelper Object)

Object: [JScriptHelper](#)

Access: Read-Only

Prerequisites: None.

Returns a null object in JScripts.

Usage

ScriptHelper.Nothing As Object

Arguments

None.

Return Values

Object. A null object.

Description

Currently, there is no null object available within JScript, even though many Mentor Graphics methods require it.

JScriptHelper Example

The following is an example that shows the use of the JScriptHelper method and property.


```
// This example shows how you can use JScriptHelper to convert an array to
// a a VBAArray which you can then pass to other Mentor Grapics apis.

// add any type librarys to be used.
Scripting.AddTypeLibrary("MGPCB.ExpeditionPCBApplication");
Scripting.AddTypeLibrary("Scripting.FileSystemObject");

// Get the application object
var pcbApp = Application;

// Get the active document
var pcbDoc = pcbApp.ActiveDocument;

// Server validation function
// License the document
ValidateServer(pcbDoc);
var oHelper = Scripting.CreateObject("JScriptHelper.ScriptHelper");

var obj = oHelper.Nothing;
if(obj == oHelper.Nothing)
    obj = pcbDoc.ConductorLayerGfxs2(0, epcbSelectSelected).Item(1);
if(obj==oHelper.Nothing)
    obj = pcbDoc.FabricationLayerGfxs(epcbFabAll,
epcbSelectSelected).Item(1);
if(obj==oHelper.Nothing)
    obj = pcbDoc.UserLayerGfxs(epcbSelectSelected).Item(1);

// Get an array from the interface
var arr = obj.Geometry.PointsArray;

// Convert it to a JScript array.
var jArr = ToJPntsArray(arr);

// Once you have a jArr you can manipulate it in JScript.

// Display the JScript array
pcbApp.Gui.Display(GetPointsArrayString(jArr));

// Once you have a JScript array, you can use JScript to modify the array.
// This function uses JScript functionality to modify all elements of a
// JScript array.
// Assume jArr is an initialized JScript array.
for (i = 0; i < jArr[0].length; i++)
{
    jArr[0][i] = parseFloat(jArr[0][i]) + 100;
}
// Before passing the array back to the interface convert
// it back to a SAFEARRAY. This can be done inline.

// Use the shifted points to create user layer gfxs.
var userLayerGfx = pcbDoc.PutUserLayerGfx(pcbDoc.UserLayers.Item(1), 0,
jArr[0].length, ToSafePntsArray(jArr), false, oHelper.Nothing,
epcbUnitCurrent)
userLayerGfx.Selected = true

// Notice that JScriptHelper is used with the PutUserLayerGfx function
// call.
```

```
// Also notice that the oHelper (JScriptHelper) specifies Nothing (the
// property) for the interface. "Nothing" is not a keyword in JScript, but
// is needed in many Mentor Graphics automation functions.

//.....
//Local functions

function ToSafePntsArray(jArr)
{
    return oHelper.ToSafeArray(jArr);
}

// This function converts a VBAArray to a two-dimensional JScript array.
// Note that this function does not require JScriptHelper, but uses
// inherent JScript functionality.
function ToJPntsArray(VBArr)
{
    // Create a VBAArray object
    var vbArr = new VBAArray(VBArr)

    // Convert it a a single dimension JArray
    var jsArrSingelDim = vbArr.toArray();

    // Make the single dim array into a multidimensional array.
    // assumes there are 2 dimensions with 3 rows in the first dimension
    var numPnts = jsArrSingelDim.length / 3;
    var i;
    var arr = new Array(3);
    arr[0] = new Array(numPnts);
    arr[1] = new Array(numPnts);
    arr[2] = new Array(numPnts);
    for (i = 0; i < numPnts; i++)
    {
        arr[0][i] = parseFloat(jsArrSingelDim[i*3]);
        arr[1][i] = parseFloat(jsArrSingelDim[i*3+1]);
        arr[2][i] = parseFloat(jsArrSingelDim[i*3+2]);
    }

    return arr;
}

// This function reads a JScript array into a stream format so that it can
// be displayed or output.
function GetPointsArrayString(pnts)
{
    var i,j;
    var str="";
    for(i = 0; i<pnts[0].length; i++) {
        str = str + "(" + pnts[0][i];
        str = str + "," + pnts[1][i];
        str = str + "," + pnts[2][i] + ")";
        str = str + ' '
    }
    return str;
}

// Server validation function
```

```
function ValidateServer(docObj) {  
  
    var keyInt;  
    var licenseTokenInt;  
    var licenseServerObj;  
    var retValInt = 1;  
  
    // Ask Xpedition Layout's document for the key;  
    keyInt = docObj.Validate(0);  
  
    // Get license server;  
    licenseServerObj = new  
ActiveXObject("MGCPBAutomationLicensing.Application");  
  
    // Ask the license server for the license token;  
    licenseTokenInt = licenseServerObj.GetToken(keyInt);  
  
    // Release license server;  
    licenseServerObj = null;  
  
    // Check for error in Validate  
    try {  
        // Ask the document to validate the license token;  
        docObj.Validate(licenseTokenInt);  
    } catch(err) {  
        retValInt = 0  
    }  
  
    return retValInt;  
}
```


Third-Party Information

For third-party information, refer to [*Third-Party Software*](#).

End-User License Agreement with EDA Software Supplemental Terms

Use of software (including any updates) and/or hardware is subject to the End-User License Agreement together with the Mentor Graphics EDA Software Supplement Terms. You can view and print a copy of this agreement at:

mentor.com/eula

