



# Maxwell Scripting Guide



ANSYS, Inc.  
Southpointe  
2600 Ansys Drive  
Canonsburg, PA 15317  
[ansysinfo@ansys.com](mailto:ansysinfo@ansys.com)  
<https://www.ansys.com>  
(T) 724-746-3304  
(F) 724-514-9494

Release 2023 R2  
July 2023

ANSYS, Inc. and  
ANSYS Europe,  
Ltd. are UL  
registered ISO  
9001:2015 com-  
panies.

## Copyright and Trademark Information

© 1986-2023 ANSYS, Inc. Unauthorized use, distribution or duplication is prohibited.

ANSYS, Ansys Workbench, AUTODYN, CFX, FLUENT and any and all ANSYS, Inc. brand, product, service and feature names, logos and slogans are registered trademarks or trademarks of ANSYS, Inc. or its subsidiaries located in the United States or other countries. ICEM CFD is a trademark used by ANSYS, Inc. under license. All other brand, product, service and feature names or trademarks are the property of their respective owners. FLEXIm and FLEXnet are trademarks of Flexera Software LLC.

## Disclaimer Notice

THIS ANSYS SOFTWARE PRODUCT AND PROGRAM DOCUMENTATION INCLUDE TRADE SECRETS AND ARE CONFIDENTIAL AND PROPRIETARY PRODUCTS OF ANSYS, INC., ITS SUBSIDIARIES, OR LICENSORS. The software products and documentation are furnished by ANSYS, Inc., its subsidiaries, or affiliates under a software license agreement that contains provisions concerning non-disclosure, copying, length and nature of use, compliance with export laws, warranties, disclaimers, limitations of liability, and remedies, and other provisions. The software products and documentation may be used, disclosed, transferred, or copied only in accordance with the terms and conditions of that software license agreement.

ANSYS, Inc. and ANSYS Europe, Ltd. are UL registered ISO 9001: 2015 companies.

## U.S. Government Rights

For U.S. Government users, except as specifically granted by the ANSYS, Inc. software license agreement, the use, duplication, or disclosure by the United States Government is subject to restrictions stated in the ANSYS, Inc. software license agreement and FAR 12.212 (for non-DOD licenses).

## Third-PartySoftware

See the legal information in the product help files for the complete Legal Notice for Ansys proprietary software and third-party software. If you are unable to access the Legal Notice, please contact ANSYS, Inc.

# Table of Contents

<b>Table of Contents .....</b>	<b>Contents-1</b>
<b>1 - Introduction to Scripting .....</b>	<b>1-1</b>
Scripting Help Conventions .....	1-1
Variable Types .....	1-2
Introduction to VBScript .....	1-2
Simple and Composite Names .....	1-3
VBScript Variables .....	1-3
Declaring Variables .....	1-3
Declaring Variables in Python .....	1-4
Variable Naming Conventions .....	1-4
Scope and Lifetime of Variables .....	1-4
Array Variables .....	1-4
VBScript Operators .....	1-5
Operator Precedence .....	1-6
Arithmetic Operators .....	1-6
Comparison Operators .....	1-7
Logical Operators .....	1-7
Controlling Program Execution .....	1-8
Using If...Then...Else .....	1-8
Using Select Case .....	1-8
Looping Through Code .....	1-9
Using a For...Next Loop .....	1-9
Using a Do Loop .....	1-9
Repeating Statements While a Condition is True .....	1-9
Repeating a Statement Until a Condition Becomes True .....	1-9
VBScript Procedures .....	1-10
Function Procedures .....	1-10
Sub Procedures .....	1-10

Converting Between Data Types .....	1-10
Including Scripts .....	1-10
Aborting Scripts .....	1-11
Interacting with a Script .....	1-11
Recommended VBScript References .....	1-12
Introduction to IronPython .....	1-12
Scope .....	1-12
Python compatibility .....	1-12
Advantages of IronPython .....	1-12
Introduction to IronPython .....	1-14
Scope .....	1-14
Python compatibility .....	1-14
Advantages of IronPython .....	1-14
IronPython Mini Cookbook .....	1-15
Comments .....	1-15
Assigning/Creating Variables .....	1-16
Create Lists/Arrays .....	1-16
Create Dictionaries/Maps .....	1-17
Boolean Values .....	1-17
Converting Numbers to Strings and Vice Versa .....	1-17
String Formatting/Concatenation .....	1-18
Looping over Lists .....	1-18
Looping over a Range .....	1-18
Indentation in IronPython .....	1-19
Indenting Functions .....	1-19
Indenting If Conditions .....	1-19
Methods in IronPython .....	1-20
Finding Methods .....	1-20
Help .....	1-20
Translating Script Commands from VBScript to IronPython .....	1-20

---

Script Method Argument .....	1-20
Primitive Types .....	1-21
Named Arrays .....	1-21
Named Functions .....	1-21
VBScript Method Call Types .....	1-21
Converting VBScript Function calls to IronPython Syntax .....	1-22
Return Values .....	1-23
Primitive Method Arguments .....	1-23
Named Array Arguments .....	1-23
Named Array Values with All Key Value Pairs .....	1-23
Named Arrays with Nested Named Arrays .....	1-24
Function Blocks .....	1-25
Scripting Using Iron Python .....	1-25
Translating a script in VBScript to IronPython .....	1-25
Writing an IronPython script from scratch .....	1-25
IronPython Script Execution Environment .....	1-26
Script Argument for IronPython .....	1-27
Scripting using Embedded VBScript or JavaScript .....	1-27
Scripting with IronPython .....	1-30
Standalone IronPython .....	1-31
Running Standalone IronPython .....	1-31
Using a Recorded Script .....	1-32
Creating an External Script .....	1-32
Example Script .....	1-33
IronPython Samples .....	1-34
Change property .....	1-34
Create a Cone using IronPython .....	1-35
Creating User Defined Primitives and User Defined Models in Python Scripts .....	1-39
Advantages Compared to C++ .....	1-39
Changes compared to C .....	1-39

---

Structures .....	1-39
Return Values for UDM and UDP Functions .....	1-40
Constants .....	1-40
Methods .....	1-40
Output Parameters .....	1-40
Comparison with C function: .....	1-41
'List Size' Parameters .....	1-41
Comparison with C function: .....	1-42
Added Parameters .....	1-43
Developing a UDM/UDP .....	1-44
Creation .....	1-44
Location .....	1-44
Organize .....	1-44
Edit/Reload .....	1-45
UDPExtension .....	1-45
Import .....	1-45
Main class: UDPExtension .....	1-45
IUDPExtension methods .....	1-45
Mandatory methods .....	1-45
GetLengthParameterUnits() .....	1-45
GetPrimitiveTypeInfo() .....	1-45
GetPrimitiveParametersDefinition2() .....	1-45
AreParameterValuesValid2(errorMsg, udpParams) .....	1-46
CreatePrimitive2(funcLib, udpParams) .....	1-46
Optional methods .....	1-46
GetPrimitiveParameters() .....	1-46
GetRegisteredFaceNames() .....	1-46
GetRegisteredEdgeNames() .....	1-46
GetRegisteredVertexNames() .....	1-46
MapParametersDefinitionVersions2(oldVersion, oldUDPPParams) .....	1-47

---

GetOldPrimitiveParametersDefinition2(version ) .....	1-47
Example UDP .....	1-47
UDMExtension .....	1-47
Import .....	1-47
Main class: UDMExtension .....	1-48
IUDMExtension methods .....	1-48
Mandatory methods. ....	1-48
GetInfo() .....	1-48
IsAttachedToExternalEditor() .....	1-48
CreateInstance(funcLib) .....	1-48
GetUnits(instanceld) .....	1-48
ReleaseInstance(instanceld) .....	1-49
GetAttribNameForEntityId() .....	1-49
GetAttribNameForPartId() .....	1-49
Optional methods .....	1-49
GetInstanceSourceInfo(instanceld) .....	1-50
ShouldAttachDefinitionFilesToProject() .....	1-50
Example UDM .....	1-50
UDMFunctionLibrary .....	1-51
Functions list: .....	1-52
UDM/UDP Functions .....	1-53
Return Values for Each UDM and UDP Function .....	1-53
UDP/UDM Structures and Constants .....	1-56
UDP/UDM Structures .....	1-56
List of structures .....	1-57
UDP/UDM Constants .....	1-63
Enum constants: .....	1-63
UDP Python Example .....	1-65
Introduction to CPython (Beta) .....	1-70
Ansys Electronics Desktop Scripting .....	1-73

---

Overview of Electronics Desktop Scripting Objects .....	1-73
oAnsoftApp .....	1-74
oDesktop .....	1-74
oProject .....	1-74
oDesign .....	1-74
oEditor .....	1-75
oModule .....	1-75
Example Script Opening .....	1-76
GetActiveProject and GetActiveDesign for Wider Use .....	1-77
Running a Script .....	1-77
Within Electronics Desktop .....	1-77
From the Command Line .....	1-78
Direct Launch .....	1-78
Recording a Script .....	1-79
Recording a Script to File .....	1-79
Recording a Script to a Project .....	1-80
Working with Project Scripts .....	1-81
Executing a Script from Within a Script .....	1-82
Electronics Desktop Scripting Conventions .....	1-82
Named Arguments .....	1-83
VBscript Example .....	1-83
IronPython Example .....	1-84
Setting Numerical Values .....	1-85
Event Callback Scripting .....	1-86
<b>2 - Application Object Script Commands .....</b>	<b>2-1</b>
GetAppDesktop .....	2-2
GetApp .....	2-2
<b>3 - Desktop Object Script Commands .....</b>	<b>3-1</b>
AddMessage .....	3-5
ClearMessages .....	3-6

---

CloseAllWindows .....	3-7
CloseProject .....	3-8
CloseProjectNoForce .....	3-8
Count .....	3-9
DeleteProject .....	3-11
EnableAutoSave .....	3-11
ExportOptionsFiles .....	3-12
GetActiveProject .....	3-13
GetAutoSaveEnabled .....	3-14
GetBuildDateTimeString .....	3-14
GetCustomMenuSet .....	3-15
GetDefaultUnit .....	3-16
GetDesigns .....	3-18
GetDesktopConfiguration .....	3-19
GetDistributedAnalysisMachines .....	3-20
GetDistributedAnalysisMachinesForDesignType .....	3-20
GetExeDir .....	3-21
GetGDIObjectCount .....	3-22
GetLibraryDirectory .....	3-22
GetLocalizationHelper .....	3-24
GetMessages .....	3-25
GetName [Desktop] .....	3-26
GetPersonalLibDirectory .....	3-27
GetProcessID .....	3-28
GetProjects .....	3-29
GetProjectDirectory .....	3-29
GetProjectList .....	3-30
GetSchematicEnvironment .....	3-31
GetScriptingToolsHelper .....	3-32
GetSysLibDirectory .....	3-33

---

GetTempDirectory .....	3-33
GetUserLibDirectory .....	3-34
GetVersion .....	3-35
ImportANF .....	3-35
ImportAutoCAD .....	3-37
ImportGDSII .....	3-38
ImportODB .....	3-39
LaunchJobMonitor .....	3-40
NewProject .....	3-40
OpenAndConvertProject .....	3-41
OpenMultipleProjects .....	3-42
OpenProject .....	3-43
OpenProjectWithConversion .....	3-44
Paste (Project Object) .....	3-44
Paste (Project Object) .....	3-45
PauseRecording .....	3-45
PauseScript .....	3-46
Print .....	3-47
QuitApplication .....	3-47
RefreshJobMonitor .....	3-48
ResetLogging .....	3-49
RestoreProjectArchive .....	3-50
RestoreWindow .....	3-51
ResumeRecording .....	3-51
RunACTWizardScript .....	3-52
RunProgram .....	3-53
RunScript .....	3-54
RunScriptWithArguments .....	3-55
SelectScheduler .....	3-56
SetActiveProject .....	3-58

---

SetActiveProjectByPath .....	3-59
SetCustomMenuSet .....	3-59
SetDesktopConfiguration .....	3-61
SetLibraryDirectory .....	3-62
SetProjectDirectory .....	3-62
SetSchematicEnvironment .....	3-63
SetTempDirectory .....	3-64
ShowDockingWindow .....	3-65
Sleep .....	3-65
SubmitJob .....	3-66
TileWindows .....	3-67
Desktop Commands For Registry Values .....	3-68
DoesRegistryValueExist .....	3-69
GetRegistryInt .....	3-70
GetRegistryString .....	3-71
SetRegistryFromFile .....	3-72
SetRegistryInt .....	3-72
SetRegistryString .....	3-73
<b>4 - Running Instances Manager Script Commands .....</b>	<b>4-1</b>
GetAllRunningInstances .....	4-1
GetRunningInstanceByProcessID .....	4-2
GetRunningInstancesMgr .....	4-2
<b>5 - Project Object Script Commands .....</b>	<b>5-1</b>
AnalyzeAll [project] .....	5-3
ClearMessages .....	5-4
Close .....	5-4
CopyDesign .....	5-5
CutDesign .....	5-6
DeleteDesign .....	5-7
DeleteToolObject .....	5-7

---

GetActiveDesign .....	5-8
GetConfigurableData (Project) .....	5-9
GetDefinitionManager .....	5-9
GetDependentFiles .....	5-10
GetDesign .....	5-11
GetEDBHandle .....	5-11
GetLegacyName .....	5-12
GetName [Project] .....	5-13
GetPath .....	5-13
GetTopDesignList .....	5-14
InsertDesign .....	5-15
InsertDesignWithWorkflow .....	5-16
InsertToolObject .....	5-17
Paste (Project Object) .....	5-17
Redo [Project Level] .....	5-18
Rename .....	5-19
RestoreProjectArchive .....	5-20
Save .....	5-21
SaveAs .....	5-21
SaveAsStandAloneProject .....	5-24
SaveProjectArchive .....	5-24
SetActiveDefinitionEditor .....	5-25
SetActiveDesign .....	5-26
SimulateAll .....	5-27
Undo [Project] .....	5-28
UpdateDefinitions .....	5-28
<b>6 - Object Oriented Property Scripting</b> .....	<b>6-1</b>
Object-Oriented Scripting .....	6-1
Material Properties and Examples .....	6-9
Body Properties and Modification .....	6-11

---

Retrieving Variables .....	6-11
Retrieve Datasets and Values .....	6-12
GetSolutionData API .....	6-14
Summary .....	6-14
Additional Details Specific to AEDT Solvers .....	6-18
Additional details on Boundaries/Excitations .....	6-20
3D component encapsulation .....	6-21
Additional details on Solve setup .....	6-22
Materials Scripting Support .....	6-23
Object Oriented Scripting for Materials .....	6-27
Examples showing change to material property type: .....	6-28
Examples showing change to a vector component value .....	6-28
Change choice property value .....	6-29
Change choice property value .....	6-30
<b>7 - Property Script Commands .....</b>	<b>7-1</b>
Object Script Property Function Summary .....	7-3
Object Path .....	7-3
Property Object .....	7-3
Project Object .....	7-5
Design Object .....	7-6
3D Modeler Object .....	7-7
Variable Object .....	7-7
Optimetrics Module Object: .....	7-9
Optimetrics Setup Object .....	7-9
ReportSetup(Results) Module Object: .....	7-10
ReportSetup(Results) Module Child Objects: .....	7-11
Radiation Module Object: .....	7-12
Radiation Module Child Objects: .....	7-12
Conventions Used in this Chapter .....	7-13
GetArrayVariables .....	7-26

---

GetProperties .....	7-26
GetPropertyValue .....	7-28
GetVariables .....	7-30
GetVariableValue .....	7-30
SetPropertyValue .....	7-31
SetVariableValue .....	7-33
<b>8 - Dataset Script Commands .....</b>	<b>8-1</b>
AddDataset .....	8-1
DeleteDataset .....	8-4
EditDataset .....	8-5
ImportDataset .....	8-7
Note About File Types: .....	8-9
<b>9 - Design Object Script Commands .....</b>	<b>9-1</b>
ApplyMeshOps .....	9-5
Analyze .....	9-6
AnalyzeAll (Maxwell menu) .....	9-6
AnalyzeDistributed .....	9-7
ClearLinkedData (Design) .....	9-8
ConfigureFluentConductivityCoupling .....	9-8
ConstructVariationString .....	9-9
CreateParametricCircuit .....	9-10
Create3DDesign [Maxwell] .....	9-11
DeleteFieldVariation .....	9-12
DeleteFullVariation .....	9-13
DeleteLinkedDataVariation .....	9-13
DeleteVariation .....	9-14
EditNotes .....	9-14
EnableHarmonicForceCalculation .....	9-15
ExportConvergence .....	9-25
ExportElementBasedHarmonicForce .....	9-26

---

ExportHarmonicTransientForce .....	9-27
ExportMeshStats .....	9-28
ExportProfile .....	9-29
GetConfigurableData .....	9-30
GetData .....	9-31
GetDesignValidationInfo .....	9-31
GetGeometryMode .....	9-32
GetManagedFilesPath .....	9-33
GetModule .....	9-34
GetName .....	9-35
GetNominalVariation .....	9-35
GetNoteText .....	9-36
GetPostProcessingVariables .....	9-37
GetSelections [Design] .....	9-37
GetSolutionType .....	9-38
GetVariationVariableValue .....	9-38
InitializeSystemCoupling .....	9-39
Is2D .....	9-40
Is3D .....	9-40
Redo [Design] .....	9-40
RenameDesignInstance .....	9-41
ResetToTimeZero .....	9-42
SetActiveEditor .....	9-42
SetConductivityThreshold .....	9-43
SetDesignSettings [Maxwell] .....	9-44
SetObjectDeformation .....	9-57
SetObjectTemperature .....	9-58
SetSolutionType (Maxwell) .....	9-60
Solve .....	9-60
StopSimLink .....	9-61

---

RunToolkit .....	9-62
Undo [Design] .....	9-63
ValidateDesign .....	9-64
ValidateLink .....	9-64
<b>10 - 3D Modeler Editor Script Commands .....</b>	<b>10-1</b>
Conventions Used in this Chapter: .....	10-1
<AttributesArray> .....	10-1
<SelectionsArray> .....	10-2
Draw Menu Commands .....	10-4
Create3DComponent .....	10-6
CreateBondwire .....	10-10
CreateBox .....	10-14
CreateCircle .....	10-16
CreateCone .....	10-19
CreateCutplane .....	10-22
CreateCylinder .....	10-24
CreateEllipse .....	10-26
CreateEquationCurve .....	10-29
CreateEquationSurface .....	10-33
CreateHelix .....	10-36
CreatePoint .....	10-38
CreateUserDefinedPart .....	10-40
CreatePolyline .....	10-47
CreateRectangle .....	10-53
CreateRegion .....	10-55
CreateRegularPolyhedron .....	10-59
CreateRegularPolygon .....	10-62
CreateSphere .....	10-65
CreateSpiral .....	10-67
CreateTorus .....	10-69

---

CreateUserDefinedModel .....	10-72
CreateUserDefinedPart .....	10-79
Edit3DComponent .....	10-87
[Beta] EditNativeComponentDefinition [Maxwell] .....	10-89
EditPolyline .....	10-92
Get3DComponentParameters .....	10-97
Get3DComponentDefinitionNames .....	10-97
Get3DComponentInstanceNames .....	10-98
Get3DComponentMaterialNames .....	10-99
Get3DComponentMaterialProperties .....	10-99
Insert3DComponent .....	10-100
[Beta] InsertNativeComponent [Layout Component to Maxwell with CS] .....	10-101
InsertPolylineSegment .....	10-105
SweepAlongPath .....	10-107
SweepAlongVector .....	10-109
SweepAroundAxis .....	10-112
SweepFacesAlongNormal .....	10-114
SweepFacesAlongNormalWithAttributes .....	10-116
UpdateComponentDefinition .....	10-119
Edit Menu Commands .....	10-120
Copy .....	10-120
DeletePolylinePoint .....	10-121
DuplicateAlongLine .....	10-122
DuplicateAroundAxis .....	10-125
DuplicateMirror .....	10-128
Mirror .....	10-131
Move .....	10-133
OffsetFaces .....	10-134
Paste (Model Editor) .....	10-136
Rotate .....	10-137

---

Scale .....	10-138
Modeler Menu Commands .....	10-140
AssignMaterial .....	10-142
Chamfer .....	10-145
Connect .....	10-147
CoverLines .....	10-148
CoverSurfaces .....	10-149
CreateEntityList .....	10-150
CreateFaceCS .....	10-152
CreateGroup .....	10-157
CreateObjectCS .....	10-158
CreateObjectFromEdges .....	10-165
CreateObjectFromFaces .....	10-166
CreateRelativeCS .....	10-168
DeleteEmptyGroups .....	10-170
DeleteLastOperation .....	10-171
DetachFaces .....	10-172
EditEntityList .....	10-173
EditFaceCS .....	10-175
EditObjectCS .....	10-181
EditRelativeCS .....	10-188
Export .....	10-190
ExportModellImageToFile .....	10-192
Fillet .....	10-195
FlattenGroup .....	10-197
GenerateHistory .....	10-197
HealObject .....	10-199
GetActiveCoordinateSystem .....	10-203
GetCoordinateSystems .....	10-204
Import .....	10-204

---

ImportDXF [Modeler] .....	10-208
ImportGDSII [Modeler] .....	10-212
Intersect .....	10-216
MoveCStoEnd .....	10-217
MoveEntityToGroup .....	10-218
MoveFaces .....	10-219
ProjectSheet .....	10-221
PurgeHistory .....	10-222
Section .....	10-223
SeparateBody .....	10-225
SetModelUnits .....	10-226
SetWCS .....	10-227
ShowWindow .....	10-228
Split .....	10-229
Subtract .....	10-232
SweepFacesAlongNormal .....	10-233
ThickenSheet .....	10-235
UncoverFaces .....	10-237
Unite .....	10-238
Ungroup .....	10-239
WrapSheet .....	10-240
Other oEditor Commands .....	10-241
AddDefinitionFromBlock .....	10-244
AddDefinitionFromLibFile .....	10-249
AddViewOrientation .....	10-253
BreakUDMConnection .....	10-261
ChangeProperty .....	10-262
Delete .....	10-268
GetBodyNamesByPosition .....	10-269
GetEdgeByPosition .....	10-271

---

GetEdgeIDsFromFace .....	10-272
GetEdgeIDsFromObject .....	10-273
GetFaceArea .....	10-273
GetFaceCenter .....	10-274
GetFaceByPosition .....	10-275
GetFaceIDs .....	10-277
GetGeometryModelerMode .....	10-277
GetModelBoundingBox .....	10-278
GetObjectIDByName .....	10-279
GetObjectName .....	10-279
GetObjectNameByFaceID .....	10-280
GetObjectsByMaterial .....	10-281
GetObjectsInGroup .....	10-281
GetMatchedObjectName .....	10-282
GetModelUnits .....	10-283
GetNumObjects .....	10-284
GetSelections [Model Editor] .....	10-284
GetUserPosition .....	10-285
GetVertexIDsFromEdge .....	10-286
GetVertexIDsFromFace .....	10-286
GetVertexIDsFromObject .....	10-287
GetVertexPosition .....	10-288
OpenExternalEditor .....	10-288
PageSetup .....	10-289
RenamePart .....	10-291
<b>11 - Output Variable Script Commands .....</b>	<b>11-1</b>
CreateOutputVariable (Maxwell) .....	11-1
DeleteOutputVariable .....	11-3
DoesOutputVariableExist .....	11-3
EditOutputVariable .....	11-4

---

GetOutputVariableValue .....	11-6
<b>12 - Reporter Editor Script Commands .....</b>	<b>12-1</b>
AddCartesianLimitLine .....	12-4
AddCartesianXMarker .....	12-6
AddCartesianYMarker .....	12-7
AddDeltaMarker .....	12-8
AddMarker .....	12-9
AddNote .....	12-10
AddTraceCharacteristics .....	12-12
AddTraces .....	12-13
ChangeProperty[ReportSetup] .....	12-16
ClearAllMarkers .....	12-20
ClearAllTraceCharacteristics .....	12-20
CopyTracesData .....	12-21
CopyReportsData .....	12-22
CopyReportDefinitions .....	12-23
CopyTraceDefinitions .....	12-23
CreateReportFromFile .....	12-24
CreateReport (Maxwell) .....	12-25
CreateReportFromTemplate .....	12-35
CreateReportOfAllQuantities .....	12-36
DeleteMarker .....	12-37
DeleteAllReports .....	12-38
DeleteReports .....	12-39
DeleteTraces .....	12-39
ExportImageToFile [Reporter] .....	12-40
ExportPlotImageToFile .....	12-41
ExportReport .....	12-43
ExportToFile .....	12-44
ExportToFile [Reporter] .....	12-46

---

ExportMarkerTable .....	12-47
FFTOnReport .....	12-47
GetAllReportNames .....	12-48
GetAllCategories .....	12-49
GetAllQuantities .....	12-50
GetAvailableDisplayTypes .....	12-51
GetAvailableReportTypes .....	12-52
GetAvailableSolutions .....	12-52
GetDataExpressions .....	12-53
GetDataUnits .....	12-54
GetDesignVariableNames .....	12-55
GetDesignVariableUnits .....	12-56
GetDesignVariableValue .....	12-57
GetDesignVariationKey .....	12-58
GetDisplayType (Maxwell) .....	12-58
GetImagDataValues .....	12-59
GetPerQuantityPrimarySweepValues .....	12-60
GetPropertyValue .....	12-61
GetRealDataValues .....	12-63
GetReportTraceNames .....	12-64
GetSolutionContexts .....	12-64
GetSolutionDataPerVariation .....	12-65
GetSweepNames .....	12-68
GetSweepUnits .....	12-69
GetSweepValues .....	12-70
GroupPlotCurvesByGroupingStrategy .....	12-71
ImportIntoReport .....	12-71
IsDataComplex .....	12-72
IsPerQuantityPrimarySweep .....	12-73
MovePlotCurvesToGroup .....	12-74

---

MovePlotCurvesToNewGroup .....	12-75
PasteReports .....	12-76
PasteTraces .....	12-77
Release Data .....	12-77
RenameReport .....	12-78
RenameTrace .....	12-79
ResetPlotSettings .....	12-80
SavePlotSettingsAsDefault .....	12-80
UpdateAllReports .....	12-81
UpdateReports .....	12-82
UpdateTraces .....	12-82
UpdateTracesContextAndSweeps .....	12-85
UnGroupPlotCurvesInGroup .....	12-87
<b>13 - Boundary and Excitation Module Script Commands in Maxwell .....</b>	<b>13-1</b>
General Commands Recognized by the Boundary/Excitations Module .....	13-2
AddAssignmentToBoundary .....	13-3
DeleteAllBoundaries .....	13-4
DeleteAllExcitations .....	13-5
DeleteBoundaries .....	13-5
GetBoundaryAssignment .....	13-6
GetBoundaries .....	13-7
GetBoundariesOfType .....	13-7
GetCoreLossEffect .....	13-8
GetCoreLossEffectOnField .....	13-9
GetDisplacementCurrent .....	13-9
GetEddyEffect .....	13-9
GetExcitations .....	13-10
GetExcitationsOfType .....	13-11
GetNumBoundaries .....	13-11
GetNumBoundariesOfType .....	13-12

---

GetNumExcitations .....	13-13
GetNumExcitationsOfType .....	13-13
ReassignBoundary .....	13-14
RemoveAssignmentFromBoundary .....	13-15
RenameBoundary .....	13-16
ReprioritizeBoundaries .....	13-17
<b>Script Commands for Creating and Modifying Boundaries in Maxwell .....</b>	<b>13-18</b>
AssignCylindricalHField .....	13-19
AssignDependent (2D) .....	13-20
AssignDependent (3D) .....	13-21
AssignFluxTangential .....	13-23
AssignImpedance (Maxwell) .....	13-23
AssignIndependent (Maxwell) .....	13-24
AssignInsulating .....	13-26
AssignRadiation (Maxwell) .....	13-26
AssignResistiveSheet .....	13-27
AssignSymmetry (Maxwell) .....	13-28
AssignTangentialHField .....	13-28
AssignThinLayer .....	13-29
AssignTouching .....	13-31
AssignVectorPotential .....	13-32
AssignZeroTangentialHField .....	13-33
EditCylindricalHField .....	13-34
EditDependent (Maxwell) .....	13-34
EditFluxTangential .....	13-35
EditImpedance (Maxwell) .....	13-35
EditIndependent (Maxwell) .....	13-36
EditInsulating .....	13-37
EditRadiation (Maxwell) .....	13-37
EditResistiveSheet .....	13-37

---

EditSymmetry .....	13-38
EditTangentialHField .....	13-39
EditThinLayer .....	13-40
EditTouching .....	13-41
EditVectorPotential .....	13-42
EditZeroTangentialHField .....	13-43
Script Commands for Creating and Modifying Excitations in Maxwell .....	13-44
AddTerminalsToWinding .....	13-45
AssignCharge .....	13-46
AssignCoilGroup .....	13-47
AssignCoilTerminal .....	13-48
AssignCoilTerminalGroup .....	13-48
AssignCurrent .....	13-50
AssignCurrentDensity .....	13-51
AssignCurrentDensityGroup .....	13-52
AssignCurrentDensityTerminal .....	13-52
AssignCurrentDensityTerminalGroup .....	13-53
AssignCurrentGroup .....	13-53
AssignFloating .....	13-53
AssignSink (Maxwell) .....	13-54
AssignVoltage (Maxwell) .....	13-54
AssignVoltageAPhi .....	13-55
AssignVoltageDrop .....	13-56
AssignVoltageDropGroup .....	13-57
AssignVoltageGroup .....	13-57
AssignVolumeChargeDensity .....	13-57
AssignWindingGroup .....	13-58
EditCharge .....	13-59
EditCoilTerminal .....	13-59
EditCurrent .....	13-59

---

EditCurrentDensity .....	13-61
EditCurrentDensityTerminal .....	13-61
EditExternalCircuit .....	13-61
EditFloating .....	13-62
EditSink .....	13-62
EditVoltage .....	13-62
EditVoltageAPhi .....	13-64
EditVoltageDrop .....	13-65
EditVolumeChargeDensity .....	13-65
EditWindingGroup .....	13-65
SetCoreLoss .....	13-66
SetEddyEffect .....	13-66
SetMagnetizationCompute .....	13-67
SetMinimumTimeStep .....	13-69
ShowWindow .....	13-69
<b>14 - Mesh Operations Module Script Commands .....</b>	<b>14-1</b>
General Commands Recognized by the Mesh Operations Module .....	14-2
DeleteOp .....	14-2
GetOperationNames .....	14-3
RenameOp .....	14-3
Script Commands for Creating and Modifying Mesh Operations .....	14-4
AssignCylindricalGapOp .....	14-5
AssignDensityControlOp .....	14-8
AssignEdgeCutLayerOp .....	14-11
AssignLengthOp .....	14-12
AssignModelResolutionOp .....	14-15
AssignSkinDepthLayerSetting .....	14-17
AssignSkinDepthOp .....	14-18
AssignTrueSurfOp .....	14-19
EditCylindricalGapOp .....	14-22

---

EditDensityControlOp .....	14-24
EditEdgeCutLayerOp .....	14-26
EditLengthOp .....	14-27
EditModelResolutionOp .....	14-30
EditSkinDepthOp .....	14-31
EditTrueSurfOp .....	14-33
InitialMeshSettings .....	14-35
<b>15 - Analysis Setup Module Script Commands .....</b>	<b>15-1</b>
ClearLinkedData (Module) .....	15-2
CopySetup .....	15-2
DeleteSetups .....	15-3
EditSetup [Maxwell] .....	15-4
ExportCircuit .....	15-10
ExportIcepak .....	15-15
ExportSolnData .....	15-15
GetSetupCount .....	15-16
GetSetups .....	15-17
GetSweepCount .....	15-18
GetSweeps .....	15-18
InsertSetup [Maxwell] .....	15-19
PasteSetup .....	15-33
PasteSweep .....	15-33
RenameSetup .....	15-34
ResetAllToTimeZero .....	15-34
ResetSetupToTimeZero .....	15-35
RevertAllToInitial .....	15-35
RevertAllToInitialTemperature .....	15-36
RevertSetupToInitial .....	15-36
RevertSetupToInitialTemperature .....	15-37
<b>16 - Optimetrics Module Script Commands .....</b>	<b>16-1</b>

---

General Commands Recognized by the Optimetrics Module .....	16-6
CopySetup .....	16-7
DeleteSetups [Optimetrics] .....	16-8
DistributedAnalyzeSetup .....	16-8
ExportDXConfigFile .....	16-9
ExportOptimetricsProfile .....	16-10
ExportOptimetricsResult .....	16-11
ExportParametricResults .....	16-12
GetSetupNames [Optimetrics] .....	16-13
GetSetupNamesByType [Optimetrics] .....	16-13
ImportSetup .....	16-14
PasteSetup [Optimetrics] .....	16-16
RenameSetup [Optimetrics] .....	16-16
SolveSetup [Optimetrics] .....	16-17
SolveAllSetup .....	16-18
Parametric Script Commands .....	16-18
EditSetup [Parametric] .....	16-19
ExportParametricSetupTable .....	16-19
GenerateVariationData [Parametric] .....	16-20
InsertSetup [Parametric] .....	16-21
Optimization Script Commands .....	16-26
EditSetup [Optimization] .....	16-26
InsertSetup [Optimization] .....	16-31
Sensitivity Script Commands .....	16-42
EditSetup [Sensitivity] .....	16-42
InsertSetup [Sensitivity] .....	16-50
Statistical Script Commands .....	16-54
EditSetup [Statistical] .....	16-54
InsertSetup [Statistical] .....	16-57
<b>17 - Solutions Module Script Commands .....</b>	<b>17-1</b>

---

DeleteSolutionVariation .....	17-1
GetValidISolutionList .....	17-2
<b>18 - Field Overlays Module Script Commands .....</b>	<b>18-1</b>
AddMarkerToPlot .....	18-2
CreateFieldPlot .....	18-4
Maxwell Field Line Trace Plot Examples .....	18-26
DeleteFieldPlot .....	18-35
EditSurfaceMeshSummaryData .....	18-36
ExportPlotImageWithViewToFile [Reporter] .....	18-40
ExportSurfaceMeshSummary .....	18-41
GetFieldPlotNames .....	18-42
ModifyFieldPlot .....	18-43
ModifyInceptionParameters .....	18-46
RenameFieldPlot .....	18-47
RenamePlotFolder .....	18-48
SetFieldPlotSettings .....	18-48
SetPlotFolderSettings .....	18-51
SetPlotsViewSolutionContext .....	18-57
UpdateAllFieldsPlots .....	18-58
UpdateQuantityFieldsPlots .....	18-58
<b>19 - Fields Calculator Script Commands .....</b>	<b>19-1</b>
AddNamedExpression .....	19-2
AddNamedExpr .....	19-3
CalcOp .....	19-4
CalcRead(deprecated) .....	19-4
CalculatorRead .....	19-5
CalcStack .....	19-6
CalculatorWrite .....	19-7
ChangeGeomSettings .....	19-8
ClcEval .....	19-9

---

ClcMaterial .....	19-10
ClearAllNamedExpr .....	19-11
CopyNamedExprToStack .....	19-11
DeleteNamedExpr .....	19-12
EnterComplex .....	19-13
EnterComplexVector .....	19-13
EnterLine .....	19-14
EnterPoint .....	19-15
EnterQty .....	19-16
EnterScalar .....	19-16
EnterScalarFunc .....	19-17
EnterSurf .....	19-18
EnterVector .....	19-18
EnterVectorFunc .....	19-19
EnterVol .....	19-20
ExportOnGrid [Fields Calculator] .....	19-20
ExportToFile [Fields Calculator] .....	19-23
GetTopEntryValue .....	19-26
LoadNamedExpressions .....	19-27
SaveNamedExpressions .....	19-28
<b>20 - Motion Setup Script Commands .....</b>	<b>A</b>
DeleteMotionSetup .....	A
ReassignMoving .....	B
Commands to Create and Edit the Band .....	B
AssignBand .....	B
EditMotionSetup .....	E
GetMotionSetupNames .....	G
Other Commands Recognized By the ModelSetup Module .....	G
GetSymmetryMultiplier .....	H
SetSymmetryMultiplier .....	H

---

<b>21 - Parameter Setup Script Commands .....</b>	<b>21-1</b>
General Parameter Setup Script Commands .....	21-1
Commands to Create and Edit Parameters .....	21-2
AddReduceOp .....	21-2
AssignForce .....	21-4
[Beta] AssignLayoutForce .....	21-5
AssignMatrix .....	21-7
AssignTorque .....	21-9
DeleteAllParameters .....	21-10
DeleteParameters .....	21-10
DeleteReduceMatrix .....	21-11
DeleteReduceOp .....	21-11
EditForce .....	21-12
[Beta] EditLayoutForce .....	21-12
EditMatrix .....	21-15
EditReduceOp .....	21-15
EditTorque .....	21-16
ReassignParameter .....	21-16
RenameParameter .....	21-17
RenameReduceMatrix .....	21-17
RenameReduceOp .....	21-18
<b>22 - User Defined Document Script Commands .....</b>	<b>22-1</b>
AddDocument .....	22-1
DeleteAllDocuments .....	22-4
DeleteDocument .....	22-5
EditDocument .....	22-5
GetDocumentDefinitionNames .....	22-7
GetDocumentNames .....	22-8
RenameDocument .....	22-8
SaveHtmlDocumentAs .....	22-9

---

SavePdfDocumentAs .....	22-10
UpdateAllDocuments .....	22-10
UpdateDocument .....	22-11
ViewHtmlDocument .....	22-12
ViewPdfDocument .....	22-13
Explication of a Sample UDD Script .....	22-13
Example Python Script: Defining a Document .....	22-15
<b>23 - User Defined Solutions Commands .....</b>	<b>23-1</b>
CreateUserDefinedSolution .....	23-1
DeleteUserDefinedSolutions .....	23-3
EditUserDefinedSolution .....	23-4
<b>24 - Network Data Explorer Script Commands .....</b>	<b>24-1</b>
AddDiffPair .....	24-4
Cascade (SPISim) .....	24-5
ClearDiffPairs .....	24-6
Clone .....	24-7
Close .....	24-8
Combine (SPISim) .....	24-9
Deembed (SPISim) .....	24-10
DeembedBack (SPISim) .....	24-11
DeembedFront (SPISim) .....	24-13
DisableDiffPairs .....	24-14
EnableDiffPairs .....	24-15
ExportCitiFile .....	24-16
ExportMatlab .....	24-18
ExportSpreadsheet .....	24-20
ExportTouchstone .....	24-22
ExportTouchstone2 .....	24-24
Extract (SPISim) .....	24-27
GetFrequencies .....	24-28

---

GetFrequencyCount .....	24-29
GetName .....	24-30
GetPortCount .....	24-30
GetPortNumber .....	24-31
GetPostProcSettings .....	24-32
GetVariation .....	24-33
HasSameData .....	24-34
LoadSolution .....	24-36
Open .....	24-37
Rename (SPISim) .....	24-38
Renormalize (SPISim) .....	24-39
Reorder .....	24-41
Reorder (SPISim) .....	24-42
Reset .....	24-43
SetAllPortImpedances .....	24-44
SetPortDeembedDistance .....	24-45
SetPortImpedance .....	24-46
SetPostProcSettings .....	24-47
Smooth .....	24-48
Stretch (SPISim) .....	24-49
Terminate .....	24-51
<b>25 - ComInstance Script Commands .....</b>	<b>25-1</b>
Callback Scripting Using ComInstance Object .....	25-1
ComInstance Functions .....	25-2
GetComponentName .....	25-2
GetInstanceID [Component Instance] .....	25-3
GetInstanceName [Component Instance] .....	25-4
GetParentDesign .....	25-4
GetPropHost .....	25-5
GetPropServerName .....	25-5

---

<b>26 - Schematic Scripting</b>	<b>26-1</b>
Method Format	26-2
Editor Scripting IDs	26-4
Format of IDs for different schematic objects:	26-4
Format for Components	26-5
Create Method List	26-6
CreateArc (Schematic Editor)	26-7
CreateCircle (Schematic Editor)	26-10
CreateComponent (Schematic Editor)	26-13
CreateGlobalPort (Schematic Editor)	26-15
CreateGround (Schematic Editor)	26-19
CreateLine (Schematic Editor)	26-22
CreatePagePort (Schematic Editor)	26-25
CreateIPort (Schematic Editor)	26-28
CreatePolygon (Schematic Editor)	26-31
CreateRectangle (Schematic Editor)	26-34
CreateText (Schematic Editor)	26-38
CreateWire (Schematic Editor)	26-42
General Method List	26-45
Activate (Schematic Editor)	26-47
AddPinGrounds (Schematic Editor)	26-48
AddPinIPorts (Schematic Editor)	26-51
AddPinPageConnectors (Schematic Editor)	26-52
AlignHorizontal (Schematic Editor)	26-53
AlignVertical (Schematic Editor)	26-54
BringToFront (Schematic Editor)	26-55
CloseEditor (Schematic Editor)	26-57
Copy (Schematic Editor)	26-57
CopyData [Schematic Editor]	26-58
CopySubdesign [Schematic Editor]	26-59

---

CreatePage (Schematic Editor) .....	26-60
Cut (Schematic Editor) .....	26-61
DeactivateOpen (Schematic Editor) .....	26-62
DeactivateShort (Schematic Editor) .....	26-63
Delete (Schematic Editor) .....	26-64
DeletePage (Schematic Editor) .....	26-65
ElectricRuleCheck (Schematic Editor) .....	26-66
ExportImage (Schematic Editor) .....	26-67
ExportNetlist .....	26-68
FindElements (Schematic Editor) .....	26-69
FitToBorder [Schematic Editor] .....	26-71
GridSetup (Schematic Editor) .....	26-72
FlipHorizontal (Schematic Editor) .....	26-75
FlipVertical (Schematic Editor) .....	26-76
Move (Schematic Editor) .....	26-78
NameNets (Schematic Editor) .....	26-79
PageBorders (Schematic Editor) .....	26-81
Pan (Schematic Editor) .....	26-81
Paste (Schematic Editor) .....	26-83
PasteData [Schematic Editor] .....	26-84
PasteDesign (Schematic Editor) .....	26-85
PushExcitations .....	26-87
Rotate (Schematic Editor) .....	26-90
SelectAll (Schematic Editor) .....	26-92
SelectPage (Schematic Editor) .....	26-93
SetPageData [Schematic Editor] .....	26-94
SendToBack .....	26-97
ShowVariableBlock (Schematic Editor) .....	26-98
SortComponents (Schematic Editor) .....	26-98
Wire (Schematic Editor) .....	26-99

---

ZoomArea (Schematic Editor) .....	26-100
ZoomIn (Schematic Editor) .....	26-101
ZoomOut (Schematic Editor) .....	26-102
ZoomPrevious (Schematic Editor) .....	26-103
ZoomToFit (Schematic Editor) .....	26-104
<b>Property Method List .....</b>	<b>26-105</b>
ChangeProperty (Schematic Editor) .....	26-105
GetEvaluatedPropertyValue (Schematic Editor) .....	26-106
GetProperties (Schematic Editor) .....	26-107
GetPropertyAttribute [Schematic Editor] .....	26-109
GetPropertyValue (Schematic Editor) .....	26-110
SetPropertyValue (Schematic Editor) .....	26-111
<b>Information Method List .....</b>	<b>26-112</b>
GetAllPorts .....	26-113
GetCompInstanceFromRefDes (Schematic Editor) .....	26-113
GetComponentInfo (Schematic Editor) .....	26-114
GetComponentPins (Schematic Editor) .....	26-116
GetComponentPinInfo (Schematic Editor) .....	26-117
GetComponentPinLocation [Schematic Editor] .....	26-118
GetEditorName (Schematic Editor) .....	26-119
GetNetConnections (Schematic Editor) .....	26-120
GetNumPages [Schematic Editor] .....	26-121
GetPortInfo (Schematic Editor) .....	26-122
GetSelections (Schematic Editor) .....	26-123
GetSignals (Schematic Editor) .....	26-124
GetWireConnections (Schematic Editor) .....	26-126
GetWireInfo (Schematic Editor) .....	26-128
GetWireSegments (Schematic Editor) .....	26-130
<b>27 - Definition Manager Script Commands .....</b>	<b>27-1</b>
AddMaterial (Maxwell) .....	27-2

---

CloneMaterial .....	27-9
ComputeCoreLossCoefficients .....	27-10
DoesMaterialExist .....	27-12
EditMaterial (Maxwell) .....	27-13
GetProjectMaterialNames .....	27-27
ExportMaterial .....	27-27
RemoveMaterial .....	27-28
RemoveUnusedDefinitions .....	27-29
Component Manager Script Commands .....	27-31
Add [component manager] .....	27-32
AddDynamicNPortData [component manager] .....	27-50
AddNPortData [component manager] .....	27-54
AddSolverOnDemandModel .....	27-59
ClearSolutionCache [component manager] .....	27-60
Edit [component manager] .....	27-60
EditSolverOnDemandModel .....	27-87
EditWithComps [component manager] .....	27-87
Export [component manager] .....	27-99
GetData [component manager] .....	27-100
GetNames [component manager] .....	27-101
GetNPortData [component manager] .....	27-102
GetSolverOnDemandData .....	27-105
GetSolverOnDemandModelList .....	27-105
IsUsed [component manager] .....	27-105
Remove [component manager] .....	27-106
RemoveSolverOnDemandModel .....	27-107
RemoveUnused [component manager] .....	27-107
Update Dynamic Link [component manager] .....	27-109
Material Manager Script Commands .....	27-109
GetData [material manager] .....	27-110

---

GetNames [material manager] .....	27-110
GetProperties [material manager] .....	27-112
IsUsed [material manager] .....	27-113
RemoveUnused [material manager] .....	27-113
Model Manager Script Commands .....	27-114
Add [model manager] .....	27-115
ConvertToDynamic .....	27-125
ConvertToParametric .....	27-126
Edit [deprecated] .....	27-126
EditWithComps [model manager] .....	27-126
Export [model manager] .....	27-138
GetData [model manager] .....	27-139
GetNames [model manager] .....	27-140
IsUsed [model manager] .....	27-141
Remove [model manager] .....	27-141
RemoveUnused [model manager] .....	27-143
Network Data Explorer Manager Script Commands .....	27-144
ExportFullWaveSpice .....	27-144
ExportNetworkData .....	27-146
ExportNMFDATA .....	27-148
Symbol Manager Script Commands .....	27-148
Add [symbol manager] .....	27-149
BringToFront [symbol manager] .....	27-159
Edit [deprecated] .....	27-160
EditWithComps [symbol manager] .....	27-160
Export [symbol manager] .....	27-172
GetData [symbol manager] .....	27-173
GetNames [symbol manager] .....	27-173
IsUsed [symbol manager] .....	27-174
Remove [symbol manager] .....	27-174

---

RemoveUnused [symbol manager] .....	27-175
Add [footprint manager] .....	27-176
Edit [footprint manager] .....	27-199
EditWithComps [footprint manager] .....	27-199
Export [footprint manager] .....	27-217
GetData [footprint manager] .....	27-218
GetNames [footprint manager] .....	27-219
IsUsed [footprint manager] .....	27-219
Remove [footprint manager] .....	27-220
RemoveUnused [footprint manager] .....	27-221
Add [padstack manager] .....	27-222
Edit [padstack manager] .....	27-228
EditWithComps [padstack manager] .....	27-235
Export [padstack manager] .....	27-243
GetData [padstack manager] .....	27-244
GetNames [padstack manager] .....	27-245
IsUsed [padstack manager] .....	27-245
Remove [padstack manager] .....	27-246
RemoveUnused [padstack manager] .....	27-247
Script and Library Scripts .....	27-248
AddScript .....	27-248
EditScript .....	27-249
ExportScript .....	27-250
RemoveScript .....	27-251
ModifyLibraries .....	27-252
<b>28 - Definition Editor Script Commands .....</b>	<b>28-1</b>
Symbol Editor Scripts .....	28-1
AddLevel (Symbol Editor) .....	28-3
AlignHorizontal (Symbol Editor) .....	28-4
AlignVertical (Symbol Editor) .....	28-5

---

BringToFront (Symbol Editor) .....	28-6
ChangeProperty (Symbol Editor) .....	28-6
CloseEditor (Symbol Editor) .....	28-19
Copy (Symbol Editor) .....	28-20
CreateArc (Symbol Editor) .....	28-20
CreateCircle (Symbol Editor) .....	28-21
CreateImage (Symbol Editor) .....	28-22
CreateLine (Symbol Editor) .....	28-23
CreatePin (Symbol Editor) .....	28-24
CreatePolygon (Symbol Editor) .....	28-25
CreateRectangle (Symbol Editor) .....	28-26
CreateText (Symbol Editor) .....	28-27
Cut (Symbol Editor) .....	28-27
Delete (Symbol Editor) .....	28-28
DisplayPinNames (Symbol Editor) .....	28-29
ExportFile (Symbol Editor) .....	28-29
FlipHorizontal (Symbol Editor) .....	28-30
FlipVertical (Symbol Editor) .....	28-31
GetProperties (Symbol Editor) .....	28-31
GetPropertyValue (Symbol Editor) .....	28-32
GetVisibleLevels (Symbol Editor) .....	28-32
GridSetup (Symbol Editor) .....	28-33
Move (Symbol Editor) .....	28-34
Pan (Symbol Editor) .....	28-35
Paste (Symbol Editor) .....	28-35
Redo (Symbol Editor) .....	28-36
RemoveLevels (Symbol Editor) .....	28-37
RemovePort (Symbol Editor) .....	28-37
Rotate (Symbol Editor) .....	28-38
Save (Symbol Editor) .....	28-39

---

SelectAll (Symbol Editor) .....	28-39
SendToBack (Symbol Editor) .....	28-40
SetActiveLevel (Symbol Editor) .....	28-40
SetInitialLevels (Symbol Editor) .....	28-41
SetPropertyValue (Symbol Editor) .....	28-42
SetVisibleLevels (Symbol Editor) .....	28-42
ToggleLevel (Symbol Editor) .....	28-43
Undo (Symbol Editor) .....	28-44
ZoomArea (Symbol Editor) .....	28-45
ZoomIn (Symbol Editor) .....	28-46
ZoomOut (Symbol Editor) .....	28-46
ZoomPrevious (Symbol Editor) .....	28-47
ZoomToFit (Symbol Editor) .....	28-47
<b>29 - Core Global Script Context Commands</b> .....	<b>29-1</b>
AddErrorMessage .....	29-1
AddFatalMessage .....	29-2
AddInfoMessage .....	29-2
AddWarningMessage .....	29-3
LogDebug .....	29-3
LogError .....	29-4
<b>30 - Example Scripts</b> .....	<b>30-1</b>
VBScript Example Scripts .....	30-1
Variable Helix Script .....	30-1
Data Export Script .....	30-5
IronPython Example Scripts .....	30-8
BH Coordinates Python Script .....	30-8
Script Contents .....	30-9
Posco_BH_Curve.tab Contents .....	30-11
Equation Based Curve Python Script .....	30-13
<b>Index</b> .....	<b>Index-1</b>

---



# 1 - Introduction to Scripting

Using scripts is a fast, effective way to accomplish tasks you want to repeat. When you execute a script, the commands in the script are performed in the order in which they appear.

Electronics Desktop can record scripts in VBScript or IronPython, and can run external scripts written in VBScript, IronPython, CPython, or JavaScript. Additionally, it contains an IronPython command shell for executing scripts.

When running Ansys Electronics Desktop from the command line, scripts can be written in any language that provides Microsoft COM methods.

The following sections contain more information about scripting:

- [Scripting Help Conventions](#) – explains the layout of the scripting help.
- [Introduction to VBscript](#) – provides a broad overview of VBscript
- [Introduction to IronPython](#) – provides a broad overview of IronPython.
- [Introduction to C-Python](#) – provides guidance on using C-Python for Ansys Electronics Desktop scripts.
- [Ansys Electronics Desktop Scripting](#) – details instructions and tips for running, recording, and working with scripts in Electronics Desktop.
- [PyAEDT](#) (Beta) – a Python library that interacts directly with the AEDT API to make scripting simpler for the end user.

## Scripting Help Conventions

The majority of this guide lists individual script commands using the following format.

### [ScriptName]

[Description of script use.]

<b>UI Access</b>	[UI commands corresponding to the script command, if any.]
<b>Parameters</b>	[List of arguments taken by the script command, if any. Includes argument types and brief descriptions.]
<b>Return Value</b>	[The script's return value, if any.]

<b>Python Syntax</b>	[Correct syntax for the command in Python. Arguments are enclosed in angle brackets (<>).]
<b>Python Example</b>	[ Sample script ]

<b>VB Syntax</b>	[Correct syntax for the command in VBscript. Arguments are enclosed in angle brackets (<>)]
<b>VB Example</b>	[Sample script]

## Variable Types

The following data types are used throughout the help:

- <**string**> – use within quotation marks.
- <**bool**> – boolean value; should be set to either True or False.
- <**int**> – an integer. For example, 1.
- <**double**> – a double precision value. For example, 1.2.
- <**array**> – in VBscript, an array; in IronPython, a list contained in square brackets.
- <**value**> – can be an integer, string, or VBscript variable, depending on context.

## Introduction to VBScript

Ansys Electronics Desktop can use Microsoft® Visual Basic® Scripting (VBScript) to record macros. VBScript is based on the Microsoft Visual Basic programming language.

This chapter provides an overview of key VBScript components.

[Simple and Composite Names](#)

[VBScript Variables](#)

[VBScript Operators](#)

[Controlling Program Execution](#)

[Looping Through Code](#)

[VBScript Procedures](#)

[Converting Between Data Types](#)

[Including Scripts](#)

[Aborting Scripts](#)

[Interacting with a Script](#)

[Recommended VBScript References](#)

[Sample HFSS Script](#)

[Sample Circuit Script](#)

[Sample Q3D Script](#)

For more details about VBScript, please see the *Recommended VBScript References* section at the end of this chapter.

## Simple and Composite Names

Components, symbols, footprints, models, and padstacks possess either "simple" names or "composite" names. Composite names are used to distinguish items from libraries that may possess the same simple name. A composite name is created by combining an item's library name with its simple name. Composite names for definitions are unique, but simple names are not.

- Composite names are used by definition manager script commands to uniquely identify script definitions.
- Materials and scripts do not have composite names, so project definitions for these items must possess a unique simple name.
- The format of a composite name is `LibraryName:SimpleDefinitionName`. For example, the composite name for the component "CAP\_in" in the system library `Nexxim Circuit Elements\Capacitors` is "`Nexxim Circuit Elements\Capacitors:CAP_in`".
- The format of a composite name in a project is `OriginLibraryName:SimpleDefinitionName`. For example, the composite name for the project component "CAP\_" that was originally from the system library `Nexxim Circuit Elements\Capacitors` is "`Nexxim Circuit Elements\Capacitors:CAP_`".
- Not all definitions in a project have a library of origin. Newly added definitions do not have a library of origin, and project definitions whose names are changed do not have a library of origin (even if they did before the name change). As a result, the composite name for items without a library of origin is the item's simple name itself. For example, the composite name for the project component "CAP\_" that came from a system library and was renamed to "MyCAP\_" is "MyCAP\_".

To construct a composite name, select **Tools > Edit Configured Libraries > Components** to open the **Edit Libraries** dialog box. The subnames used to construct a composite name can be found in the **Name** and **Origin** columns that correspond to a particular component. The **Origin** column contains the library portion of the composite name, while the **Name** column contains the simple portion of the composite name.

## VBScript Variables

A VBScript variable is a placeholder representing information that may change during the time your script is running. Variables are useful because they let you assign a short and easy to remember name to each piece of data you plan to use. Use a variable name in a script to view or modify its value.

## Declaring Variables

To declare variables explicitly in a script, use the Dim, Public, or Private statements. For example:

```
Dim box_xsize
```

After declaring a variable, you can assign information to it. For example:

```
box_xsize = "3mm"
```

You can declare multiple variables by separating each variable name with a comma. For example:

```
Dim Top, Bottom, Left, Right
```

You can also declare a variable implicitly by simply using its name in your script. Doing so is not generally a good practice because you could misspell the variable name in one or more places, causing unexpected results when your script is run. For that reason, the **Option Explicit** statement is available to require explicit declaration of all variables. The **Option Explicit** statement should be the first statement in your script.

### Declaring Variables in Python

Python does not require you to declare variables before you assign a value to them.

You can directly assign information to it. For example:

```
box_xsize = "3mm"
```

### Variable Naming Conventions

You should use names that are short but intuitive and easy to remember. Use the following conventions for naming variables in VBScript:

- Begin with an alphabetic character.
- Cannot contain an embedded period.
- Must not exceed 255 characters.
- Must be unique in the scope in which it is declared.
- Do not use VBScript keywords.

### Scope and Lifetime of Variables

Variables at the script level are available to all procedures within the script. At the procedure level, variables are available only within the procedure. It has local scope and is a procedure-level variable.

The lifetime of a variable depends on how long it exists. The script-level variables exist from declaration until the end of the script. A procedure-level variable exists only as long as you are in the procedure and is destroyed when the procedure exits.

### Array Variables

Create an array variable when you want to assign more than one related value to a single variable. An array variable contains a series of values. For example:

```
Dim Primitives(2)
```

All arrays in VBScript are zero-based, so the array above actually contains 3 elements. You assign data to each of the array's elements using an index into the array. Data can be assigned to the elements of an array as follows:

```
Primitives(0) = "Box1"  
Primitives(1) = "Cone1"  
Primitives(2) = "Cylinder1"
```

Similarly, the data can be retrieved from any element using an index into a particular array element. For example:

```
one_prim = Primitives(1)
```

You can also use the **Array** function to assign an array of elements to a variable. For example:

```
Dim Primitives  
Primitives = Array ("Box1", "cone1", "Cylinder1")
```

#### Note:

When using the **Array** function, do not use parentheses on the variable when it is declared. For example, use **Dim myarray**, not **Dim myarray()**.

If you do not know the size of the array at declaration or the size changes during the time your script is running, you can use dynamic arrays. They are declared without size or number of dimensions inside the parentheses. For example:

```
Dim FirstArray()  
ReDim SecondArray()
```

To use a dynamic array, you must subsequently use **ReDim** to determine the number of dimensions and the size of each dimension. You can also use the **Preserve** keyword to preserve the contents of the array as the resizing takes place.

```
ReDim FirstArray(25)  
ReDim Preserve FirstArray(30)
```

## VBScript Operators

VBScript provides operators, which are grouped into these categories: arithmetic operators, comparison operators, and logical operators.

Please see the online *VBScript User's Guide* for more details.

## Operator Precedence

When several operations occur in an expression, each part is evaluated and resolved in a pre-determined order, called operator precedence. You can use parentheses to override the order of precedence and force some parts of an expression to be evaluated before others. Operations within parentheses are always performed before those outside the parentheses. Within parentheses, however, standard operator precedence is maintained.

When an expression contains operators from more than one category, they are evaluated in the following order:

1. [Arithmetic Operators](#)
2. [String Concatenation Operator \(&\)](#)
3. [Comparison Operators](#)
4. [Logical Operators](#)

**Arithmetic** operators within a single expression are evaluated in the following order of precedence:

1. Exponentiation (^)
2. Multiplication and Division (\*,/): These two operators are of equal precedence and are evaluated in the left-to-right order in which they appear within the expression.
3. Integer Division (\)
4. Modulus Arithmetic (Mod)
5. Addition and Subtraction (+,-): These two operators are of equal precedence and are evaluated in the order in which they appear within the expression.

If the same arithmetic operator appears multiple times within a single expression, they are evaluated in the left-to-right order in which they appear.

**Comparison** operators all have equal precedence and are evaluated in the left-to-right order in which they appear within the expression.

**Logical** operators all have equal precedence and are evaluated in the left-to-right order in which they appear within the expression.

### Arithmetic Operators

Following is a list of VBScript's arithmetic operators:

Symbol	Description
^	Exponentiation
-	Unary negation
*	Multiplication
/	Division

---

\	Integer division
Mod	Modulus arithmetic
+	Addition
-	Subtraction

**Note:**

For the order of precedence for these operators, see [Operator Precedence](#).

**Comparison Operators**

Following is a list of VBScript's comparison operators:

Symbol	Description
=	Equality
<>	Inequality
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
Is	Object equivalence

**Note:**

All comparison operators have the same precedence. When multiple comparisons exist in a single expression, evaluate them in the left-to-right order in which they appear.

**Logical Operators**

Following is a list of VBScript's logical operators:

Symbol	Description
Not	Logical negation
And	Logical conjunction
Or	Logical disjunction
Xor	Logical exclusion

Eqv	Logical equivalence
Imp	Logical implication

**Note:**

All logical operators have the same precedence. When multiple logical operators exist in a single expression, evaluate them in the left-to-right order in which they appear.

## Controlling Program Execution

You can use conditional statements to control the flow of a script. There are two types of conditional statements in VBScript:

- [If...Then...Else](#)
- [Select Case](#)

### Using If...Then...Else

Following is an example that demonstrates the If...Then...Else conditional statement:

```
If obj = "Box1" Then  
    <statements to execute>  
ElseIf obj = "Cylinder1" Then  
    <statements to execute>  
Else  
    <statements to execute>  
End If
```

### Using Select Case

Following is an example that demonstrates the Select Case conditional statement:

```
Select Case primitive_name  
    Case "Box1"  
        <statements to execute>  
    Case "Cylinder1"  
        <statements to execute>  
    Case Else  
        <statements to execute>
```

```
End Select
```

## Looping Through Code

Looping allows you to run a group of statements repeatedly. There are two types of loops:

- [For...Next](#): Uses a counter to run statements a specified number of times.
- [Do...Loop](#): Loops while or until a condition is True.

When using conditional statements that test for zero voltage/current, it is important to note that a real voltage or current should not be trusted to be exactly zero, even when it should be. Typically, the voltage or current is often on the order of 'epsilon' (1e-16) or smaller; hence, it is nonzero in value.

### Using a For...Next Loop

The For...Next type of loop allows you to run a group of statements repeatedly. It uses a counter to run statements a specified number of times. Following is an example that demonstrates the For...Next loop:

```
For variable = start To end  
    <statements to execute>  
Next
```

You can exit early from a For...Next loop with the Exit For statement.

### Using a Do Loop

You can use Do...Loop statements to run a block of statements until (or while) a condition is true.

#### Repeating Statements While a Condition is True

Use the While keyword to check a condition in a Do...Loop statement. The syntax is as follows:

```
Do While condition  
    <statements to execute>  
Loop
```

#### Repeating a Statement Until a Condition Becomes True

Following is the syntax:

```
Do Until condition  
    <statements to execute>  
Loop
```

You can exit early from a loop by using the Exit For statement.

## VBScript Procedures

In VBScript, there are two kinds of procedures, [Sub](#) and [Function](#). These procedures are called by name, they can receive arguments, and each performs a specific task with a group of VBScript statements. If there is no argument, then the Sub or Function statement must include an empty set of parentheses.

### Function Procedures

A Function returns a value by assigning a value to its name in one or more statements. Following is the syntax of a Function:

```
Function FunctionName([arguments])  
    <Function statements>  
End Function
```

### Sub Procedures

A Sub procedure is like a function procedure, except that it does not return a value through its name. Following is the syntax of a Sub:

```
Sub ProcedureName([arguments])  
    <Procedure statements>  
End Sub
```

## Converting Between Data Types

To convert data from one subtype to another, use the following VBScript functions:

<b>CStr</b>	Syntax: CStr(variablename).  Converts variablename to a string. For example, it can be used to convert the number 2.5 to the string "2.5".
<b>CBool</b>	Syntax: CBool(variablename).  Converts variablename to a boolean. If variablename is 0 or "0", CBool returns False. Otherwise it returns True.
<b>CDbl</b>	Syntax: CDbl(variablename).  Converts variablename to a double precision number. For example, it can be used to convert the string "2.5" to the number 2.5.
<b>CInt</b>	Syntax: CInt(variablename).  Converts variablename to an integer.

## Including Scripts

You can include one script within another using the following command:

---

```
#include "<scriptfilename>"
```

Where scriptfilename is the full path name to a file that contains script text, or is the name of a script in the project library or script library (listed in the project window under the Definitions/Scripts directory).

The command works for VBScript, JScript, and for the following:

- Scripts in the project library that are run by right-clicking the script icon in the project window and choosing **Run Script**
- Scripts in files that are external are run by choosing **Tools> Run Script**
- Scripts that are specified as callbacks in the **Property** dialog box
- Scripts that are run to draw parameterized footprints in layout

An include command can be placed anywhere in a script, but for readability it is recommended that commands be placed at the beginning of a file. The same script can be included multiple times without error, and circular inclusions will be ignored.

## Aborting Scripts

You can abort a script that is running in the desktop simply by pressing the ESC key. Terminating a script in this manner works for each of the following:

- Scripts in the project library that are run by right-clicking the script icon in the project window and choosing **Run Script**.
- Scripts in files that are external can be run by choosing **Tools > Run Script**.
- Scripts that are specified as callbacks in the **Property** dialog box.
- Scripts that are run to draw parameterized footprints in layout.

## Interacting with a Script

VBScript provides two functions that enable you to interact with a script while it is running:

the InputBox function and the MsgBox function.

The InputBox function displays a dialog box with an input field. The value that is typed into the input field is returned. For example:

```
Dim users_string  
  
users_string = InputBox ("text prompt", "title of the pop-up dialog _  
box", "default text for the input box")
```

The last two arguments to the function are optional.

The MsgBox function shows a message and returns a number based on the button the user presses. For example:

```
MsgBox ("message text")
```

## Recommended VBScript References

Microsoft Corporation. *VBScript User's Guide*.

Available <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/vbstutor.asp>.

Childs, M., Lomax, P., and Petrusha, R. *VBScript in a Nutshell: A Desktop Quick Reference*.

May 2002. O'Reilly & Associates. ISBN: 1-56592-720-6.

## Introduction to IronPython

IronPython is an implementation of the Python programming language targeting the .NET runtime. What this means in practical terms is that IronPython uses the Python programming language syntax and standard python libraries and can additionally use .NET classes and objects to give one the best of both worlds. This usage of .NET classes is fairly seamless in that a class defined in a .NET assembly can be used as a base class of a python class.

### Scope

Functioning as a tutorial on Python or IronPython is way out of the scope of this document. There are several excellent resources online that do a very good job in that regard. This document only attempts to provide a limited introduction to IronPython as used to script Ansys EM products.

This document is also not a tutorial on the scripting of Ansys EM products. It complements the existing scripting guide (available from a product's Help menu) and provides a pythonic interpretation of that information. The reader might have to refer to either the scripting guide or recorded samples of VBScript to follow some of the sections.

### Python compatibility

The version of IronPython in use is **2.7** and built on the .NET framework version 4.0: this version targets **Python 2.7** language compatibility. While most python files will execute under IronPython with no changes, python libraries that make use of extensions written in the C programming language (NumPy or SciPy for instance), are not expected to work under IronPython. In such cases, it might be possible to locate .NET implementation of such libraries or explore the use of IronClad.

(<http://code.google.com/p/ironclad/>).

### Advantages of IronPython

The advantages that IronPython use provides are significant:

- Python has a large eco-system with plenty of supporting libraries, Visual IDEs and debuggers. It is actively developed and enhanced.

- IronPython, in addition, has access to the entire .NET eco system. This allows us, for instance, to create a modern GUI using the **System.Windows.Forms** assembly from IronPython code and call any other .NET assembly for that matter.
- The use of IronPython's technologies enables the ability to interactively script Desktop (feature in development). This allows better discovery of the scripting APIs as well as directly programming to the scripting API in python, a language more tractable and platform independent compared with VBScript.
- The Python syntax of dictionaries is somewhat easier to read and write when supplying arguments to the scripting methods.

This document describes IronPython briefly and then goes on to describe the desktop provided IronPython scripting console and scripting with IronPython. You can open an IronPython Command Window by clicking **Tools > Open Command Window**.

---

 IronPython Command Window

---

```
=====
ElectronicsDesktop 2019.3.0
IronPython 2.7.0.40 on .NET 4.0.30319.42000
-----
- With Tab completion
- dir()      - lists all available methods and objects
- dir(obj)   - lists all available attributes/methods on obj
- help(obj)  - provides available help on a method or object
- tutorial() - provides more help on using the console
-----
try executing "dir(oDesktop)" or dir_sig(oDesktop,"ver")
=====
>>> |
```

---

The document assumes that you know how desktop scripting works using VBScript or JavaScript.

[Introduction to IronPython](#)

[IronPython Mini Cookbook](#)

[Translating Script Commands from VBScript to IronPython](#)

[Scripting Using Iron Python](#)

[Standalone IronPython and Desktop IronPython](#)

[IronPython Examples](#)

[Creating User Defined Primitives and User Defined Models in Python Scripts](#)

## Introduction to IronPython

IronPython is an implementation of the Python programming language targeting the .NET runtime. What this means in practical terms is that IronPython uses the Python programming language syntax and standard python libraries and can additionally use .NET classes and objects to give one the best of both worlds. This usage of .NET classes is fairly seamless in that a class defined in a .NET assembly can be used as a base class of a python class.

### Scope

Functioning as a tutorial on Python or IronPython is way out of the scope of this document. There are several excellent resources online that do a very good job in that regard. This document only attempts to provide a limited introduction to IronPython as used to script Ansys EM products.

This document is also not a tutorial on the scripting of Ansys EM products. It complements the existing scripting guide (available from a product's Help menu) and provides a pythonic interpretation of that information. The reader might have to refer to either the scripting guide or recorded samples of VBScript to follow some of the sections.

### Python compatibility

The version of IronPython in use is **2.7** and built on the .NET framework version 4.0: this version targets **Python 2.7** language compatibility. While most python files will execute under IronPython with no changes, python libraries that make use of extensions written in the C programming language (NumPy or SciPy for instance), are not expected to work under IronPython. In such cases, it might be possible to locate .NET implementation of such libraries or explore the use of IronClad.

(<http://code.google.com/p/ironclad/>).

### Advantages of IronPython

The advantages that IronPython use provides are significant:

- Python has a large eco-system with plenty of supporting libraries, Visual IDEs and debuggers. It is actively developed and enhanced.
- IronPython, in addition, has access to the entire .NET eco system. This allows us, for instance, to create a modern GUI using the **System.Windows.Forms** assembly from IronPython code and call any other .NET assembly for that matter.
- The use of IronPython's technologies enables the ability to interactively script Desktop (feature in development). This allows better discovery of the scripting APIs as well as directly programming to the scripting API in python, a language more tractable and platform independent compared with VBScript.
- The Python syntax of dictionaries is somewhat easier to read and write when supplying arguments to the scripting methods.

This document describes IronPython briefly and then goes on to describe the desktop provided IronPython scripting console and scripting with IronPython. You can open an IronPython Command Window by clicking **Tools > Open Command Window**.



IronPython Command Window

```
=====
ElectronicsDesktop 2019.3.0
IronPython 2.7.0.40 on .NET 4.0.30319.42000
-----
- With Tab completion
- dir()      - lists all available methods and objects
- dir(obj)   - lists all available attributes/methods on obj
- help(obj)  - provides available help on a method or object
- tutorial() - provides more help on using the console
-----
try executing "dir(oDesktop)" or dir_sig(oDesktop,"ver")
=====
```

>>> |

The document assumes that you know how desktop scripting works using VBScript or JavaScript.

[Introduction to IronPython](#)

[IronPython Mini Cookbook](#)

[Translating Script Commands from VBScript to IronPython](#)

[Scripting Using Iron Python](#)

[Standalone IronPython and Desktop IronPython](#)

[IronPython Examples](#)

[Creating User Defined Primitives and User Defined Models in Python Scripts](#)

## IronPython Mini Cookbook

This topic presents simple counterparts between IronPython and VBScript. It does not provide a full tutorial on IronPython syntax. Because IronPython is a Python implementation, you can consult Python documentation for additional information.

### Comments

VBScript	IronPython
Comments start with a single quote: 'comment	Comments start with a hash: # comment

### Assigning/Creating Variables

VBScript	IronPython
Declare with a Dim: <pre>Dim doc</pre> Assignment then needs a Set instruction: <pre>Set doc = app.GetActiveProject()</pre>	No Set syntax. Simply create and assign: <pre>doc = app.GetActiveProject()</pre>

### Create Lists/Arrays

VBScript	IronPython
Declare as array of String with 11 indices, from 0 through 10: <pre>Dim myArray(0 to 10) as String</pre> <pre>myArray(0) = "Hello"</pre> <pre>myArray(1) = "bye"</pre> Declare an array with no size: <pre>Dim array2() as String</pre> Re-dimension an array once size is known: <pre>ReDim array2(0 to 2) as String</pre> <pre>array2(0) = "this"</pre> <pre>array2(1) = "also"</pre>	Declare an empty array: <pre>myEmptyArray = []</pre> Declare an array and initialize it with 5 ints: <pre>myInitiatedArray = [ 1, 2, 3, 4, 5]</pre> Python lists can have items of any type and there is no pre-declaration. Declare an array and init with mixed types: <pre>mixed = ["hello", 1 ,2 ["nested"]]</pre> Append to an array: <pre>mixed.append( 3.5 )</pre>

### Create Dictionaries/Maps

VBScript	IronPython
<p>Declare with a Dim:</p> <pre>Dim dict</pre> <p>Use the CreateObject function with ProgID Scripting.Dictionary:</p> <pre>Set dict = CreateObject _ ("Scripting.Dictionary")</pre> <p>Add items using the object, key, item syntax:</p> <pre>dObject.Add key, item</pre>	<p>An IronPython dictionary is a collection of name value pairs. Just like arrays, there is no restriction on the keys or the values. <u>For purposes of Ansys EM scripting, however, all keys must be strings</u></p> <p>Delimiters are curly braces. Use a colon between the key and the value. Separate key value pairs with a comma:</p> <pre>myDict = {     "a" : 1,     "b" : "hello there",     "c" : [ 1, 2, "abc"] }</pre>

### Boolean Values

VBScript	IronPython
<p>Boolean literals are in lower case:</p> <pre>true false</pre>	<p>The first letter is capitalized:</p> <pre>True False</pre>

### Converting Numbers to Strings and Vice Versa

VBScript	IronPython
<p>Use <b>CInt</b>, <b>CDbl</b>, <b>CBool</b>, <b>CLng</b> to convert the string representation to the number representation. Use <b>IsNumber</b> to check before conversion:</p> <pre>Dim nStr = "100" Dim n = CInt(nStr)</pre> <p>Use <b>CStr</b> to convert a number to its string representation:</p> <pre>Dim v, vStr v = 100 vStr = CStr(v)</pre>	<p>Use <b>integer()</b> or <b>float()</b> or <b>double()</b> functions to cast a string CONTAINING the string representation of whatever you are casting to:</p> <pre>strInt = "3" intVal = int(strVal) floatVal = float(strVal)</pre> <p>Invoke the <b>str()</b> function with the int/float values as needed. You can alternately use the string formatting method listed below:</p> <pre>strVal = str(42) strVal = str(42.345)</pre>

### String Formatting/Concatenation

VBScript	IronPython
<p>String concatenation uses the &amp; operator:</p> <pre>Dim allStr, str1 str1 = " how are you" allStr = "Hello " &amp; " There" &amp; str1</pre> <p>There seems to be no direct string formatting function in VBScript. Using string concatenation or using Replace are the two built-in options:</p> <pre>Dim fmt = "{1} climbs stalk {2}" Dim str = Replace(fmt, "{1}", "jack") str = Replace(str, "{2}", 10)</pre>	<p>If you have two strings, you can always concatenate them using the '+' operator:</p> <pre>str1 = "hello" str2 = "world" str12 = str1 + " " + str2</pre> <p>If you have different types (for instance a string and an int), you must use the string formatting commands. When formatting multiple arguments, they must be entered as a tuple ( item1, item2, ):</p> <pre>num = 10 str3 = "%s climbs stalk %d" % ("jack", num) str4 = "%d stalks" % num</pre>

### Looping over Lists

VBScript	IronPython
<pre>Dim myArray(0 to 2) as String myArray(0) = "alpha" myArray(1) = "bravo" myArray(2) = "charlie"  For Each i in myArray Print i Next</pre>	<pre>vals = [1, 3, 3.456]  def process(val):     return 2*val  for i in vals:     print i     print " -&gt; " process(i)</pre>

### Looping over a Range

VBScript	IronPython
<p>To loop over a range, specify start, end, and step:</p> <pre>For i = 0 To 10 Step 1 Print i Next</pre>	<pre>for i in range(0, 10):     print i</pre>

**Related Topics:**[Indentation in IronPython](#)[Methods in IronPython](#)[Introduction to IronPython](#)[Translating Script commands from VBScript to IronPython](#)[Scripting Using Iron Python](#)[IronPython Samples](#)**Indentation in IronPython**

Python is a language where white space (spaces and tabs) is syntactically significant. You must understand the basics of indentation before scripting in python.

Any statement that introduces a block of code should be written so that every line of the block has the same indent (leading spaces or tabs) and the indent should be at least one more than the indent of the introducing statement.

**Note:**

Python recommends the use of spaces over tabs.

**Indenting Functions**

Define a function that starts at 0 indentation:

```
def multInt(a,b) :
```

Every line following `def multInt` that is expected to be a part of the function, must be indented to line up with the function.

```
def multInt(a,b) :  
    return a
```

**Indenting If Conditions**

Each line that belongs to the body of this function should have an indent that is more than the indent used by the if statement.

```
def multInt(a,b) :  
    if a%2 == 0 :  
        return (a * b) + 100  
    else :  
        return (a * b) + 1000
```

## Methods in IronPython

### Finding Methods

To list all methods available in the `string module`, import the module:

```
import string
```

Then get the directory listing:

```
dir(string)
```

This returns a list of all the methods available (as well as some `__somename__` internal names that can be ignored).

### Help

Once you know a function name, you can get more help on it using the built-in `help` method.

## Related Topics

[Introduction to IronPython](#)

[Translating Script commands from VBScript to IronPython](#)

[Scripting Using Iron Python: Putting it all Together](#)

[IronPython Samples](#)

## Translating Script Commands from VBScript to IronPython

This topic briefly describes scripting methods and arguments via VBScript samples. The distinctions made here are significant and useful when translating scripts written in VBScript to IronPython.

## Related Topics

[Script Method Argument](#)

[VBscript Method Call Types](#)

[Converting VBScript Function calls to IronPython Syntax](#)

[Introduction to IronPython](#)

[IronPython Mini Cookbook](#)

[Scripting Using Iron Python](#)

[IronPython Samples](#)

## Script Method Argument

[Script method calls in VBscript](#) generally take the form:

```
objectName .methodName ( arg1, arg2, ... )
```

The function call syntax is a standard followed by several programming languages. However, the argument types in VBScript objects used for product scripting are restricted to the following:

- Primitive Types
- Named Arrays
- Named Functions

### Primitive Types

Primitive types are the standard `bool`, `int`, `float`, `double`, and `string`.

### Named Arrays

Named arrays are a special construct used very commonly and can be found in many recorded script samples.

A named array begins with **Array("NAME:someName")** followed by a collection of comma separated values which can be:

- Primitive values
- Arrays of primitive values
- Additional named arrays
- Keys, in the form "**keyName:=**" followed by a primitive value or function

### Named Functions

Named functions are arrays which start with **Array(** and *do not* have a leading "NAME:name" item. **They are always introduced by a key** and can contain comma separated values of the following type:

- A primitive value
- A key (of the form "**keyName:=**") followed by
  - A primitive value
  - Another function (nested function)

## Related Topics

[Translating Script commands from VBScript to IronPython](#)

### VBScript Method Call Types

VBScript method calls fall into two categories and the distinction between the two results in syntax differences. These syntax differences are significant when converting VBScript to IronPython.

### VBScript Functions

In VBScript terminology, functions return values. The syntax for this is one shared with practically all programming languages:

```
Set oDesktop = oAnsoftApp.GetAppDesktop()  
Set oProject = oDesktop.NewProject
```

**Note:**

If there are arguments, the method name is *always* followed by an argument list enclosed in parentheses. If the argument list is empty, as shown above for the *NewProject* call, the parentheses can be omitted.

## VBScript Sub-Routines

VBScript subroutines are those that do not have any return value. VBScript allows these to be written without any parentheses even if they have a non-empty argument list.

```
oModule.CreateReport "XY Plot1", "Standard", "XY Plot", "optimtee  
: optimtee", _  
    Array("Domain:=", "Sweep"), Array("Freq:=", Array("All"), "off-  
set:=", _  
    Array("0uin")), Array("X Component:=", "Freq", "Y Component:=", _  
    Array("dB20(S(1,1))", "dB20(S(1,2))", "dB20(S(1,3))", _  
    "dB20(S(2,1))", "dB20(S(2,2))", "dB20(S(2,3))", "dB20(S(3,1))",  
    "dB20(S(3,2))", "dB20(S(3,3)))), Array()
```

## Related Topics

[Translating Script commands from VBScript to IronPython](#)

## Converting VBScript Function calls to IronPython Syntax

When used for scripting, IronPython function names are always followed by parentheses.

So:

- If you see a VBScript snippet that looks like a VBScript subroutine, remember to add parentheses.
- If you see a VBScript function that has no arguments and no parentheses, remember to add them around an empty argument list.

The parentheses change is the only one to keep in mind when converting VBScript function calls syntax to IronPython.

## Return Values

VBscript return values are sometimes assigned via the Set declaration. IronPython return values are simple assignment (See: [Iron Python Mini Cookbook](#)).

## Primitive Method Arguments

Replace each VBScript primitive with an equivalent IronPython primitive.

Boolean values in IronPython have their first letter capitalized (`True` instead of `true` and `False` instead of `false`).

For arrays, the recommended approach is to simply replace a VBScript array with a Python array. The mapping is simple:

- Change `Array(` to a square bracket `[` and close with another square bracket `]` instead of a parenthesis `)`
- Remove the line continuation underscore symbol: `_`
- Map Boolean values correctly

## Named Array Arguments

Formatting helps readability immensely but is not required.

All that *must* be done is:

- Add the parentheses, since the VBScript subroutine omits them
- Replace the `Array( )` delimiters with `[]`
- Remove the `Char(34)` function (which introduced a double quote) and replace it with the escaped double quote literal: `\\"`
- Replace `true` with `True`
- Remove the line continuation symbol: `_`

## Named Array Values with All Key Value Pairs

While it is generally not allowed to replace arrays and nested arrays with Python dictionaries, in the case where the named array consists entirely of key value pairs, you can use a dictionary and avoid typing the trailing `:=` symbols after the keys. This further aids readability of the script.

- If all key value pairs
- Remove the trailing `:=` after each key
- Replace the `,` after the key with a `:`
- If the named array is the top level argument, ensure that the `NAME:name` is present and is split into `NAME : name` as a key value pair
- Enclose the converted array in a `{ }` pair to declare the dictionary.

### Named Arrays with Nested Named Arrays

- Split the NAME:name field into a key value pair
- Translate array key value pair to a dictionary key value pair.
- Create a new key with the name of the nested array and keep the nested array (as an array or as a dictionary) as its value. If the nested array is being retained as an array, the NAME:name field should be retained in the array. If the nested array is being converted to a dictionary, the name is optional: if also retained in the nested array, it must match the outer key.

```
[ "NAME:name",
  "key1": 1,
  "key2": 2,
  ["NAME:name2", "R": 255]
]
```

### Sample Script: Named array with nested named array in array syntax

The above named array with a nested named array (after conversion to IronPython as named array) can be converted to a dictionary as well. The dictionary can take any of the following forms

```
{ "NAME" : "name",
  "key1" : 1,
  "key2" : 2,
  "name2" : ["NAME:name2", "R": 255]
}
```

### Sample Script: Named array with nested named array as mixed dictionary + array

```
{ "NAME" : "name",
  "key1" : 1,
  "key2" : 2,
  "name2" : {"R" : 255}
}
```

### Sample Script: Named array with nested named array in all dictionary syntax

```
{ "NAME" : "name",
  "key1" : 1,
  "key2" : 2,
```

```
"name2": {  
    "NAME": "name2",  
    "R": 255  
}
```

### Function Blocks

Function blocks in VBScript argument syntax are represented as arrays without the "NAME..." field. However, functions are always introduced by a key in a parent structure. Function blocks can therefore never exist as a top-level argument. They are only found as the value pairs inside a named array or inside another function block.

#### Important:

Function blocks and their items cannot be converted to dictionaries even though they might be composed entirely of key value pairs.

The reason for this is the need to maintain the user-entered order. Every item in a function block is expected to be transmitted to the script method in exactly the same order as typed out and this is impossible to achieve when a dictionary is used (as the keys get reordered according to the dictionary's internal tree/key sorting scheme).

When you see a function block, simply replace the Array( ) delimiters with python array delimiters [ ]

## Scripting Using Iron Python

If you have existing VBScript/Javascript scripts use existing scripts them as much as possible by either embedding the test into the IronPython script or invoking them.

### Translating a script in VBScript to IronPython

Read the [chapter on translation](#) and study the samples in that chapter as well as those in the appendix. For python syntax and the differences, the [mini-cookbook chapter](#) will also be useful.

### Writing an IronPython script from scratch

Read through the scripting guide available from the product's help menu and translate the VBScript methods described to IronPython using the information provided in the [chapter on translation](#). Studying the samples in the document will also prove helpful.

For python syntax and the differences, the [mini-cookbook chapter](#) will also be useful.

[IronPython Script Execution Environment](#)

[Scripting using Embedded VBScript or JavaScript](#)

[Scripting with IronPython](#)

[Standalone IronPython and Desktop IronPython](#)

**Related Topics**

[Introduction to IronPython](#)

[IronPython Mini-cookbook](#)

[Translating Script commands from VBScript to IronPython](#)

[Appendix: IronPython Samples](#)

**IronPython Script Execution Environment**

Scripts written in IronPython are executed by desktop in four different ways:

- **Tools > Open Command Window**, to open the **IronPython Command Window**:

---

 IronPython Command Window

```
=====
ElectronicsDesktop 2019.3.0
IronPython 2.7.0.40 on .NET 4.0.30319.42000
-----
- With Tab completion
- dir()      - lists all available methods and objects
- dir(obj)   - lists all available attributes/methods on obj
- help(obj)  - provides available help on a method or object
- tutorial() - provides more help on using the console
-----
try executing "dir(oDesktop)" or dir_sig(oDesktop,"ver")
=====
>>> |
```

---

- **Tools > Run Script** menu item, select "IronPython" from the file type drop-down list.
- Launch the product with a script argument.
  - Maxwell -runscript someScript.py to keep Maxwell GUI open after completing script execution.
  - Maxwell -features=beta -ng -runscriptandexit someScript.py to run Maxwell in a non-graphical mode and exit after script completion. Note that this is a beta feature.
- Register an IronPython script as an external tool using the **Tools > External Tools** menu item.

When desktop executes a script, it does so in an execution environment setup with predefined variables and functions. These predefined variables and functions are how the script

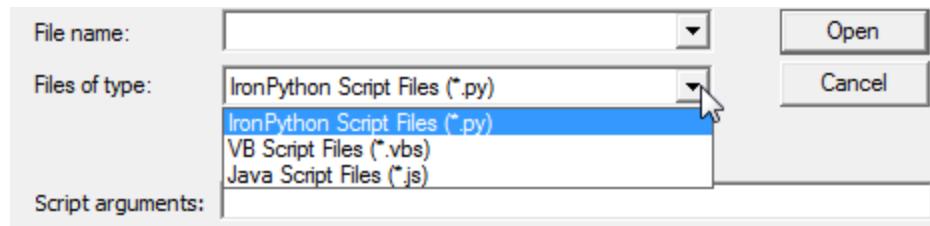
---

communicates with the desktop, and they come in four flavors addressed in the following sub-topics:

### [Script Argument for IronPython](#)

#### Script Argument for IronPython

When scripts are launched using the **Tools > Run Script** menu item, the dialog that pops up allows the user to specify arguments.



**Figure 1: Run Script dialog and script arguments**

Any argument specified here is communicated to the script being executed as the predefined variable **ScriptArgument**.

#### Related Topics

[IronPython Script Execution Environment](#)

#### Scripting using Embedded VBScript or JavaScript

Since script recording is still done in VBScript and users are expected to have a significant collection of VBScript or JavaScript assets, it is useful to continue to use existing script files and snippets even when scripting in IronPython. The various **Run<\*>Command** methods have been designed for this purpose.

For instance: one can create a parameterized cone in HFSS by executing the following IronPython script from the **Tools > Run Script** menu.

```
# assign the VBScript snippet obtained from a script recording from
HFSS to

# coneScript and replace the BottomRadius recorded value with botRa-
dius

coneScript = """Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
```

```
Dim oModule

Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")

Set oDesktop = oAnsoftApp.GetAppDesktop()

oDesktop.RestoreWindow

Set oProject = oDesktop.GetActiveProject()

oProject.InsertDesign "HFSS", "HFSSPyTestDesign", "DrivenModal", ""

Set oDesign = oProject.SetActiveDesign("HFSSPyTestDesign")

Set oEditor = oDesign.SetActiveEditor("3D Modeler")

oEditor.CreateCone Array("NAME:ConeParameters", _
    "XCenter:=", "0mm", "YCenter:=", "0mm", "ZCenter:=", "0mm", _
    "WhichAxis:=", "Z", "Height:=", "2mm", _
    "BottomRadius:=", "3mm", _
    "TopRadius:=", "0mm"), Array("NAME:Attributes", "Name:=", _
    "Cone1", "Flags:=", "", "Color:=", "(132 132 193)", "Trans-
parency:=", 0, _
    "PartCoordinateSystem:=", "Global", "UDMID:=", "", "Mater-
ialValue:=", _
    "" & Chr(34) & "vacuum" & Chr(34) & "", "SolveInside:=", _
    true)

"""

SetScriptingLanguageToVBScript()

RunScriptCommand(coneScript)
```

### Sample Script 11: Hybrid VBScript + IronPython scripting: parameterized Cone Creation

Even though recorded VBScript is used for scripting, the incremental functionality that is provided using IronPython is the ability to write a GUI using IronPython/.NET, collect information from the user and then modify or generate the VBScript commands to actually script the Ansys EM desktop. This GUI functionality is cross platform and a significant positive. The following example demonstrates a contrived use of a .NET window form to display the argument supplied to the IronPython script (via the **ScriptArgument** variable).

```
#import the CLR references

import clr
```

```
clr.AddReference("System.Windows.Forms")

from System.Windows.Forms import Application, Form, Label, Button,
DockStyle

# the GUI form to show some text
# the class below derives from Form (System.Windows.Forms.Form)
# imported above from the .NET assembly.
class ShowPropertiesForm(Form):

    def __init__(self, name, text):
        self.Name = name
        self._label = Label()
        self._label.Text = text
        self._label.Dock = DockStyle.Fill

        _button = Button()
        _button.Text = "Close"
        _button.Dock = DockStyle.Bottom
        _button.Click += self._buttonPressed

        self.Controls.Add(self._label)
        self.Controls.Add(_button)

    def _buttonPressed(self, sender, args):
        self.Close()

#-----
# Main script code
#-----
#display the ScriptArgument variable as the text label
```

```
# in the form.  
  
gui = ShowPropertiesForm("Sample Form", ScriptArgument)  
  
# This makes it a modal dialog.  
  
gui.ShowDialog()  
  
# the following will make it a non-modal dialog  
  
#Application.Run(gui)
```

### **Sample Script 12: Demonstrates the use of a .NET form from IronPython**

While creating cross platform user interfaces from scripts is one of the main motivations driving the adoption of IronPython, any .NET assembly can be used with the caveat that Linux use requires Mono compatibility of any used assemblies.

While this hybrid approach is useful when you have existing VBScript commands that you want to reuse or when you want to quickly parameterize a recorded sample, the one significant limitation of this approach is the inability to capture return values from VBScript or JavaScript calls that do return something. Full two way communication with the product requires the use of pure IronPython to directly invoke the script objects as described below.

### **Related Topics**

[IronPython Script Execution Environment](#)

### **Scripting with IronPython**

While this section talks about directly interacting with the script objects, note that you can execute VBScript or Javascript at any point using any of the available Run\*Command functions. using your existing script assets in this fashion and mixing with IronPython code for new functionality as needed is a viable and option.

Access to the application scripting objects is provided via the predefined **oDesktop** object (as listed in Script Objects). Interacting with the script objects is very natural, method calls are made just like in VBScript except that the argument syntax is somewhat simplified to follow natural Python syntax. All primitive types (string, integer, double) map to the natural primitive types in python. The only differences from the VBScript syntax are seen when specifying array type arguments. The differences are described in earlier chapters.

**Note:**

The typical VBScript calls to obtain the registered COM scripting interface via CreateObject calls and then obtain the oDesktop object from it using the GetAppDesktop() is not needed (or even supported on all platforms). Since all scripting occurs in the context of a running workbench, the available Desktop object is always provided and expected to be used directly.

Scripting using the IronPython scripting API is very much like scripting with VBScript except that

- Any argument is supplied via the built in **ScriptArgument** variable
- The **oDesktop** object is always available
- The scripting method names are identical to the ones used with VBScript
- Method calls, while the name is the same have to adhere to the rule of ensuring trailing parentheses irrespective of whether the function returns anything or has any arguments.
- Any compound/block arguments should be translated to the appropriate IronPython array or dictionary syntax.

The [samples section](#) lists a collection of pure IronPython snippets: these, along with the various script snippets listed in this document should serve as a guide and reference.

## Related Topics

[IronPython Script Execution Environment](#)

[Standalone IronPython and Desktop IronPython](#)

### Standalone IronPython

In general, it is easier to run a script directly from Electronics Desktop. Standalone IronPython does not implement all the functionality available when a script is run from Electronics Desktop. It only implements full support for COM functions.

#### Running Standalone IronPython

Standalone IronPython uses COM to get the handle to the AnsysEDT app. To run standalone IronPython, you'll need to call the IronPython interpreter ipy64.exe.

It is located in:

\\\<AnsysEDTInstallationPath>\common\IronPython\ipy64.exe

For example, to run myScript.py, type the following in the command line:

```
"C:\Program Files\AnsysEM\v232\Win64\common\IronPython\ipy64.exe"
"<filePath>\myScript.py"
```

You can set the interpreter to be the default program when double-clicking the .py script. You can use any recorded script as the basis for a standalone script and simply add an installation-internal path to the python module search path (as shown below) and end the script with a new shutdown call.

### Using a Recorded Script

A python script recorded in AnsysEDT already has the required lines to be run as a standalone, except for the first two lines (path settings) and the final Shutdown () call. See the [example script](#) below.

### Creating an External Script

When creating a script outside of Electronics Desktop, the following lines should be included at the beginning of your script:

- import sys  
  
# Imports the sys module containing system-specific functions native to IronPython.
- sys.path.append("<InstallationPath>")  
  
# Adds the Electronics Desktop installation path to the list of directories Python searches for modules and files.
- sys.path.append("<InstallationPath>/PythonFiles/DesktopPlugin")  
  
# Adds the PythonFiles/DesktopPlugin subfolder to the list of directories Python searches for modules and files.
- import ScriptEnv  
  
# This imports ScriptEnv.py from the installation path specified above. ScriptEnv.py performs an operating system check and defines functions used in Electronics Desktop scripts. See the annotations in the ScriptEnv.py file for more information.
- ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")  
  
**or** ScriptEnv.InitializeNew(NonGraphical=True)  
  
# Initialize and InitializeNew are functions within ScriptEnv.py. The first option launches Electronics Desktop. The second allows you to run a script without launching Electronics Desktop. See the annotations in the ScriptEnv.py file for more information.

You must end the script with:

- `ScriptEnv.Shutdown()`
- # This stops ScriptEnv.py. If you are running multiple scripts, include this only at the end of the last script.

### Example Script

```
import sys  
  
sys.path.append(r"C:\Program Files\AnsysEM\v232\Win64")  
sys.path.append(r"C:\Program Files\An-  
sysEM\v232\Win64\PythonFiles\DesktopPlugin")  
  
  
import ScriptEnv  
  
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")  
oDesktop.RestoreWindow()  
  
oProject = oDesktop.NewProject()  
  
oProject.InsertDesign("HFSS", "HFSSDesign1", "DrivenModal", "")  
oDesign = oProject.SetActiveDesign("HFSSDesign1")  
oEditor = oDesign.SetActiveEditor("3D Modeler")  
oEditor.CreateRectangle(  
[  
    "NAME:RectangleParameters",  
    "IsCovered:= ", True,  
    "XStart:= ", "-0.2mm",  
    "YStart:= ", "-3mm",  
    "ZStart:= ", "0mm",  
    "Width:= ", "0.8mm",  
    "Height:= ", "1.2mm",  
    "WhichAxis:= ", "Z"  
,  
[  
    "NAME:Attributes",  
    "Name:= ", "Rectangle1",
```

```
"Flags:= "", "",  
"Color:= ", "(132 132 193)",  
"Transparency:= ", 0,  
"PartCoordinateSystem:=", "Global",  
"UDMId:= ", "",  
"MaterialValue:= ", "\\"vacuum\\\"",  
"SolveInside:= ", True  
])  
oDesign.SetDesignSettings(['NAME:Design Settings Data', 'Allow Material Override:=' , True, 'Calculate Lossy Dielectrics:=' , True])  
oEditor.SetModelUnits(['NAME:Units Parameter', 'Units:=' , 'mil', 'Rescale:=' , False ])  
ScriptEnv.Shutdown()
```

## IronPython Samples

### Change property

The following snippets show how a change property command (in this case, to change the color of a cone) looks in VBScript and its two possible IronPython variants.

```
oEditor.ChangeProperty Array("NAME:AllTabs", Array("NAME:Geometry3DAttributeTab", _  
    Array("NAME:PropServers", "Cone1"), _  
    Array("NAME:ChangedProps", _  
        Array("NAME:Color", "R:=" , 255, "G:=" , 255, "B:=" , 0))))
```

### Sample Script 13: ChangeProperty command to change color of a cone in VBScript

```
oEditor.ChangeProperty(  
    ["NAME:AllTabs",  
        ["NAME:Geometry3DAttributeTab",  
            ["NAME:PropServers", "Cone1"],  
            ["NAME:ChangedProps",  
                ["NAME:Color", "R:=" , 0, "G:=" , 0, "B:=" , 64]  
            ]  
    ]
```

```
])
```

### Sample Script 14: ChangeProperty command to change color of cone using Python arrays

Any time there are named arrays composed purely of key-value pairs, they can always be represented using a Python dictionary, irrespective of the nesting of said named array.

```
oEditor.ChangeProperty(  
    ["NAME:AllTabs",  
     ["NAME:Geometry3DAttributeTab",  
      ["NAME:PropServers", "Cone1"],  
      ["NAME:ChangedProps",  
       {  
         "NAME": "Color",  
         "R" : 0,  
         "G" : 64,  
         "B" : 0  
       }]]  
)
```

### Sample Script 15: ChangeProperty command to change the color of a cone using Python arrays and dictionaries

#### Create a Cone using IronPython

Most scripting tasks using IronPython are expected to be formatted as the following example. One starts with the predefined **oDesktop** object and drills down to the design, editors, modules etc and issues any required commands on the object while formatting the script command arguments in natural python syntax.

```
oProject = oDesktop.GetActiveProject()  
  
oDesign = oProject.InsertDesign("HFSS", "Random", "DrivenModal", "")  
  
oEditor = oDesign.SetActiveEditor("3D Modeler")  
  
oEditor.CreateCone(  
{  
  "NAME" : "ConeParameters",  
  "XCenter" : "0mm",  
  "YCenter" : "0mm",
```

```
"ZCenter" : "0mm",
"WhichAxis" : "Z",
"Height" : "2mm",
"BottomRadius" : "1.56204993518133mm",
"TopRadius" : "0mm"
},
{
"NAME" : "Attributes",
"Name" : "Cone1",
"Flags" : "",
"Color" : "(132 132 193)",
"Transparency" : 0,
"PartCoordinateSystem": "Global",
"UDMId" : "",
"MaterialValue" : "\\"vacuum\\\"",
"SolveInside" : True
}
)
```

### Sample Script 16: IronPython script to create a cone

#### Create geometry and then create a grid from it using copy/paste/move

The following script demonstrates slightly more advanced use of scripting and the use of return values from script methods. It creates a 5x5 grid of cones and also demonstrates the adding of information messages to the application's message window.

```
oProject = oDesktop.GetActiveProject()

oDesign = oProject.InsertDesign("HFSS", "Hersheys Kisses", "DrivenModal", "")

oEditor = oDesign.SetActiveEditor("3D Modeler")

# create the first cone
AddInfoMessage("Creating first cone")
firstConeName = "firstCone"
```

---

```
coneBotRad = "1.5mm"

oEditor.CreateCone(
{
    "NAME" : "ConeParameters",
    "XCenter" : "0mm",
    "YCenter" : "0mm",
    "ZCenter" : "0mm",
    "WhichAxis" : "Z",
    "Height" : "2mm",
    "BottomRadius": coneBotRad,
    "TopRadius" : "0mm"
},
{
    "NAME" : "Attributes",
    "Name" : firstConeName,
    "Flags" : "",
    "Color" : "(132 132 193)",
    "Transparency" : 0,
    "PartCoordinateSystem": "Global",
    "UDMId" : "",
    "MaterialValue" : "\\"vacuum\\\"",
    "SolveInside" : True
}
)

# Now replicate this a few times and create an array out of it
AddInfoMessage("Replicating it 24 times")
for x in range(5):
    for y in range(5):
        # leave the first one alone in it's created
```

```
# position
    if x == 0 and y == 0:
        continue

# all other grid positions, replicate from the
# first one

# copy first
oEditor.Copy(
{
    "NAME" : "Selections",
    "Selections" : firstConeName
}
)

# paste it and capture the pasted name
# the pasted names come in an array as we could
# be pasting a selection composed of multiple objects
pasteName = oEditor.Paste()[0]

# now move the pasted item to it's final position
oEditor.Move(
{
    "NAME" : "Selections",
    "Selections" : pasteName
},
{
    "NAME" : "TransalateParameters",
    "CoordinateSystemID" : -1,
    "TranslateVectorX" : "%d * 3 * %s" % (x, coneBotRad),
```

```
"TranslateVectorY" : "%d * 3 * %s" % (y, coneBotRad),  
"TranslateVectorZ" : "0mm"  
}  
  
)  
  
# Now fit the display to the created grid  
oEditor.FitAll()
```

### **Sample Script 17: Sample script to create a cone and then use copy/paste/move to replicate it.**

#### **Related Topics**

[Introduction to IronPython](#)

[IronPython Mini-cookbook](#)

[Translating Script commands from VBScript to IronPython](#)

[Scripting Using Iron Python: Putting it all Together](#)

### **Creating User Defined Primitives and User Defined Models in Python Scripts**

You can create User Defined Primitives and User Defined Models in Python scripts (based on the IronPython implementation).

#### **Advantages Compared to C++**

- No need to create and build project; all you need to do is create a Python script
- Python script is platform independent
- Scripts can inherit functionality from existing scripts
- Garbage collector - no need to free memory
- Easy debugging

#### **Changes compared to C**

Though methods, constants and structures are kept as close to the C implementation as possible, some changes had to be made to make code Python-compatible.

#### **Structures**

- Structures have the same names as in C implementation.
- Structures fields names are capitalized.
- Arrays in structures become lists in Python (Technically a.NET IList container)

- Structure instances are created using the supplied constructors and members are accessed using the provided access methods.

For a complete list of structures and examples please see [UDP/UDM Structures](#).

#### Return Values for UDM and UDP Functions

For information on return values for each UDM and UDP function, see the [Return Values](#) section.

#### Constants

Enumeration/Enum constants have almost the same names as in C but the enum must be qualified by the type. Additionally, redundant "UDP", "UDM" or type prefixes have been removed. This allows for better human-readability.

```
# Example of specifying the LengthUnit enum by qualifying it  
# with the type of the enum: UnitType  
unitType = UnitType.LengthUnit
```

For a complete list of enum constants please see [UDP/UDM Constants](#).

#### Methods

Methods are described in [IUDPExtension methods](#), [IUDMExtension methods](#), and [UDMFuncLibrary](#) listed further in this document. A separate chapter includes a [UDP IronPython example of fillet and chamfer](#).

The main differences in functions parameters (from C implementation):

- functions names in UDPFunctionLibrary and UDMFunctionLibrary are capitalized
- arrays become a python list of objects
- void \* callback parameter is dropped from the parameter list
- output parameters (pointer types that are filled during the function call) usually become return values
- 'list size' parameter usually will be omitted as redundant

#### Output Parameters

The rule for the output parameters is as follows:

- If the function has one output parameter variable and no return value, the variable will become function's return value. The same will happen if the return value is a 'success/failure' boolean ('None' will be returned on failure and parameter variable - on success).
- If the function has one output parameter and a return value, the function will return a Python tuple where function return value will be the first one in the tuple.

- If there is more than one out variable, the function will return a Python tuple with all output parameters in the specified order. If function has a return value, it must always be the first in the tuple.

```
# one output parameter; return value is ignored
udmDefinition = udmFunctionLibrary.GetDefinition()

# one output parameter; return value must be preserved. return
# and output values are packed into the return tuple, in order
(lRet, partIdsList) = udpFunctionLibrary.DetachFaces(nPartIds, faceId-
sList)

# Two output parameter; return value must be preserved
# the return tuple is (returnVal, output1, output2)

(bRet, udpPositionLow, udpPositionHigh) = udmFunc-
tionLibrary.GetBoundingBox(partId,exact);
```

#### Comparison with C function:

C	Python
<pre>bool getDefinition(UDMDefinition* udmDefinition, void* callbackData );</pre> <p>where udmDefinition is an output parameter</p>	<pre>udmDefinition = udmFunctionLibrary.GetDefinition()</pre> <p>(Note: callbackData is omitted in py interface)</p>
<pre>long detachFaces( int nFacesAndPartIds, long* faceIds, long* partIds, void* callbackData);</pre> <p>where partIds is an output para- meter</p>	<pre>(bRet, partIds) = udmFunctionLibrary.DetachFaces (nFacesAndPartIds, faceIds)</pre> <p>(Note: callbackData is omitted in py interface)</p>

#### 'List Size' Parameters

The rule for the 'list size' is as follows:

- If function has input 'List' parameter and input 'list size' parameter, 'list size' parameter will be omitted.
- If function has output 'List' parameter and output 'list size' parameter, 'list size' parameter will be omitted.
- If function has output 'List' parameter and input 'list size' parameter, 'list size' parameter won't be omitted as it's needed for memory allocation in the corresponding C++ function from the UDP/UDM function library.

Example:

```
# input list, input list size
lret = udpFunctionLibrary.Unite(objectIds)

# output list, output list size
faceIdList = udmFunctionLibrary.GetAllFaces(PartId)

# output list, input list size
(lret, partIdList) = udpFunctionLibrary.DetachFaces(listSize,
faceIdList)
```

Comparison with C function:

C	Python
bool getAllFaces( long partId, long* numFaces, long** facelds, void* callbackData);  where numFaces and facelds are output parameters and numFaces is the size of facelds.	facelds = udmFunctionLibrary.GetAllFaces(partId)  (ignore numFaces as redundant: folded into facelds, return value is omitted: folded into the facelds is None check callbackData is omitted)
long unite( long numObjects, long* objectIds, void* callbackData);  where numObjects and objectIds are input parameters	lret = udpFunctionLibrary.Unite(objectIds)  (ignore numObjects as redundant: folded into objectIds)

---

C	Python
and numObjects is the size of objectIds.	callbackData is omitted)
long detachFaces( long nSize, long* faceIds, long* partIds, void* callbackData);  where partIds is and output list and nSize is an input parameters and nSize is the size of partIds.	(lret, partIdList) = udpFunctionLibrary.DetachFaces(nSize, faceIds)  (nSize is not ignored, callbackData is omitted)

**Added Parameters**

There is a special case in UDPFunctionLibrary: two functions - DuplicateAlongLine and DuplicateAroundAxis - have new integer listSize parameter added to their signatures.

This parameter defines the size of the output List. This is done for compliance with C++ geometry library as the size of the List must be predefined and this size is different from the existing parameter's values.

Example:

```
(ret, cloneIDs) = funcLib.DuplicateAlongLine(partID, transVec,  
numCubes, cloneIDsSize)

(ret, cloneIDs) = funcLib.DuplicateAroundAxis(partID, axis, angle,  
nClones, cloneIDsSize)
```

Here cloneIDsSize is a new integer parameter.

Comparison with C function:

C	Python
long duplicateAlongLine( long partId, UDPVector transVector, int nClones, long* nClones, void* callbackData);	(lret, cloneIds) = udmFunctionLibrary.DuplicateAlongLine(partId, transVec, nClones, cloneIdsSize)  (callbackData is omitted cloneIdsSize is a new parameter)
long duplicateAroundAxis( long partId, UDPCoordinateSystemAxis axis,	(lret, cloneIds) = udmFunctionLibrary.DuplicateAroundAxis (partId, axis, angle, nClones, cloneIdsSize)

C	Python
double angle, int nClones, long* nClones, void* callbackData);	(callbackData is omitted clonelidsSize is a new parameter)

## Developing a UDM/UDP

### Creation

To create a User Defined Primitive in Python you write a Python script that implements [UDPExtension class](#). To create a User Defined Model in Python you write a Python script that implements [UDMExtension](#) class (see links for full description).

### Location

The scripts are located the same way the C based UDM/UDP are. They are expected to be under the UserDefinedParts or UserDefinedModels sub-directories of one of the library folders (SysLib, UserLib or PersonalLib). They will then appear under the appropriate menu items:  
**Draw > User Defined Primitives for UDP** or **Draw > User Defined Model for UDM**.

The sub-directories structure created in one of the specified directory will be displayed in the UDP/UDM menu.

Keep in mind that there is no difference between the menu display for C and Python implementations of UDM or UDP - only the file names without extensions are displayed

### Organize

"Lib" sub-directory is a special directory. The contents of this directory is not shown in the menu. In the "Lib" directory you can create Python scripts with base classes and utilities to be used in UDP/UDM Python scripts. All the Lib directories upstream of a script (till the UserDefinedModels or UserDefinedPrimitives) are included in the Python search path and this allows for easy import of helper modules in such directories.

To use UDM data structures, constants, and/or classes in your Lib sub-directory scripts you have to add import statement to the scripts:

For UDM:extension:

```
from UDM import *
```

For UDP:extension:

```
from UDP import *
```

## Edit/Reload

Python is a scripting language, so if you have errors in your script, you will see them at the time you try to run the script. The errors will be displayed in the Message Manager Window. If you need more information, you might be able to get it from log files. See: Debug Logging.

You can always change your script, call **Update Menu** command from **Draw > User Defined Model > menu** or **Draw > User Defined Primitives > menu** and run the script again. If you delete script you might want to restart the application instead of calling **Update Menu**.

## UDPExtension

### Import

You do not have to add import statements for the predefined classes, structures, and constants - it is done for you and all data types described in this document can be used in your Python script.

However you have to add import statements to your helper scripts in your Lib sub-directory.

```
from UDP import *
```

### Main class: UDPExtension

You must write a class derived from IUDPExtension with a mandatory name UDPExtension:

```
class UDPExtension(IUDPExtension):
```

The class should implement [IUDPExtension methods](#) described below.

### IUDPExtension methods

All methods are same as the methods in the C UDP implementation. The changes to the methods signatures are just to conform to the Python style.

#### Mandatory methods.

These methods must be implemented in the UDP Python script as methods of UDPExtension class.

##### GetLengthParameterUnits()

- returns string.

##### GetPrimitiveTypeInfo()

- returns UDPPrimitiveTypeInfo.

##### GetPrimitiveParametersDefinition2()

- returns a list of UDPPrimitiveParameterDefinition2 or None on failure

**AreParameterValuesValid2(errorMsg, udpParams)**

- errorMsg is a c# list of strings
- udpParams is a c# list of UDPPParam
- returns True if udpParams are valid, False otherwise.

**CreatePrimitive2(funcLib, udpParams)**

- funcLib is [UDMFunction library](#)
- udpParams is a c# list of UDPPParam
- returns True on success, False on failure.

**Optional methods**

These methods, which have default implementations, can be implemented as methods of UDPExtension class as needed. Default methods will return NULL or FALSE depending on the return type.

**GetPrimitiveParameters()**

- returns Python list of strings or NULL

**GetRegisteredFaceNames()**

- returns Python list of strings or NULL

**GetRegisteredEdgeNames()**

- returns Python list of strings or NULL

**GetRegisteredVertexNames()**

- returns Python list of strings or NULL

**ProjectParametersOnToValidPlane2(currentUDPPParams, projectedUDPPParams)**

- currentUDPPParams is a list of UDPPParam
- projectedUDPPParams is a list of UDPPParam
- returns True on success, False on failure.

**MapParametersDefinitionVersions2(oldVersion, oldUDPPParams)**

- oldVersion is a string
- oldUDPPParams is a list of UDPPParam
- returns Python list of UDPPParam or NULL

**GetOldPrimitiveParametersDefinition2(version )**

- version is a string
- returns a list of UDPPrimitiveParameterDefinition2 or None on failure.

**Example UDP**

```
import sys

class UDPExtension(IUDPExtension):


    def GetLengthParameterUnits(self):
        return "mm"

    def GetPrimitiveTypeInfo(self):
        typeInfo = UDPPrimitiveTypeInfo(
            name = "SampleUDP",
            purpose = "example",
            company="Ansys",
            date="12.21.12",
            version = "1.0")

        return typeInfo

    ...
    ...
```

**UDMExtension****Import**

You do not have to add import statements for the predefined classes and structures - it is done for you, and all data types described in this document can be used in your Python script.

However you have to add import statements to your helper scripts in your Lib sun-directory.

---

```
from UDM import *
```

**Main class: UDMExtension**

You must write a class derived from IUDMExtension with a mandatory name UDMExtension:

```
class UDMExtension(IUDMExtension):
```

The class should implement [IUDMExtension methods](#) described below.

**IUDMExtension methods**

All methods are the same as the methods in the C UDM implementation. The changes to the methods signatures are just to conform to the Python style.

**Mandatory methods.**

These methods must be implemented in the UDM Python script as methods of UDMExtension class.

**GetInfo()**

- returns UDMInfo object populated with appropriate UDM information.

**IsAttachedToExternalEditor()**

- returns True if UDM dll is attached to external editor.
- In case of python UDMs, this should typically return False

**CreateInstance(funcLib)**

- funcLib is UDMFunctionLibrary
- returns UDMPARAMETERS.

**GetUnits(instanceId)**

- instanceId is a long
- returns string containing units for the instance.

**Refresh(funcLib, udmlnParams, updatedParams, refreshModifiedPartsOnly, nonEditedPartRefIds )**

This Method is called every time a UDM is refreshed. Geometry creation/refresh should happen in this method.

- funcLib is UDMFunctionLibrary
- udmlnParams is a list of UDMPARAMETERS that comes from desktop

- `updatedParams`: UDM script can change the UDM parameters it receives. Updated parameters need to be sent back to desktop. If the UDM script is not going to change any of the parameters that it received, it needs to copy `udmInParams` to `updatedParams`.
- `refreshModifiedPartsOnly` is a boolean

Supporting this flag is optional. For UDMs where the refresh performance is not an issue, it is recommended to ignore this flag and update all parts every time.

This flag can be used to optimize performance of Refresh method when the model created by UDM is large. If the UDM consists of multiple parts, and new parameters change only a few parts amongst them, UDM script can only modify parts that are changed by the new parameters.

- `nonEditedPartRefIds`: If `RefreshModifiedPartsOnly` is true and the UDM script supports partial update, Refresh method needs to return ids of parts that are unchanged.

returns True on success, False on failure.

#### **ReleaseInstance(instancId)**

- `instancId` is a long
- This should release any resources assigned to this particular instance of UDM.
- returns True on success, False on failure.

#### **GetAttribNameForEntityId()**

- Returns string that acts as a the name of the attribute containing entity IDs.
- For example, it can return a unique string such as "ATTRIB\_XACIS\_ID"
- Python UDMs should implement this method.

#### **GetAttribNameForPartId()**

- Returns string that acts as a the name of the attribute containing entity IDs.
- For example, it can return a unique string such as "ATTRIB\_XACIS\_ID" (Can be same as `GetAttribNameForEntityId()`)
- Python UDMs should implement this method.

#### **Optional methods**

These methods have default implementations (default is to return NULL or FALSE depending on the return type) but can be overridden by the user as needed as methods of `UDMExtension` class.

#### **DialogForDefinitionOptionsAndParams(self, defData, optData, params):**

---

Replaces the old UDMDialogForDefinitionAndOptions method, which is still supported, but users are urged to use UDMDialogForDefinitionOptionsAndParams. If both methods are present, application will use UDMDialogForDefinitionOptionsAndParams.

- UDM can pop up dialog for UDM definition, options, parameters in this method. Definition, options, and parameters are set/modified by user and returned to application. Dll can also just give default definition, options and parameters.
- Returns two booleans and a string
- First boolean returns whether the method was successful or not.
- Second boolean returns whether the application should popup a dialog box. If it is True, application will populate a dialog with definition, options, parameters that are returned.
- String returned contains length units for parameters.

#### **DialogForDefinitionAndOptions(self, defData, optData) [Deprecated]**

UDM can pop up dialog for UDM definition and options in this method. Definition, and options are set/modified by user and returned to application. Dll can also just give default definition and options.

- Returns two booleans.
- First boolean provides whether the call to this method was successful or not.
- Second boolean determines whether the application should pop up a dialog box. If this is true, application will populate the dialog with the definitions and options that are returned. As no parameters are returned, no parameters are shown in this dialog.

#### **GetInstanceSourceInfo(instanceld)**

- instanceld is a long
- returns string containing source information of UDM instance. It is used to create initial name for UDM instance.

#### **ShouldAttachDefinitionFilesToProject()**

- returns true if any of definition files needs to be attached to project
- returns python list of string containing definition names of files or NULL

#### **Example UDM**

```
class UDMExtension(IUDMExtension):  
  
    def IsAttachedToExternalEditor(self):  
        return False
```

```
def GetInfo(self):
    udmInfo = UDMInfo(
        name = "SampleUDM",
        purpose = "udm example",
        company="Ansys",
        date="12.21.12",
        version = "1.0")

    return udmInfo
...
...
```

## UDMFunctionLibrary

UDMFunctionLibrary implements IUDMFunctionLib interface. The IUDMFunctionLib object is passed as a parameter to Python script in the following functions

- CreateInstance
- Refresh

You can call any of the functions from the functions list (shown below).

```
partRefId = udmFunctionLib.GetPartRefId(partId)
```

For example sample code that calls GetBoundingBox in Python script can look like this:

```
partId = 10
exact = True
udpPosition = UDPPosition(0,0,0)

(bret, udpPositionLow, udpPositionHigh) = udmFunctionLibrary.GetBoundingBox(partId, exact);

if bret:
    udpPosition.X = udpPositionLow.X
```

As you can see udpPositionLow and udpPositionHigh output parameters are defined in the call to GetBoundingBox function. There is no need to define them before the function call.

**Functions list:**

1. ***List\_of\_UDMDefinition***: udmDefinitionList = **GetDefinition()**
2. ***List\_of\_UDMOption***: udmOptionList = **GetOptions()**
3. ***bool***: bret = **SetMaterialName(*string*: matName, *int*: partId)**
4. ***bool***: bret = **SetMaterialName2(*string*: matName, *string*: partName)**
5. ***bool***: bret = **SetPartName(*string*: partName, *int*: partId)**
6. ***int***: iret = **GetInstanceId()**
7. ***string***: str = **GetPartRefId(*int*: partId)**
8. ***bool***: bret = **SetPartRefId(*int*: partId, *string*: refId)**
9. ***List\_of\_int***: facelds = **GetAllFaces(*int*: partId)**
10. ***List\_of\_int***: edgelds = **GetAllEdges(*int*: partId)**
11. ***List\_of\_int***: vertexIds = **GetAllVertices(*int*: partId)**
12. ***bool***: bret = **SetFaceAttrbs(*List\_of\_int*: facelds, *List\_of\_string*: attrbs)**
13. ***bool***: bret = **SetEdgeAttrbs(*List\_of\_int*: edgelds, *List\_of\_string*: attrbs)**
14. ***bool***: bret = **SetVertexAttrbs(*List\_of\_int*: vertexIds, *List\_of\_string*: attrbs)**
15. ***string***: str = **GetModelerUnit()**
16. ***string***: str = **GetCacheFileForUDMResume()**
17. ***bool***: bret = **SetPartColor(*int*: partId, *int*: nColor)**
18. ***bool***: bret = **SetPartFlags(*int*: partId, *int*: nFlags)**
19. (***bool***: bret, ***UDPPosition***: low, ***UDPPosition***: high) = **GetBoundingBox(*int*: partId, *bool*: exact)**
20. ***bool***: bret = **IsParametricUpdate()**
21. ***bool***: bret = **SetMaterialNameByRefId(*string*: partRefId, *string*: matName)**
22. ***bool***: bret = **SetPartNameByRefId(*string*: partRefId, *string*: partName)**
23. ***bool***: bret = **SetPartColorByRefId(*string*: partRefId, *int*: nColor)**
24. ***bool***: bret = **SetPartFlagsByRefId(*string*: partRefId, *int*: nFlags)**

In addition to the above functions all functions defined in the UDPFunctionLib are available in the IUDMFunctionLib and can be called directly exactly the same way as the IUDMFunctionLib functions.

**Example:**

```
udmFunctionLib.CreateCircle(center, radius, ratio, isCovered)
```

## UDM/UDP Functions

Return Values for Each UDM and UDP Function

*ID* – *ID of created Object*

*SI* – *Success Indicator. Identifies whether or not operation was successful.*

**Functions list:**

1. **bool: SI = AddMessage(*MessageSeverity*: messageSeverity, *string*: message)**
2. **bool: SI = NameAFace(*UDPPosition*: pointOnFace, *string*: faceName)**
3. **bool: SI = NameAEdge(*UDPPosition*: pointOnEdge, *string*: edgeName)**
4. **bool: SI = NameAVertex(*UDPPosition*: pointOnVertex, *string*: vertexName)**
5. **int: ID = GetFaceIDFromPosition(*UDPPosition*: pointOnFace)**
6. **int: ID = GetEdgeIDFromPosition(*UDPPosition*: pointOnEdge)**
7. **int: ID = CreatePolyline(*UDPPolylineDefinition*: polylineDefinition)**
8. **int: ID = CreateRectangle(*CoordinateSystemPlane*: whichPlane, *UDPPosition*: centerPoint, *List\_of\_double*: widthAndHeight, *int*: isCovered)**
9. **int: ID = CreateArc(*CoordinateSystemPlane*: whichPlane, *UDPPosition*: centerPoint, *UDPPosition*: startPoint, *double*: fAngle)**
10. **int: ID = CreateCircle(*CoordinateSystemPlane*: whichPlane, *UDPPosition*: centerPoint, *double*: fRadius, *int*: isCovered)**
11. **int: ID = CreateEllipse(*CoordinateSystemPlane*: whichPlane, *UDPPosition*: centerPoint, *double*: fMajorRadius, *double*: fRadiusRatio, *int*: isCovered)**
12. **int: ID = CreateRegularPolygon(*CoordinateSystemPlane*: whichPlane, *UDPPosition*: centerPoint, *UDPPosition*: startPoint, *int*: numOfSides, *int*: isCovered)**
13. **int: ID = CreateEquationBasedCurve(*UDPEquationBasedCurveDefinition*: curveDefinition)**
14. **int: ID = CreateEquationBasedSurface(*UDPEquationBasedSurfaceDefinition*: surfaceDefinition)**
15. **int: ID = CreateSpiral(*UDPSpiralDefinition*: spiralDefinition)**
16. **int: ID = CreateBox(*UDPPosition*: startPoint, *List\_of\_double*: boxXYZsize)**
17. **int: ID = CreateSphere(*UDPPosition*: centerPoint, *double*: fRadius)**
18. **int: ID = CreateCylinder(*CoordinateSystemAxis*: whichAxis, *UDPPosition*: centerPoint, *double*: fRadius, *double*: fHeight)**
19. **int: ID = CreateCone(*CoordinateSystemAxis*: whichAxis, *UDPPosition*: centerPoint, *double*: fBottomRadius, *double*: fTopRadius, *double*: fHeight)**

20. **int**: ID = **CreateTorus**(**CoordinateSystemAxis**: whichAxis, **UDPPosition**: centerPoint, **double**: fMajorRadius, **double**: fMinorRadius)
  21. **int**: ID = **CreatePolyhedron**(**CoordinateSystemAxis**: whichAxis, **UDPPosition**: centerPoint, **UDPPosition**: startPosition, **int**: numOfSides, **double**: fHeight)
  22. **int**: ID = **CreateHelix**(**UDPHelixDefinition**: helixDefinition)
  23. **bool**: SI = **Unite**(**List\_of\_int**: pObjectIDArray)
  24. **bool**: SI = **Subtract**(**List\_of\_int**: pBlankObjectIDArray, **List\_of\_int**: pToolObjectIDArray)
  25. **bool**: SI = **Intersect**(**List\_of\_int**: pObjectIDArray)
  26. **bool**: SI = **Imprint**(**List\_of\_int**: pBlankObjectIDArray, **List\_of\_int**: pToolObjectIDArray)
  27. **bool**: SI = **SweepAlongVector**(**int**: profileID, **UDPVector**: sweepVector, **UDPSweepOptions**: sweepOptions)
  28. **bool**: SI = **SweepAroundAxis**(**int**: profileID, **CoordinateSystemAxis**: whichAxis, **double**: sweepAngle, **UDPSweepOptions**: sweepOptions)
  29. **bool**: SI = **SweepAlongPath**(**int**: profileID, **int**: pathID, **UDPSweepOptions**: sweepOptions)
  30. **bool**: SI = **Translate**(**int**: partID, **UDPVector**: translateVector)
  31. **bool**: SI = **Rotate**(**int**: partID, **CoordinateSystemAxis**: whichAxis, **double**: rotateAngle)
  32. **bool**: SI = **Mirror**(**int**: partID, **UDPPosition**: mirrorPlaneBasePosition, **UDPVector**: mirrorPlaneNormalVector)
  33. **bool**: SI = **Transform**(**int**: partID, **List\_of\_double**: rotationMatrix, **UDPVector**: translateVector)
  34. **bool**: SI = **Scale**(**int**: partID, **double**: xScale, **double**: yScale, **double**: zScale)
  35. (**bool**: SI, **List\_of\_int**: cloneIDs) = **DuplicateAlongLine**(**int**: partID, **UDPVector**: translateVector, **int**: numTotalObjs, **int**: cloneIDsListSize)
  36. (**bool**: SI, **List\_of\_int**: cloneIDs) = **DuplicateAroundAxis**(**int**: partID, **CoordinateSystemAxis**: whichAxis, **double**: rotateAngle, **int**: numTotalObjs, **int**: cloneIDsListSize)
  37. **int**: ID = **DuplicateAndMirror**(**int**: partID, **UDPPosition**: mirrorPlaneBasePosition, **UDPVector**: mirrorPlaneNormalVector)
  38. **bool**: SI = **Connect**(**List\_of\_int**: objectIDArray)
  39. **bool**: SI = **Offset**(**int**: partID, **double**: offsetDistance)
  40. **int**: ID? = **Section**(**int**: partID, **CoordinateSystemPlane**: sectionPlane)
  41. (**bool**: SI, **int**: ID) = **Split**(**int**: partID, **CoordinateSystemPlane**: splitPlane, **SplitWhichSideToKeep**: whichSideToKeep, **bool**: bSplitCrossingObjectsOnly)
-

42. (**bool**: SI, **List\_of\_int**: importedObjectIDs) = **ImportNativeBody2**(**string**: fileNameWithFullPath)
  43. (**bool**: SI, **List\_of\_int**: importedObjectIDs) = **ImportAnsoftGeometry**(**string**: fileNameWithFullPath, **List\_of\_string**: overridingParamsNameArray, **List\_of\_UDPPParam**: overridingParamsArray)
  44. **int**: ID = **Clone**(**int**: partID)
  45. **bool**: SI = **DeletePart**(**int**: partID)
  46. **int**: ID = **CreateObjectFromFace**(**int**: faceID)
  47. **bool**: SI = **Fillet**(**UDPBLNDElements**: entitiesToFillet, **UDPBLNDFilletOptions**: filletOptions)
  48. **bool**: SI = **Chamfer**(**UDPBLNDElements**: entitiesToChamfer, **UDPBLNDChamferOptions**: chamferOptions)
  49. (**bool**: SI, **List\_of\_int**: newPartIDs) = **DetachFaces**(**int**: newPartIDArraySize, **List\_of\_int**: faceIDs)
  50. (**bool**: SI, **List\_of\_int**: newPartIDs) = **DetachEdges**(**int**: newPartIDArraySize, **List\_of\_int**: edgeIDs)
  51. **int**: ID = **CreateObjectFromEdge**(**int**: edgeID)
  52. **bool**: SI = **SheetThicken**(**int**: partID, **double**: fThickness, **bool**: bThickenBothSides)
  53. (**bool**: SI, **List\_of\_int**: newPartIDArray) = **SweepFaceAlongNormal**(**int**: newPartIDArraySize, **List\_of\_int**: faceIDArray, **double**: sweepLength)
  54. **bool**: SI = **CoverLine**(**int**: partID)
  55. **bool**: SI = **CoverSurface**(**int**: partID)
  56. **bool**: SI = **UncoverFaces**(**List\_of\_int**: faceIDArray)
  57. (**bool**: SI, **int**: numPartsCreated, **List\_of\_int**: faceIDArray) = **SeparateBodies**(**int**: partID, **int**: numPartsCreated)
  58. **bool**: SI = **MoveFaces**(**List\_of\_int**: faceIDArray, **bool**: bMoveAlongNormal, **double**: fOffsetDistance, **UDPVector**: moveVector)
  59. **bool**: SI = **WrapSheet**(**int**: sheetBodyID, **int**: targetBodyID)
  60. **bool**: SI = **ImprintProjection**(**int**: blankBodyID, **List\_of\_int**: toolBodyIDArray, **bool**: bNormalProjection, **UDPVector**: projectDirection, **double**: projectDistance)
  61. **string**: path = **GetTempDirPath**()
  62. **string**: path = **GetSysLibDirPath**()
  63. **string**: path =  **GetUserLibDirPath**()
  64. **string**: path = **GetPersonalLibDirPath**()
-

65. **string**: path = **GetInstallDirPath()**
66. **string**: path = **GetProjectPath()**
67. (**bool**: SI, **bool**: abort) = **SetProgress(UDPProgress**: progress)

## UDP/UDM Structures and Constants

The following sections describe:

- [UDP/UDM Structures](#)
- [UDP/UDM Constants](#)

### UDP/UDM Structures

Differences compared to C API

- **UDMDefinition**
- **UDMOptions**
- **UDMParameters**

Instead of containing arrays of data, the structures contain single fields where each field corresponds to an item in a different array from the original C API. The structure objects thus constructed are added to the Python list. Alternately the Python list can be initialized using the structure objects.

Example (creating UDMParameter list):

```
udmParamList = [  
    UDMParameter(  
        "cubeSizeName", UnitType.LengthUnit,  
        UDPParam(ParamDataType.Double, cubeSize),  
        ParamPropType.Value,  
        ParamPropFlag.MustBeReal),  
  
    UDMParameter(  
        "cubeDistanceName", UnitType.LengthUnit,  
        UDPParam(ParamDataType.Double, cubeDistance),  
        ParamPropType.Value,  
        ParamPropFlag.MustBeReal),
```

```
UDMParameter("numCubesName", UnitType.LengthUnit,  
             UDPPParam(ParamDataType.Int, numCubes),  
             ParamPropType.Number,  
             ParamPropFlag.MustBeInt]
```

- **UDPParam**
- **UDPParamData**

**Data** field in UDPPParam is now an object - the same for all types of data used - as Python can work with any type of data.

**UDPParamData** is obsolete, thus not implemented. Be sure to set proper data type to UDPPParam.DataType when setting UDPPParam.Data.

Example:

```
nCubesParam = UDPPParam(ParamDataType.Int, numCubes)  
nCubes = nCubesParam.Data
```

```
distanceParam = UDPPParam()  
distanceParam.setDouble(10.5)  
doubleDistance = distanceParam.Data * 2
```

- **UDP3x3Matrix**

The structure is not implemented. Use size 9 Python List of doubles instead.

Example:

```
rotationMatrix =[0,0,1, 1,0,0, 0,0,1]  
  
udpFunctionLib.Transform(partId, rotationMatrix, trans-  
lationVector)
```

### List of structures

You can use constructors to create a structure. You can also modify fields - directly or by provided methods.

Example:

```
pos1 = UDPPosition(1,2,3)  
pos2 = UDPPosition(x=1,y=10,z=0)  
pos2.Z = pos1.Z  
udpParam = UDPParam(ParamDataType.Double,1)  
value = udpParam.Data
```

Structure	Construction	Members
UDPPrimitiveTypeInfo	UDPPrimitiveTypeInfo( string name, string purpose, string company, string date, string version)	string Name string Purpose string Company string Date string Version
UDPPrimitiveParameterDefinition	UDPPrimitiveParameterDefinition( string name, string description, UnitType unitType, double defaultValue)	string Name string Description UnitType UnitType double DefaultValue
UDPParam	UDPParam()  UDPParam( ParamDataType dataType, object data)  object can be int, double , string, bool or UDPPosition  <b>methods:</b> setInt(int val) setBool(bool val) setString(string val) setDouble(double val)	ParamDataType DataType object Data object can be int, double , string, bool or UDPPosition

Structure	Construction	Members
	setPosition(UDPPosition val)	
UDPPrimitiveParameterDefinition2	UDPPrimitiveParameterDefinition2( string name, string description, UnitType unitType, ParamPropType propType, ParamPropFlag propFlag, UDPParam defaultValue)	string Name string Description UnitType UnitType ParamPropType PropType ParamPropFlag PropFlag UDPParam DefaultValue
UDPPosition	UDPPosition( double x, double y, double z)	double X double Y double Z
UDPVector	UDPVector( double x, double y, double z)	double X double Y double Z
UDPSweepOptions	UDPSweepOptions( SweepDraftType draftType, double draftAngle, double twistAngle)	SweepDraftType DraftType double DraftAngle double TwistAngle
UDPPolylineSegmentDefinition	UDPPolylineSegmentDefinition( PolylineSegmentType segmentType, int segmentstartIndex, int numberOfPoints, double angle, UDPPosition centerPoint, CoordinateSystemPlane arcPlane)	PolylineSegmentType SegmentType int segmentstartIndex, int numberOfPoints, double angle, UDPPosition centerPoint, CoordinateSystemPlane arcPlane)

Structure	Construction	Members
	CoordinateSystemPlane arcPlane)	
UDPPolylineDefinition	UDPPolylineDefinition() UDPPolylineDefinition( List_of_UDPPosition positions, List_of_UDPPolylineSegmentDefinition segDefs, int closed, int covered)	int IsClosed int IsCovered List_of_UDPPosition ArrayOfPosition List_of_UDPPolylineSegmentDefinition ArrayOfSegmentDefinition
UDPEquationBasedCurveDefinition	UDPEquationBasedCurveDefinition( string functionXt, string functionYt, string functionZt, double tStart, double tEnd, int numPointsOnCurve)	string FunctionXt string FunctionYt string FunctionZt double TStart double TEnd int NumOfPointsOnCurve
UDPEquationBasedSurfaceDefinition	UDPEquationBasedSurfaceDefinition( string functionXuv, string functionYuv, string functionZuv, double uStart, double uEnd, double vStart, double vEnd int reserved1 int reserved2)	string FunctionXuv string FunctionYuv string FunctionZuv double UStart double UEnd double VStart double VEnd two integer arguments that are reserved for future use. They need to be provided, for example as 0. For example:  theSurfaceDefinition = UDPEquationBasedSurfaceDefinition ("u", "v", "1", 0, 1, 0, 1, 0, 0)

Structure	Construction	Members
UDPHelixDefinition	UDPHelixDefinition( int profileID, UDPPosition ptOnAxis, UDPPosition axisDir, double noOfTurns, bool isRightHanded, double radiusChangePerTurn, double pitch)	int ProfileID UDPPosition PtOnAxis UDPPosition AxisDir double NoOfTurns bool IsRightHanded double RadiusChangePerTurn double Pitch
UDPSpiralDefinition	UDPSpiralDefinition( int profileID, UDPPosition ptOnAxis, UDPPosition axisDir, double noOfTurns, bool isRightHanded, double radiusChangePerTurn)	int ProfileID UDPPosition PtOnAxis UDPPosition AxisDir double NoOfTurns bool IsRightHanded double RadiusChangePerTurn
UDPBLNDElements	UDPBLNDElements( int partID, int noOfEdges; int* listOfEdges; )  UDPBLNDElements( int partID, int noOfVertices; int* listOfVertices; )	UDPBLNDElements can hold either edges or vertices, but not both at the same time. Edges should be applied to solids, and vertices should be applied to sheets.  int PartID /* part to be blended i.e. filleted/chamfered */ int noOfEdges; int* listOfEdges; /* edges to be blended */ int noOfVertices; int* listOfVertices; /* vertices to be blended */
UDPBLNDFilletOptions	UDPBLNDFilletOptions()	bool SupressFillet /* Reserved for future */

Structure	Construction	Members
	bool supressFillet, BLNDFilletRadiusLaw filletRadiusLaw, double filletStartRadius, double filletEndRadius, bool followSmoothEdgeSequence, BLNDFilletType filletType, double setbackDistance, double bulgeFactor)	BLNDFilletRadiusLaw FilletRadiusLaw double FilletStartRadius double FilletEndRadius bool FollowSmoothEdgeSequence /* Reserved for future */ BLNDFilletType FilletType double SetbackDistance double BulgeFactor /* Reserved for future */
UDPBLNDChamferOptions	UDPBLNDChamferOptions( bool supressChamfer, BLNDChamferRangeLaw chamferRangeLaw, double chamferLeftRange, double chamferRightRange)	bool SupressChamfer BLNDChamferRangeLaw ChamferRangeLaw double ChamferLeftRange double ChamferRightRange
UDPProgress	UDPProgress( int prog, int subProg, string mesg, string subMesg)	int Prog int SubProg string Mesg string SubMesg
UDMInfo	UDMInfo( string name, string purpose, string company, string date, string version)	string Name string Purpose string Company string Date string Version
UDMDefinition	UDMDefinition() UDMDefinition( string name,	string DefName UDPParam DefValue ParamPropType PropType ParamPropFlag PropFlag

Structure	Construction	Members
	UDParam value, ParamPropType propType, ParamPropFlag propFlag)	
UDMOption	UDMOption()  UDMOption( string name, UDParam value, ParamPropType propType, ParamPropFlag propFlag)	string OptName  UDPParam OptValue  ParamPropType PropType ParamPropFlag PropFlag
UDMParameter	UDMParameter()  UDMParameter( string name, UDParam value, UnitType unitType, ParamPropType propType, ParamPropFlag propFlag)	string ParamName  UDPParam ParamValue  UnitType UnitType ParamPropType PropType ParamPropFlag PropFlag

**UDP/UDM Constants**

Full names of enum constants must be used in scripts.

**Example:**

```
unitType = UnitType.LengthUnit
dataType = ParamDataType.Int
```

**Enum constants:**

enum Constant	Parameters
UnitType	NoUnit LengthUnit AngleUnit
ParamDataType	Int

enum Constant	Parameters
	Double String Bool Position Unknown
ParamPropType	Text Menu Number Value FileName Checkbox Position Unknown
ParamPropFlag	NoFlag ReadOnly MustBeInt MustBeReal Hidden Unknown
CoordinateSystemAxis	XAxis YAxis ZAxis
CoordinateSystemPlane	XYPlane YZPlane ZXPlane
SweepDraftType	ExtendedDraft RoundDraft NaturalDraft MixedDraft
SplitWhichSideToKeep	SplitKeepBoth

---

enum Constant	Parameters
	SplitKeepPositiveOnly SplitKeepNegativeOnly
PolylineSegmentType	LineSegment ArcSegment SplineSegment AngularArcSegment
MessageSeverity	WarningMessage ErrorMessage InfoMessage IncompleteMessage FatalMessage
BLNDFilletRadiusLaw	BLNDConstantRadius BLNDVariableRadius
BLNDFilletType	BLNDRound /* The outward surface of the fillet is curved.*/ BLNDMitered /* The outward surface of the fillet is flat and cut at an angle.*/
BLNDChamferRangeLaw	BLNDConstantRange BLNDVariableRange
PartPropertyFlags	PropNonModel PropDisplayWireFrame PropReadOnly PostprocessingGeometry PropInvisible PropShowDirection PropDummy

### UDP Python Example

This Python script example demonstrates how to use the UDPBLNDElements structure and the UDP chamfer and fillet functions.

```
import sys

primitive_info = UDPPrimitiveTypeInfo(
    name="Fillet_Chamfer",
```

```
purpose="Fillet Chamfer Example",
company="Ansys",
date="09/11/2020",
version="1.0")

primitive_param_definitions = [
    UDPPrimitiveParameterDefinition2(
        "x_size",
        "",
        UnitType.LengthUnit,
        ParamPropType.Value,
        ParamPropFlag.MustBeReal,
        UDPParam(ParamDataType.Double, 10)),
    UDPPrimitiveParameterDefinition2(
        "y_size",
        "",
        UnitType.LengthUnit,
        ParamPropType.Value,
        ParamPropFlag.MustBeReal,
        UDPParam(ParamDataType.Double, 5)),
    UDPPrimitiveParameterDefinition2(
        "z_size",
        "",
        UnitType.LengthUnit,
        ParamPropType.Value,
        ParamPropFlag.MustBeReal,
        UDPParam(ParamDataType.Double, 2))
]

length_units = "mm"

#####
```

```
# Class Implementation
#####
class UDPExtension(IUDPExtension):
    def CreatePrimitive2(self, func_lib, param_values):
        """
            Inbuilt function that is called to generate a UDP after successful validation
        :param func_lib: drawing inbuilt class, see in Help: UDMFunctionLibrary
        :param param_values: list of udp parameter values (user input) generated by UDP Core
        :return: None
        """
        param_dict = self.get_param_dict(param_values)

        start_point = UDPPosition(0, 0, 0)
        box = func_lib.CreateBox(start_point, [
            param_dict["x_size"],
            param_dict["y_size"],
            param_dict["z_size"]
        ])

        # points on the middle of 4 vertical edges
        points = [
            [0, 0, param_dict["z_size"]/2],
            [param_dict["x_size"], 0, param_dict["z_size"]/2],
            [param_dict["x_size"], param_dict["y_size"], param_dict["z_size"]/2],
            [0, param_dict["y_size"], param_dict["z_size"]/2]
        ]
```

```
edges = [func_lib.GetEdgeIDFromPosition(UDPPosition(point[0], point[1], point[2])) for point in points]

fillet_rad = 0.1 * param_dict["x_size"] # 10% of X size
fillet_opt = UDPBLNDFilletOptions(True, BLNDFilletRadiusLaw.BLNDConstantRadius, fillet_rad, 0.0, True, BLNDFilletType.BLNDRound, 0.0, 0.0)

chamfer_length = 0.1 * param_dict["x_size"] # 10% of X size
chamfer_opt = UDPBLNDChamferOptions(False, BLNDChamferRangeLaw.BLNDConstantRange, chamfer_length, 0.0)

# select your geometry to which to apply operations
blend_element = UDPBLNDElements(box)

# specify attribute ListOfEdges to which edges to apply fillet operation
blend_element.ListOfEdges = edges[0:2]
func_lib.Fillet(blend_element, fillet_opt)

# redeclare attribute ListOfEdges to which edges to apply chamfer operation
blend_element = UDPBLNDElements(box)
blend_element.ListOfEdges = edges[2:4]
func_lib.Chamfer(blend_element, chamfer_opt)

# Provide to the user Info message indicating success
func_lib.AddMessage(MessageSeverity.InfoMessage, "Completed!")

def GetPrimitiveTypeInfo(self):
    return primitive_info
```

```
def GetLengthParameterUnits(self):
    return length_units

def GetPrimitiveParametersDefinition2(self):
    return primitive_param_definitions

def AreParameterValuesValid2(self, error, udp_params):
    return True

# Custom Functions

def get_param_value_by_name(self, param_values, param_name):
    """
        Function to get a value of a single parameter accessing it
        by name
        :param param_values: list of udp parameter values (user
        input) generated by UDP Core
        :param param_name: name of the parameter as specified in
        definition list
        :return: Value of the parameter or None if parameter does
        not exist
    """
    param_dict = self.get_param_dict(param_values)
    value = param_dict.get(param_name, None)
    return value

def get_param_dict(self, param_values):
    """
        Function to return a dictionary of UDP parameter name and
        value (key: value) pairs
        :param param_values: list of udp parameter values (user
        input) generated by UDP Core
        :return: dict of parameter name and values
    """
```

```
"""
udm_param_def = self.GetPrimitiveParametersDefinition2()
param_dict = {}
for i, param in enumerate(udm_param_def):
    param_value = param_values[i].Data
    if str(param.PropType) != "Menu":
        param_dict[param.Name] = param_value
    else:
        param_dict[param.Name] = param_value.replace("'",',
').split(",")[0]
return param_dict
```

## Introduction to CPython (Beta)

An Ansys Electronics Desktop Beta feature allows CPython to be used for standalone scripting, as an alternative to IronPython, to:

- Launch Ansys Electronics Desktop ([InitializeNew](#))
- Connect with a running instance of Ansys Electronics Desktop ([Initialize](#))
- Execute Ansys Electronics Desktop script functions

One advantage of CPython is the large set of libraries and tools that are available. See below for instructions on modifying a script so that it can be launched with CPython interpreters.

### Important:

Launching Ansys Electronics Desktop on a remote machine is not supported in this release. Initialize() will connect to a remote instance that is already running on the remote machine, but cannot launch a new instance on the remote machine.

### Important:

CPython client scripting is not yet supported on Linux, though `ansysedt -grpcsv` can be used to launch an Electronics Desktop instance on Linux, which can be connected to or from a Windows machine via Initialize().

## Creating an External Script

While [the same as IronPython](#) when run externally, a CPython recorded script must be modified by adding the following lines to the beginning of your script *before* `import ScriptEnv`

---

```

import sys

    # Imports the sys module containing system-specific functions native to Python.

    sys.path.append(r"<InstallationPath>/PythonFiles/DesktopPlugin")

        # Adds the PythonFiles/DesktopPlugin subfolder to the list of directories Python searches for modules and files.

```

Those lines are followed by:

```

import ScriptEnv

    # This imports ScriptEnv.py from the installation path specified above.

```

Follow that with:

- Either InitializeNew() or Initialize(), as described below.
- Any desired Electronics Desktop scripting commands.
- Closing command, as described below.

## Launching Electronics Desktop

To launch a new instance of Electronics Desktop and connect oApplication and oDesktop to it:

```

InitializeNew(NonGraphical = <True|False>, Module = None, Machine
= "", Port = <Port#>)

```

Where:

- **NonGraphical** – Specifies whether to launch Electronics Desktop in non-graphical mode.
- **Module** – Behavior remains unchanged from [Iron Python](#) and should be left defaulted to "None." See the code in ScriptEnv.py for more details.
- **Machine** – Currently an empty string, as InitializeNew() will only launch Electronics Desktop on the current machine.
- **Port** – Electronics Desktop will launch using the first unused port it finds starting at <Port#>. If Port = 0, the starting port will be 50051.

### Note:

InitializeNew() will *always* launch a new instance of Electronics Desktop. Please use Initialize() to connect to an existing instance. See below.

## Connecting with a Running Instance of Electronics Desktop

To connect oApplication and oDesktop to an existing Electronics Desktop instance, or launch a new instance and connect to it if necessary:

```
Initialize(name, NG = <True|False>, machine = "", port = <Port#>)
```

Where:

- **Name** – Ignored.
- **NG** – If launch is necessary, specifies whether to launch Electronics Desktop in non-graphical mode.
- **Machine** – The machine on which to launch/connect. For current machine, pass empty string or use localhost.
- **Port** – If port is nonzero, the script tries to connect to an existing instance on <port#> running on <machine>. If there is no instance running on that <port>, a new instance of Electronics Desktop launches on that port and then connects to it. If port = 0, the new instance is launched on the first free port, starting at 50051.

## Closing Electronics Desktop/Ending the Script

To close Electronics Desktop, add the following line to the end of the script:

```
ScriptEnv.Shutdown()
```

```
# This stops ScriptEnv.py. If you are running multiple scripts, include this only at the end of the last script.
```

### -grpcsrv Flag

This flag will launch the application in a mode where the executable serves as a scripting server that can be used for CPython scripting in conjunction with the CPython stand alone scripting instructions that were mentioned earlier. The -ng flag can be combined with -grpcsrv.

On Windows:

```
ansysedt.exe -grpcsrv <optional port number>.
```

On Linux:

```
ansysedt -grpcsrv <optional port number>.
```

If the port number is omitted, the default of 50051 will be used.

With -grpcsrv, a message will be displayed in the **Messages** window indicating that the server was started. If the requested port is in use by another application, starting the sever may fail.

### Related Topics:

[Standalone Scripting in Iron Python](#)

# Ansys Electronics Desktop Scripting

This chapter provides an overview of scripting in Ansys Electronics Desktop.

[Overview of Ansys Electronics Desktop Scripting Objects](#)

[Running a Script](#)

[Recording a Script](#)

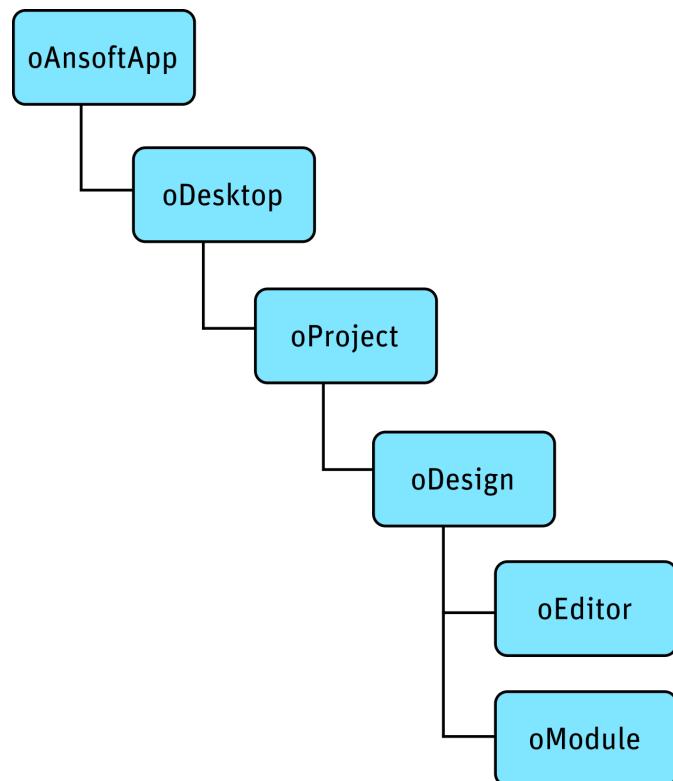
[Working with Project Scripts](#)

[Executing a Script from Within a Script](#)

[Ansys Electronics Desktop Scripting Conventions](#)

## Overview of Electronics Desktop Scripting Objects

When you record a script using Ansys Electronics Desktop, the beginning of the script must contain some standard commands, as illustrated in the following chart. The commands in the chart define the objects used by Electronics Desktop in the script and assign values to these objects. The objects are used in the hierarchical order shown.



The commands are described below, followed by examples.

## **oAnsoftApp**

The **oAnsoftApp** object provides a handle for Iron Python or VBscript to access the Ansoft.ElectronicsDesktop product.

In Iron Python, for example:

```
oAnsoftApp = CreateObject('Ansoft.ElectronicsDesktop')
```

In VBscript, for example:

```
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
```

## **oDesktop**

The **oDesktop** object is used to perform desktop-level operations, including project management.

In Iron Python, for example:

```
oDesktop = oAnsoftApp.GetAppDesktop()
```

In VBscript, for example:

```
Set oDesktop = oAnsoftApp.GetAppDesktop()
```

For script commands recognized by the **oDesktop** object, consult: [Desktop Object Script Commands](#).

## **oProject**

The **oProject** object corresponds to one project open in Electronics Desktop. It is used to manipulate the project and its data. Its data includes variables, material definitions, and one or more designs.

In Iron Python, for example:

```
oProject = oDesktop.GetActiveProject()
```

In VBscript, for example:

```
Set oProject = oDesktop.GetActiveProject()
```

Consult the following for details about script commands recognized by **oProject**:

- [Project Object Script Commands](#)
- [Property Script Commands](#)
- [Dataset Script Commands](#)

## **oDesign**

The **oDesign** object corresponds to a design in the project. This object is used to manipulate the design and its data, including variables, modules, and editors.

In Iron Python, for example:

```
oDesign = oProject.GetActiveDesign()
```

In VBscript, for example:

```
Set oDesign = oProject.GetActiveDesign()
```

Consult the following for details about script commands recognized by `oDesign`:

- [Design Object Script Commands](#)
- [Output Variable Script Commands](#)
- [Reporter Editor Script Commands](#)

## **oEditor**

The `oEditor` object corresponds to an editor, such as the 3D Modeler, Layout, or Schematic editor. This object is used to add and modify data in the editor.

In Iron Python, for example:

```
oEditor = oDesign.SetActiveEditor('3D Modeler')
```

In VBscript, for example:

```
Set oEditor = oDesign.SetActiveEditor("3D Modeler")
```

Consult the following for details about script commands recognized by `oEditor`:

- [3D Modeler Editor Script Commands](#)
- [Schematic Editor Script Commands](#)

### **Important:**

There is no Reporter Editor object for `oEditor`. Reporter Editor commands are executed by `oDesign`.

See: [Reporter Editor Script Commands](#).

## **oModule**

The `oModule` object corresponds to a module in the design. Modules are used to handle a set of related functionalities.

In IronPython, for example:

```
oModule = oDesign.GetModule('BoundarySetup')
```

In VBscript, for example:

```
Set oModule = oDesign.GetModule("BoundarySetup")
```

Consult the following for details about script commands recognized by `oModule`:

- Analysis Module Script Commands
- Boundary and Excitation Module Script Commands
- Field Overlays Module Script Commands
- Mesh Operations Module Script Commands
- Optimetrics Module Script Commands
- Radiation Module Script Commands
- Reduce Matrix Module Script Commands
- Solutions Module Script Commands

### **Example Script Opening**

Combining the above objects, a script in Iron Python could begin like the following:

```
oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
oDesktop = oAnsoftApp.GetAppDesktop()
oProject = oDesktop.SetActiveProject("Project1")
oDesign = oProject.SetActiveDesign("Design1")
oEditor = oDesign.SetActiveEditor("3D Modeler")
oModule = oDesign.GetModule("BoundarySetup")
```

In VBscript, this would be:

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule

Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
Set oProject = oDesktop.SetActiveProject("Project1")
Set oDesign = oProject.SetActiveDesign("Design1")
Set oEditor = oDesign.SetActiveEditor("3D Modeler")
Set oModule = oDesign.GetModule("BoundarySetup")
```

## GetActiveProject and GetActiveDesign for Wider Use

The sample script above only works for "Design1" within "Project1". To create a script that is usable for any open project, you can use one or both of `GetActiveProject` and `GetActiveDesign`.

In IronPython:

```
oProject = oDesktop.GetActiveProject()
oDesign = oProject.GetActiveDesign()
```

In VBscript:

```
Set oProject = oDesktop.GetActiveProject()
Set oDesign = oProject.GetActiveDesign()
```

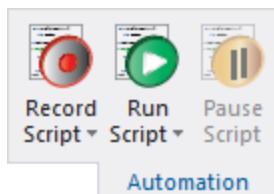
## Running a Script

Electronics Desktop scripts can be run from within the software or from the command line.

### Within Electronics Desktop

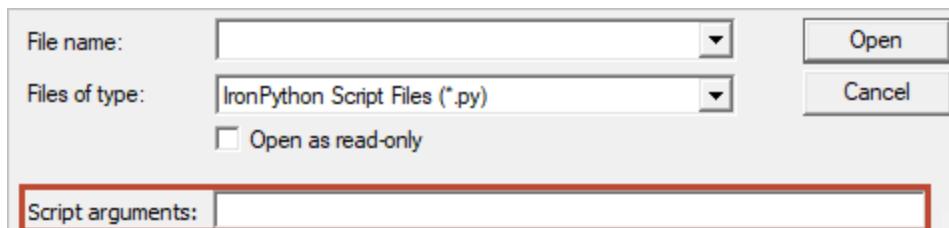
To run scripts in Electronics Desktop:

1. Click **Tools > Run Script**, or select the **Automation** tab and click the **Run Script** icon:



The **Run Script** file browser appears.

2. Use the file browser to locate the script file (\*.vbs, \*.py, or \*.js).
3. If desired, type script arguments in the Script Arguments field:



You can access script arguments using the **AnsoftScriptHost.arguments** collection from VBScript. This is a standard COM collection.

4. Click **Open**.

Electronics Desktop executes the script.

While script execution is in progress, the **Run Script** button transforms into a **Stop Script** button. Click **Stop Script** to stop the script execution.

To temporarily pause a running script, click **Pause Script**. This button transforms into a **Resume Script** button, which you can click to resume script execution.

### From the Command Line

To run a script from a command line, add the **-runscriptandexit** or **-runscript** argument to the Electronics Desktop command line syntax.

To use script arguments, add the **-scriptargs** parameter and specify the arguments. For example:

```
ansysedt.exe -scriptargs "hello there"
```

In Iron Python, the command line parameter following **-scriptargs** is passed without modification as a single string in the **ScriptArgument** python variable.

In VBscript, the command line parameter following **-scriptargs** is split into multiple strings and converted to a VBscript collection which is accessible via the **AnsoftScript.Arguments** collection. To access these arguments, for example:

```
msgbox AnsoftScript.Arguments(0) // Returns 'hello'  
msgbox AnsoftScript.Arguments(1) // Returns 'there'
```

For more information about running a script from the command line, consult the Maxwell help topic "Running Maxwell from the Command Line".

### Direct Launch

If you run a script directly from the command line without launching Electronics Desktop, script arguments will be in the **WSH.arguments** collection instead of the **AnsoftScriptHost.arguments** collection. The following script ensures the arguments are accessed regardless of the collection in which they reside:

```
on error resume next  
  
dim args  
  
Set args = AnsoftScript.arguments  
  
if (IsEmpty(args)) then  
  
Set args = WSH.arguments
```

```

End if

on error goto 0

    // At this point, args has the arguments regardless of col-
lection

msgbox "Count is " & args.Count
for i = 0 to args.Count - 1
msgbox args(i)
next

```

## Recording a Script

Electronics Desktop can record a script based on UI actions and save this script in either IronPython (\*.py) or VBscript (\*.vbs) format.

Scripts can be saved to an [external file](#), or [to the project](#).

### Important:

When you record a script, every subsequent action you take is recorded. You must manually stop recording.

## Recording a Script to File

To record a script to file:

1. Click **Tools > Record Script to File**, or select the **Automation** tab and click the **Record Script** icon:

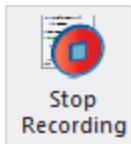


A **Save As** file browser appears.

2. Navigate to the location where you want to save the script.
3. In the **File Name** field, type a name for the script file.
4. Use the **Save as Type** drop-down menu to select either IronPython or VBscript.

5. Click **Save**.

The **Record Script** button transforms into a **Stop Recording** button, and Electronics Desktop begins recording your actions.



6. Perform the steps you want to record.

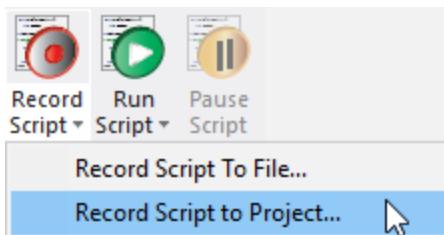
7. When you have finished recording the script, click **Stop Recording**, or select **Tools > Stop Script Recording**.

The recorded script is saved to the folder you specified.

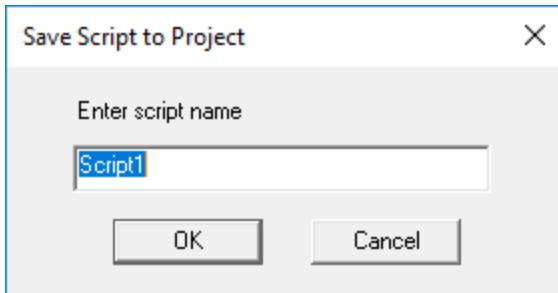
### Recording a Script to a Project

To record a script to a project:

1. Click **Tools > Record Script to Project**, or select the **Automation** tab and use the **Record Script** drop-down menu to select **Record Script to Project**.



The **Save Script to Project** dialog box appears:



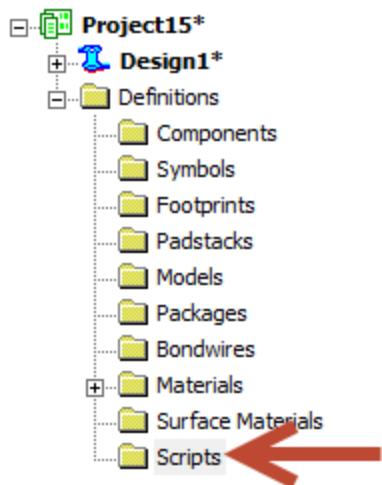
2. Enter a name for the script in the text box, then click **OK**.
3. Perform the steps you want to record.
4. When you have finished, click **Stop Recording**, or select **Tools > Stop Script Recording**.

The recorded script is saved to *scriptname.vbs* in the Scripts library and can be accessed from the Project Manager. See: [Working with Project Scripts](#).

## Working with Project Scripts

Scripts can be [recorded to a project](#).

Once a script has been recorded to the project, you can manage it in the **Project Manager** from the **Definitions** folder:



Individual scripts appear in this folder. Right-click a script to edit or run it:



You can also run project scripts from the **Automation** tab by selecting **Run Script > Project Scripts > [Script Name]**.

**Note:**

Project scripts are stored in the project scripts library. Refer to the topic "Managing Library Contents" for information on working with libraries.

## Executing a Script from Within a Script

Electronics Desktop provides a script command that enables you to launch another script from within the script that is being executed.

In IronPython:

```
oDesktop.RunScript (<ScriptName>)
```

In VBscript:

```
oDesktop.RunScript <ScriptName>
```

If the full path to the script is not specified, Electronics Desktop searches for the specified script in the following locations, in order:

1. Personal Library Directory (PersonalLib)
2. User Library Directory (UserLib)
3. System Library Directory (SysLib)
4. Installation Directory

Each of the library directories can be specified in Electronics Desktop under **Tools > Options > General Options**, on the **Project Options** tab.

## Electronics Desktop Scripting Conventions

A number of scripting conventions exist for Electronics Desktop regarding syntax, arguments, and numerical values.

Consult the following topics:

- [Named Arguments](#)
- [Setting Numerical Values](#)

- [Event Callback Scripting](#)

## Named Arguments

Many Electronics Desktop script commands use named arguments. The names can appear in three ways:

1. Named data, where name precedes data.

For example:

```
..., "SolveInside:=", true, ...
```

2. Named Array, where name precedes array.

For example:

```
..., "Attributes:=", Array(...), ...
```

3. Named Array, where name is inside an array.

For example:

```
..., Array("NAME:Attributes", ...), ...
```

In the first and second examples, the name is formatted as "<Name>:=". This signals to Electronics Desktop that this is a name for the next argument in the script command. In the third example, the name is formatted as "NAME:<name>" and is the first element of the array.

The names are used both to identify what the data means to you and to inform Electronics Desktop which data is being given. The names must be included or the script will not play back correctly. However, if you are writing a script, you do not need to pass in every piece of data that the command can take. For example, if you are modifying a boundary, the script will be recorded to include every piece of data needed for the boundary, whether or not it was modified. If you are writing a script by hand, you can add only the data that changed and omit anything that you do not want to change. Electronics Desktop will use the names to determine which data you provided.

## VBscript Example

For example, when editing an impedance boundary, Electronics Desktop records the `EditImpedance` command as follows in VBscript:

```
oModule.EditImpedance "Imped1", Array("NAME:Imped1",
    "Resistance:=", "100", "Reactance:=", "50",
    "InfGroundPlane:=", false)
```

If you only wish to change the resistance, you can leave out the other data arguments when manually writing a script:

```
oModule.EditImpedance "Imped1", Array("NAME:Imped1",
```

```
"Resistance:=", "100")
```

### IronPython Example

When editing a port excitation, Electronics Desktop records the `Edit` command as follows in IronPython:

```
oModule.Edit("Port1",
    [
        ["NAME:Port1",
            [
                ["NAME:Properties",
                    [
                        "PortSolver:=", "true",
                        "Phase:=", "0deg",
                        "Magnitude:=", "2mA",
                        "Impedance:=", "500Ohm",
                        "Theta:=", "0deg",
                        "Phi:=", "0deg",
                        "PostProcess:=", "false",
                        "Renormalize:=", "50Ohm + 0i Ohm",
                        "Deembed:=", "0mm",
                        "RefToGround:=", "false"
                    ],
                    [
                        "Type:=", "EdgePort",
                        "IsGapSource:=", true,
                        "UpperProbe:=", false,
                        "LayoutObject:=", "Port1",
                        "Pin:=", "",
                        "ReferencePort:=", ""
                    ]
                ]
            )
        ]
    ]
```

If you only wish to change the magnitude, you can leave out the other data arguments when manually writing a script:

```
oModule.Edit("Port1",
    [
        ["NAME:Port1",
            [
                ["Magnitude:=", "1mA"]
            ]
        ]
    ]
```

```
]  
)
```

## Setting Numerical Values

For script arguments that expect a number, the following options are possible:

- Pass in the number directly.

In IronPython:

```
oModule.EditVoltage('Voltage1', ['NAME:Voltage1',  
'Voltage:=' , 3.5])
```

In VBscript:

```
oModule.EditVoltage "Voltage1", Array("NAME:Voltage1",  
"Voltage:=" , 3.5)
```

- Pass in a string containing the number with units.

In IronPython:

```
oModule.EditVoltage('Voltage1', ['NAME:Voltage1',  
'Voltage:=' , '3.5V'])
```

In VBscript:

```
oModule.EditVoltage "Voltage1", Array("NAME:Voltage1",  
"Voltage:=" , "3.5V" )
```

- Pass in a variable name.

In IronPython:

```
var = 3.5  
  
oModule.EditVoltage('Voltage1', ['NAME:Voltage1',  
'Voltage:=' , var])
```

In VBscript:

```
dim var  
  
var = "3.5V"  
  
oModule.EditVoltage "Voltage1", Array("NAME:Voltage1",  
"Voltage:=" , var)
```

## Event Callback Scripting

Event Callback scripting allows you to define custom JavaScript and VBScript routines that will run automatically after detecting a triggering event, such as placing a component or running a simulation. When defining an Event Callback script, specify one or more scripts that will be run after a particular event is detected.

A callback script can only access functions and other scripts defined by its callback definition. For example, a Simplorer callback script can call PropHost.GetValue — and all other PropHost functions — but only from scripts defined in the Property Dialog callback. As a result, "PropHost" is a Simplorer script item that is only visible in "Property" callback scripts. For more information, see [Callback Scripting Using PropHost Object](#) and [Callback Scripting Using ComplInstance Object](#).

The following table lists allowable callback events, items that are visible from the associated callback script, and the set of accessible functions that can be called.

Callback Event	Scripts Visible from the Event Callback Script	Functions Callable from the Visible Script
Place Component	ComplInstance	<b>ComplInstance.GetParentDesign()</b> — Returns a oDesign item that can be used to call Design functions.  <b>ComplInstance.GetPropserverName()</b> – Returns a ComplInstance identification name that can be used in oEditor property-method scripts, such as GetPropertyValue(), SetPropertyValue(), etc.  <b>ComplInstance.GetComponentName()</b> — Returns the component name, e.g. "MS_TRL".
Simulate Component	ComplInstance	<b>ComplInstance.GetDesign()</b> — Returns the interface to the specified design simulation.  <b>ComplInstance.GetProgress()</b> – Returns the completion percentage (from 0 to 100) of the specified design simulation.  <b>ComplInstance.GetRunStatus()</b> — Returns the status number of the specified design simulation.  <b>ComplInstance.Abort()</b> — Aborts the specified design simulation.

The function, **ExecuteAnsoftScript(<ScriptName>)**, searches the configured Ansys Electronics Desktop script libraries by name for the script passed to it, and invokes the found ScriptName. The invoked script will run with the same set of visible script items as the originally called script. That is, **ComlInstance** is visible from the invoked sub-script, ScriptName, and ComlInstance's functions can be called from ScriptName.

This page intentionally  
left blank.

## 2 - Application Object Script Commands

The Application object commands permit you to get the AppDesktop. Application object commands should be executed by the **oAnsoftApp** object.

```
oAnsoftApp.<CommandName> <args>
```

### General Application Script Commands

The following are general script commands recognized by the **oAnsoftApp** object:

- [GetAppDesktop](#)

The following deprecated commands are no longer supported and produce an error if used.

- GetDesiredRamMBLimit (deprecated)
- GetHPCLicenseType (deprecated)
- GetMaximumRamMBLimit (deprecated)
- GetMPISpawnCmd(deprecated)
- GetMPIVendor (deprecated)
- GetNumberOfProcessors (deprecated)
- GetUseHPCForMP (deprecated)
- SetDesiredRamMBLimit (deprecated)
- SetHPCLicenseType (deprecated)
- SetMaximumRamMBLimit (deprecated)
- SetMPISpawnCmd (deprecated)
- SetMPIVendor (deprecated)

- SetNumberOfProcessors (deprecated)
- SetUseHPCForMP (deprecated)

## GetAppDesktop

GetAppDesktop is a function of oAnsoftApp. This function does not take an input and it returns an object. The object is assigned to the variable oDesktop.

<b>UI Access</b>	NA		
<b>Parameters</b>	Name	Type	Description
	None		
<b>Return Value</b>	Object		

<b>Python Syntax</b>	GetAppDesktop()
<b>Python Example</b>	<pre>oDesktop = oAnsoftApp.GetAppDesktop()</pre>

<b>VB Syntax</b>	GetAppDesktop()
<b>VB Example</b>	<pre>Set oDesktop = oAnsoftApp.GetAppDesktop()</pre>

## GetApp

Obtains the product application object.

<b>UI Access</b>	NA		
<b>Parameters</b>	Name	Type	Description
	<productname>	String	One of "Twin Builder", "HFSS", "Maxwell", "Q3D Extractor", "Maxwell Circuit Editor", "Designer"
<b>Return Value</b>	Object		

<b>Python Syntax</b>	GetApp(<productname>)
<b>Python Example</b>	<pre>oHFSSApp = oAnsoftApp.GetApp("HFSS")</pre>

<b>VB Syntax</b>	GetApp(<productname>)
<b>VB Example</b>	<pre>Dim oHFSSApp  oHFSSApp = oAnsoftApp.GetApp("HFSS")</pre>

This page intentionally  
left blank.

## 3 - Desktop Object Script Commands

Desktop commands should be executed by the oDesktop object. Some new commands permit you to query objects when you do not know the names.

```
Set oDesktop =  
    CreateObject("Ansoft.ElectronicsDesktop")  
oDesktop.CommandName <args>
```

[AddMessage](#)

[AreSimulationsRunning](#)

[ClearMessages](#)

[CloseAllWindows](#)

[CloseProject](#)

[CloseProjectNoForce](#)

[Count](#)

[DeleteProject](#)

[DownloadJobResults](#)

[EnableAutoSave](#)

[ExportOptionsFiles](#)

[GetActiveProject](#)

[GetActiveScheduler](#)

[GetActiveSchedulerInfo](#)

[GetAutoSaveEnabled](#)

[GetBuildDateTimeString](#)

[GetCustomMenuSet](#)

[GetDefaultUnit](#)

[GetDesktopConfiguration](#)

[GetDistributedAnalysisMachines](#)

[GetDistributedAnalysisMachinesForDesignType](#)

[GetExeDir](#)

[GetGDIObjectCount](#)

[GetLibraryDirectory](#)

[GetLocalizationHelper](#)

[GetMessages](#)

[GetPersonalLibDirectory](#)

[GetPPELicensingEnabled](#)

[GetProcessID](#)

[GetProjectDirectory](#)

[GetProjectList](#)

[GetProjects](#)

[GetRegistryInt](#)

[GetRegistryString](#)

[GetRunningInstancesMgr](#)

[GetSchematicEnvironment](#)

[GetScriptingToolsHelper](#)

[GetSysLibDirectory](#)

[GetTempDirectory](#)

[GetUserLibDirectory](#)

[GetVersion](#)

[IsFeatureEnabled](#)

[KeepDesktopResponsive](#)

[LaunchJobMonitor](#)

[NewProject](#)

[OpenAndConvertProject](#)

[OpenMultipleProjects](#)

[OpenProject](#)

[OpenProjectWithConversion](#)

[PauseRecording](#)

[PauseScript](#)

[Print](#)

[QuitApplication](#)

[RefreshJobMonitor](#)

[ResetLogging](#)

[RestoreProjectArchive](#)

[RestoreWindow](#)  
[ResumeRecording](#)  
[RunACTWizardScript](#)  
[RunProgram](#)  
[RunScript](#)  
[RunScriptWithArguments](#)  
[SelectScheduler](#)  
[SetActiveProject](#)  
[SetActiveProjectByPath](#)  
[SetCustomMenuSet](#)  
[SetDesktopConfiguration](#)  
[SetLibraryDirectory](#)  
[SetProjectDirectory](#)  
[SetRegistryFromFile](#)  
[SetRegistryInt](#)  
[SetRegistryString](#)  
[SetSchematicEnvironment](#)  
[SetTempDirectory](#)  
[ShowDockingWindow](#)  
[Sleep](#)

[StopSimulations](#)[SubmitJob](#)[Tile Windows](#)**Related Topics:**[Desktop Commands For Registry Values](#)[ImportExport Tool Commands](#)

## AddMessage

Add a message with severity and context to message window.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<projectName>	String	Project name. Passing an empty string adds the message as the desktop global message.
	<designName>	String	Design name. Ignored if project name is empty. Passing an empty string adds the message to project node in the message tree.
	<severity>	Integer	One of "Error", "Warning" or "Info". Anything other than the first two is treated as "Info" 0 = Informational, 1 = Warning, 2 = Error, 3 = Fatal
	<msg>	String	The message for the message window.
	<category>	String	Optional. The category is created with the message under the design tree node if the category does not exist. If the category already exists, the new message is added to the end of the existing category. It is ignored if the project or design is empty. If missing or empty, the message is added to the Design node in the message tree.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	AddMessage( <projectName>, <designName>, <severity>, <msg>, <category>)
<b>Python Example</b>	<code>oDesktop.AddMessage("Project1", "Maxwell", 0, "This is a test message", "")</code>

<b>VB Syntax</b>	AddMessage <projectName>, <designName>, <severity>, <msg>, <category>
<b>VB Example</b>	<code>oDesktop.AddMessage "Project1", "Maxwell11", 0, "This is a test message", ""</code>

## ClearMessages

Clears messages, optionally specifying severity and design.

<b>UI Access</b>	In Message Manager, right-click the project name and click <b>Clear messages for Project#</b>												
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;projectName&gt;</td><td>String</td><td>Project name, an empty string clears all project messages.</td></tr><tr><td>&lt;designName&gt;</td><td>String</td><td>Design name; ignored if project name is empty; an empty string clears messages in all designs.</td></tr><tr><td>&lt;severity&gt;</td><td>Integer</td><td>0 - clear all info messages 1 - clear all info and warning messages 2 - clear all info, warning and error messages 3 - clear all messages included info, warning, error, and fatal-error</td></tr></tbody></table>	Name	Type	Description	<projectName>	String	Project name, an empty string clears all project messages.	<designName>	String	Design name; ignored if project name is empty; an empty string clears messages in all designs.	<severity>	Integer	0 - clear all info messages 1 - clear all info and warning messages 2 - clear all info, warning and error messages 3 - clear all messages included info, warning, error, and fatal-error
Name	Type	Description											
<projectName>	String	Project name, an empty string clears all project messages.											
<designName>	String	Design name; ignored if project name is empty; an empty string clears messages in all designs.											
<severity>	Integer	0 - clear all info messages 1 - clear all info and warning messages 2 - clear all info, warning and error messages 3 - clear all messages included info, warning, error, and fatal-error											
<b>Return Value</b>	None												

<b>Python Syntax</b>	ClearMessages(<projectName>, <designName>, <severity>)
<b>Python Example</b>	<code>oDesktop.ClearMessages("", "", 3)</code>

<b>VB Syntax</b>	ClearMessages <projectName>, <designName>, <severity>
<b>VB Example</b>	<pre>Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop") Set oDesktop = oAnsoftApp.GetAppDesktop() oDesktop.ClearMessages "", "", 3</pre>

## CloseAllWindows

Closes all MDI child windows on the desktop.

<b>UI Access</b>	From main menu, <b>Window &gt; CloseAll</b> .
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	CloseAllWindows()
<b>Python Example</b>	<code>oDesktop.CloseAllWindows ()</code>

<b>VB Syntax</b>	CloseAllWindows
------------------	-----------------

**VB Example**

```
oDesktop.CloseAllWindows
```

## CloseProject

Closes a specified project. Changes to the project are not saved. Save the project using the Project command **Save** or **Save As** before closing to save changes.

UI Access	File > Close		
Parameters	Name <ProjectName>	Type String	Description The name of the project already in the Desktop that is to be closed, without path or extension
Return Value	None		

**Python Syntax**

```
CloseProject (<ProjectName>)
```

**Python Example**

```
oDesktop.CloseProject ("MyProject")
```

**VB Syntax**

```
CloseProject <ProjectName>
```

**VB Example**

```
oDesktop.CloseProject "MyProject"
```

## CloseProjectNoForce

**Use:** Close a named project currently open in the Desktop, unless a simulation is running. Changes to the project will not be saved. Save the project using the Project command **Save** or **Save As** before closing to save changes. To determine if the project has been

---

closed, use `GetProjectList` and see if the named project is present.

<b>UI Access</b>	<b>File &gt; Close</b>		
<b>Parameters</b>	Name <ProjectName>	Type String	Description The name of the project already on the Desktop that is to be closed, without path or extension
<b>Return Value</b>	None		

<b>Python Syntax</b>	<code>CloseProjectNoForce (&lt;ProjectName&gt;)</code>
<b>Python Example</b>	<code>oDesktop.CloseProjectNoForce ("MyProject")</code>

<b>VB Syntax</b>	<code>CloseProjectNoForce &lt;ProjectName&gt;</code>
<b>VB Example</b>	<code>oDesktop.CloseProjectNoForce "MyProject"</code>

## Count

Gets the total number of queried projects or designs obtained by `GetProjects()` and `GetDesigns()` commands.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Integer
<b>Python Syntax</b>	<code>Count</code>

<b>Python Example</b>	<pre>projects = oDesktop.GetProjects() numprojects = projects.Count</pre>
<b>VB Syntax</b>	<b>Count</b>
<b>VB Example</b>	<pre>Dim oAnsoftApp Dim oDesktop Dim oProject Dim oDesign Dim oEditor Dim oModule Dim oProjects Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop") Set oDesktop = oAnsoftApp.GetAppDesktop() oDesktop.RestoreWindow Dim projects set projects = oDesktop.GetProjects() for i = 0 to projects.Count - 1     msgbox projects(i).GetName()     dim designs     set designs = projects(i).GetDesigns()</pre>

```

for j = 0 to designs.Count - 1
    msgbox designs(j).GetName()
    next
next

```

## DeleteProject

Deletes a project from disk.

<b>UI Access</b>	<b>Edit &gt; Delete</b>		
<b>Parameters</b>	Name <ProjectName>	Type String	Description Name of the project
<b>Return Value</b>	None		

<b>Python Syntax</b>	DeleteProject (<ProjectName>)
<b>Python Example</b>	oDesktop.DeleteProject ("MyProject")

<b>VB Syntax</b>	DeleteProject <ProjectName>
<b>VB Example</b>	oDesktop.DeleteProject "MyProject"

## EnableAutoSave

Enable or disable autosave feature.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;enable&gt;</td><td>Boolean</td><td>True to enable autosave; False disables it.</td></tr></tbody></table>			Name	Type	Description	<enable>	Boolean	True to enable autosave; False disables it.
Name	Type	Description							
<enable>	Boolean	True to enable autosave; False disables it.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	EnableAutoSave(<enable>)
<b>Python Example</b>	<code>oDesktop.EnableAutoSave (True)</code>

<b>VB Syntax</b>	EnableAutoSave <enable>
<b>VB Example</b>	<code>oDesktop.EnableAutoSave True</code>

## ExportOptionsFiles

Copies the options config files to the *DestinationDirectory*.

<b>UI Access</b>	Tools > Options > Export Options Files ...								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;DestinationDirectory&gt;</td><td>String</td><td>The path to the destination directory.</td></tr></tbody></table>			Name	Type	Description	<DestinationDirectory>	String	The path to the destination directory.
Name	Type	Description							
<DestinationDirectory>	String	The path to the destination directory.							
<b>Return Value</b>	None								

<b>Python Syntax</b>	ExportOptionsFiles( <DestinationDirectory>)
<b>Python Example</b>	<code>oDesktop.ExportOptionsFiles ("D:/test/export/")</code>

<b>VB Syntax</b>	ExportOptionsFiles <DestinationDirectory>
<b>VB Example</b>	<code>oDesktop.ExportOptionsFiles "D:/test/export/"</code>

## GetActiveProject

Obtains the project currently active in the Desktop, as an object.

**Note:**

**GetActiveProject** returns normally if there are no active objects.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Object: the project that is currently active in the desktop.

<b>Python Syntax</b>	GetActiveProject()
<b>Python Example</b>	<code>oProject = oDesktop.GetActiveProject()</code>

<b>VB Syntax</b>	GetActiveProject
------------------	------------------

**VB Example**

```
Set oProject = oDesktop.GetActiveProject
```

## GetAutoSaveEnabled

Checks whether the autosave feature is enabled.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	<p>Integer:</p> <ul style="list-style-type: none"><li>• 1 – Autosave is enabled.</li><li>• 0 – Autosave is not enabled.</li></ul>

<b>Python Syntax</b>	GetAutoSaveEnabled()
<b>Python Example</b>	Enabled = oDesktop.GetAutoSaveEnabled()

<b>VB Syntax</b>	GetAutoSaveEnabled
<b>VB Example</b>	Enabled = oDesktop.GetAutoSaveEnabled()

## GetBuildDateTimeString

Returns a string representing the build date and time of the product;

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String build date and time, in the format: year-month-day hour:minute:second. Example: 2019-01-18 21:59:33

<b>Python Syntax</b>	<code>GetBuildDateString()</code>
<b>Python Example</b>	<code>oDesktop.GetBuildDateString()</code>

<b>VB Syntax</b>	<code>GetBuildDateString</code>
<b>VB Example</b>	<code>dnt = oDesktop.GetBuildDateString</code>

## GetCustomMenuSet

Returns the name of the current selected menu set in **Tools > Options > General Options > General > Desktop Configuration**.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String containing current menu set. For example, 'Default', 'EM', 'Twin Builder'.

<b>Python Syntax</b>	<code>GetCustomMenuSet ()</code>
<b>Python Example</b>	<code>oDesktop.GetCustomMenuSet ()</code>

--	--

<b>VB Syntax</b>	GetCustomMenuSet
<b>VB Example</b>	Set oProject = oDesktop.GetCustomMenuSet

## GetDefaultUnit

Returns the default unit for a physical quantity.

<b>UI Access</b>	<b>Tools &gt; Options &gt; General Options &gt; Default Units.</b> Note that this menu only displays units that can be changed, while the script can be used to view additional default units.								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;type&gt;</td><td>String</td><td>String containing a type of measurement.  Valid strings are (case insensitive):<ul style="list-style-type: none"><li>• "Acceleration"</li><li>• "Angle"</li><li>• "AngularAcceleration"</li><li>• "AngularDamping"</li><li>• "AngularSpeed"</li><li>• "Capacitance"</li><li>• "Conductance"</li><li>• "Current"</li></ul></td></tr></tbody></table>			Name	Type	Description	<type>	String	String containing a type of measurement.  Valid strings are (case insensitive): <ul style="list-style-type: none"><li>• "Acceleration"</li><li>• "Angle"</li><li>• "AngularAcceleration"</li><li>• "AngularDamping"</li><li>• "AngularSpeed"</li><li>• "Capacitance"</li><li>• "Conductance"</li><li>• "Current"</li></ul>
Name	Type	Description							
<type>	String	String containing a type of measurement.  Valid strings are (case insensitive): <ul style="list-style-type: none"><li>• "Acceleration"</li><li>• "Angle"</li><li>• "AngularAcceleration"</li><li>• "AngularDamping"</li><li>• "AngularSpeed"</li><li>• "Capacitance"</li><li>• "Conductance"</li><li>• "Current"</li></ul>							

- "CurrentChangeRate"
- "DataRate"
- "DeltaH" (Magnetic Field Strength)
- "Density"
- "Flux"
- "Force"
- "Frequency"
- "Inductance"
- "Length"
- "MagneticReluctance"
- "Mass"
- "MassFlowRate"
- "MomentInertia"
- "Power"
- "Pressure"
- "PressureCoefficient"
- "Resistance"
- "SaturateMagnetization" (Magnetic Inductance)
- "Speed"
- "Temperature"
- "Time"
- "Torque"
- "Voltage"

			<ul style="list-style-type: none"><li>• "VoltageChangeRate"</li><li>• "Volume"</li><li>• "VolumeFlowPerPressureRoot"</li><li>• "VolumeFlowRate"</li></ul>
<b>Return Value</b>	String containing the default unit (for example, "mm").		

<b>Python Syntax</b>	GetDefaultUnit(<type>)
<b>Python Example</b>	<code>oDesktop.GetDefaultUnit("Length")</code>

<b>VB Syntax</b>	GetDefaultUnit <type>
<b>VB Example</b>	<code>oDesktop.GetDefaultUnit("Length")</code>

## GetDesigns

Obtains all designs in the current project.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	List of objects for all designs in the project.

<b>Python Syntax</b>	GetDesigns()
----------------------	--------------

<b>Python Example</b>	<code>oProject.GetDesigns()</code>
-----------------------	------------------------------------

<b>VB Syntax</b>	<code>GetDesigns</code>
<b>VB Example</b>	<code>oProject.GetDesigns</code>

## GetDesktopConfiguration

Returns the name of the current selected configuration in **Tools > Options > General Options > General > Desktop Configuration**.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String containing current Desktop configuration. For example, 'All', 'Twin Builder'.

<b>Python Syntax</b>	<code>GetDesktopConfiguration()</code>
<b>Python Example</b>	<code>oDesktop.GetDesktopConfiguration()</code>

<b>VB Syntax</b>	<code>GetDesktopConfiguration</code>
<b>VB Example</b>	<code>Set oProject = oDesktop.GetDesktopConfiguration</code>

## GetDistributedAnalysisMachines

Gets a list of machines used for distributed analysis. You can iterate through the list using standard VBScript methods.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Returns a collection of names of machines used for distributed analysis.

<b>Python Syntax</b>	GetDistributedAnalysisMachines()
<b>Python Example</b>	<code>oDesktop.GetDistributedAnalysisMachines()</code>

<b>VB Syntax</b>	GetDistributedAnalysisMachines
<b>VB Example</b>	<code>oDesktop.GetDistributedAnalysisMachines()</code>

## GetDistributedAnalysisMachinesForDesignType

To obtain a list of the machines set up for analysis of the specified design type.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><code>&lt;designTypeName&gt;</code></td><td>String</td><td>The name of the type of design, such as "Twin Builder", "HFSS", HFSS-IE", "Maxwell 3D", "Maxwell 2D", "RMxprt", "EM Design", "Circuit", "System",</td></tr></tbody></table>	Name	Type	Description	<code>&lt;designTypeName&gt;</code>	String	The name of the type of design, such as "Twin Builder", "HFSS", HFSS-IE", "Maxwell 3D", "Maxwell 2D", "RMxprt", "EM Design", "Circuit", "System",
Name	Type	Description					
<code>&lt;designTypeName&gt;</code>	String	The name of the type of design, such as "Twin Builder", "HFSS", HFSS-IE", "Maxwell 3D", "Maxwell 2D", "RMxprt", "EM Design", "Circuit", "System",					

		"Q3D Extractor", "2D Extractor"
<b>Return Value</b>	Object; returns a collection of machine names.	

<b>Python Syntax</b>	GetDistributedAnalysisMachinesForDesignType (<designTypeName>)
<b>Python Example</b>	<pre>machineNames =oDesktop. GetDistributedAnalysisMachinesForDesignType ("Maxwell")</pre>

<b>VB Syntax</b>	GetDistributedAnalysisMachinesForDesignType <designTypeName>
<b>VB Example</b>	<pre>Set machineNames =oDesktop. GetDistributedAnalysisMachinesForDesignType ("Maxwell")</pre>

## GetExeDir

Returns the path where the executable is located.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	<p>String path where executable is located.  Example: 'C:/Program Files/AnsysEM/v232/Win64/'</p>

<b>Python Syntax</b>	GetExeDir()
<b>Python Example</b>	<code>oDesktop.GetExeDir()</code>

<b>VB Syntax</b>	GetExeDir
<b>VB Example</b>	<code>oDesktop.GetExeDir</code>

## GetGDIObjectCount

**Note:**

This command is for internal Ansys use only.

<b>Python Syntax</b>	GetGDIObjectCount()
<b>Python Example</b>	<code>oDesktop.GetGDIObjectCount()</code>

<b>VB Syntax</b>	GetGDIObjectCount()
<b>VB Example</b>	<code>oDesktop.GetGDIObjectCount()</code>

## GetLibraryDirectory

Get the path to the SysLib directory.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>None</td> <td></td> <td></td> </tr> </tbody> </table>	Name	Type	Description	None		
Name	Type	Description					
None							
<b>Return Value</b>	<p>String</p> <p>The path to the SysLib directory.</p>						

<b>Python Syntax</b>	GetLibraryDirectory()
<b>Python Example</b>	AddInfoMessage(str(oDesktop.GetLibraryDirectory()))

<b>VB Syntax</b>	GetLibraryDirectory
<b>VB Example</b>	MsgBox oDesktop.GetLibraryDirectory

*VB Example:*

---

message box returns the path in this example

---

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
```

```
Dim oEditor  
  
Dim oModule  
  
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")  
  
Set oDesktop = oAnsoftApp.GetAppDesktop()  
  
oDesktop.RestoreWindow  
  
libdir = oDesktop.GetLibraryDirectory  
  
msgbox(oDesktop.GetLibraryDirectory())
```

## GetLocalizationHelper

**Note:**

This command is for internal Ansys use only.

Returns the object for the localization helper.

<b>UI Access</b>	NA								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>None</td><td></td><td></td></tr></tbody></table>			Name	Type	Description	None		
Name	Type	Description							
None									
<b>Return Value</b>	<p>Object Localization helper object, such as "IDispatch(ILocalizationHelper)"</p>								

**Python Syntax**

GetLocalizationHelper()

<b>Python Example</b>	<code>oDesktop.GetLocalizationHelper()</code>
-----------------------	---

<b>VB Syntax</b>	<code>GetLocalizationHelper</code>
<b>VB Example</b>	<code>oDesktop.GetLocalizationHelper()</code>

## GetMessages

Get the messages from a specified project and design.

<b>UI Access</b>	NA		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;ProjectName&gt;</code>	String	Name of the project for which to collect messages. An incorrect project name results in no messages (design is ignored). An empty project name results in all messages (design is ignored)
	<code>&lt;DesignName&gt;</code>	String	Name of the design in the named project for which to collect messages. An incorrect design name results in no messages for the named project. An empty design name results in all messages for the named project
<b>Return Value</b>	Array of string messages.		

<b>Python Syntax</b>	GetMessages (<ProjectName>, <DesignName>, <Severity>)
<b>Python Example</b>	Messages = oDesktop.GetMessages ("MyProject", "Maxwell1", 1)

<b>VB Syntax</b>	GetMessages <ProjectName>, <DesignName>, <Severity>
<b>VB Examples</b>	Messages = oDesktop.GetMessages "MyProject", "Maxwell1", 1

## GetName [Desktop]

Gets names of queried projects or designs obtained by GetProjects() and GetDesigns() commands. See the example query.

<b>UI Access</b>	N/A
<b>Parameters</b>	N-
<b>Return Value</b>	<sup>O-</sup> String containing name. e- .

<b>Python Syntax</b>	GetName()
<b>Python Example</b>	set projects = oDesktop.GetProjects() project_name = projects(0).GetName()

<b>VB Syntax</b>	GetName()
------------------	-----------

**VB Example**

In this example, message box returns project name. projects(0) is the first of the several projects. Similarly projects(1) displays name of second project

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Dim oProjects
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
set projects = oDesktop.GetProjects()
project_name = projects(0).GetName()
msgbox(project_name)
```

## GetPersonalLibDirectory

Get the path to the PersonalLib directory.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String path to the PersonalLib directory.

<b>Python Syntax</b>	<code>GetPersonalLibDirectory()</code>
<b>Python Example</b>	<code>oDesktop.GetPersonalLibDirectory()</code>

<b>VB Syntax</b>	<code>GetPersonalLibDirectory</code>
<b>VB Example</b>	<code>oDesktop.GetPersonalLibDirectory</code>

## GetProcessID

Returns the process ID of ansysedt.exe.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Integer process ID of ansysedt.exe. For example, 12716.

<b>Python Syntax</b>	<code>GetProcessID ()</code>
----------------------	------------------------------

<b>Python Example</b>	<code>oDesktop.GetProcessID()</code>
-----------------------	--------------------------------------

<b>VB Syntax</b>	<code>GetProcessID</code>
<b>VB Example</b>	<code>oDesktop.GetProcessID</code>

## GetProjects

Returns a list of all the projects that are currently open in Electronics Desktop. Once you have the projects, you can iterate through them using standard VBScript methods.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Returns a collection containing objects for all open projects in Electronics Desktop.

<b>Python Syntax</b>	<code>GetProjects()</code>
<b>Python Example</b>	<code>oDesktop.GetProjects ()</code>

<b>VB Syntax</b>	<code>GetProjects</code>
<b>VB Example</b>	<code>oDesktop.GetProjects</code>

## GetProjectDirectory

Gets the path to the Project directory.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String path to the Project directory.

<b>Python Syntax</b>	GetProjectDirectory()
<b>Python Example</b>	<code>oDesktop.GetProjectDirectory()</code>

<b>VB Syntax</b>	GetProjectDirectory
<b>VB Example</b>	<code>oDesktop.GetProjectDirectory</code>

## GetProjectList

Returns a list of all projects that are open in Electronics Desktop.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array of strings containing the names of all open projects in Electronics Desktop.

<b>Python Syntax</b>	GetProjectList()
----------------------	------------------

<b>Python Example</b>	<code>list_of_projects = oDesktop.GetProjectList()</code>
-----------------------	---

<b>VB Syntax</b>	<code>GetProjectList</code>
<b>VB Example</b>	<code>list_of_projects = oDesktop.GetProjectList</code>

## GetSchematicEnvironment

Returns the name of the current schematic environment set in **Tools > Options > General Options > General > Desktop Configuration**.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	<p>Integer representing a schematic environment:</p> <ul style="list-style-type: none"> <li>• 0 = Circuit</li> <li>• 1 = Twin Builder</li> <li>• 2 = Maxwell Circuit</li> </ul>

<b>Python Syntax</b>	<code>GetSchematicEnvironment()</code>
<b>Python Example</b>	<code>oDesktop.GetSchematicEnvironment ()</code>

<b>VB Syntax</b>	<code>GetSchematicEnvironment</code>
------------------	--------------------------------------

**VB Example**

```
Set oProject = oDesktop.GetSchematicEnvironment
```

## GetScriptingToolsHelper

**Note:**

This command is for internal Ansys use only.

Returns the object for the scripting tools helper.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>None</td><td></td><td></td></tr></tbody></table>	Name	Type	Description	None		
Name	Type	Description					
None							
<b>Return Value</b>	Object ScriptingTools helper object						

<b>Python Syntax</b>	GetScriptingToolsHelper()
<b>Python Example</b>	<code>oDesktop.GetScriptingToolsHelper()</code>

<b>VB Syntax</b>	GetScriptingToolsHelper
<b>VB Example</b>	<code>oDesktop.GetScriptingToolsHelper()</code>

## GetSysLibDirectory

Get the path to the SysLib directory.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String path to the SysLib directory.

<b>Python Syntax</b>	GetSysLibDirectory()
<b>Python Example</b>	<code>oDesktop.GetSysLibDirectory()</code>

<b>VB Syntax</b>	GetSysLibDirectory
<b>VB Example</b>	<code>oDesktop.GetSysLibDirectory</code>

## GetTempDirectory

Gets the path to the Temp directory.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String path to the Temp directory.

<b>Python Syntax</b>	GetTempDirectory()
<b>Python Example</b>	<code>oDesktop.GetTempDirectory()</code>

<b>VB Syntax</b>	GetTempDirectory
<b>VB Example</b>	<code>oDesktop.GetTempDirectory</code>

## GetUserLibDirectory

Gets the path to the UserLib directory.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Stringpath to the UserLib directory.

<b>Python Syntax</b>	GetUserLibDirectory()
<b>Python Example</b>	<code>oDesktop.GetUserLibDirectory()</code>

<b>VB Syntax</b>	GetUserLibDirectory
<b>VB Example</b>	<code>oDesktop.GetUserLibDirectory</code>

## GetVersion

Returns a string representing the version.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String containing version of the product.

<b>Python Syntax</b>	<code>GetVersion()</code>
<b>Python Example</b>	<code>oDesktop.GetVersion()</code>

<b>VB Syntax</b>	<code>GetVersion()</code>
<b>VB Example</b>	<code>oDesktop.GetVersion</code>

## ImportANF

Imports an ANF file into a new project. For older ANFv2 files, use [ImportANFv2](#).

<b>UI Access</b>	<b>File &gt; Import &gt; ANF.</b>		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;ANFfilename&gt;</code>	String	Full path of ANF file.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ImportANF(<ANFfilename>)
<b>Python Example</b>	<pre>oDesktop.RestoreWindow()  Set oTool = oDesktop.GetTool('ImportExport') oTool.ImportANF('C:\\\\AnsTranslator\\\\results\\\\package4.anf')</pre>

<b>VB Syntax</b>	ImportANF <ANFfilename>
<b>VB Example</b>	<pre>Dim oAnsoftApp  Dim oDesktop  Dim oProject  Dim oDesign  Dim oEditor  Dim oModule  Dim oProjects  Dim omachine  Dim oTool  Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")  Set oDesktop = oAnsoftApp.GetAppDesktop()  oDesktop.RestoreWindow  Set oTool = oDesktop.GetTool("ImportExport")</pre>

	<code>oTool.ImportANF("%UserProfile%\Documents\HFSS Examples\package.anf")</code>
--	---

## ImportAutoCAD

Imports an AutoCAD file into a new project.

UI Access	<b>File &gt; Import &gt; AutoCAD.</b>		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;dxFileName&gt;</code>	String	Full path of DXF file.
	<code>&lt;outputPathFileName&gt;</code>	String	Full path of EDB file to create during import.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>ImportAutoCAD (&lt;dxFileName&gt;, &lt;outputPathFileName&gt;, &lt;controlFileName&gt;)</code>
<b>Python Example</b>	<pre>Set oTool = oDesktop.GetTool('ImportExport') oTool.ImportAutoCAD('C:/MyPath/a4lines.dxf', 'C:/MyPath/a4lines.aedb.edb', 'C:/MyPath/a4lines.xml')</pre>

<b>VB Syntax</b>	<code>ImportAutoCAD &lt;dxFileName&gt;, &lt;outputPathFileName&gt;, &lt;controlFileName&gt;</code>
<b>VB Example</b>	<pre>Set oTool = oDesktop.GetTool("ImportExport") oTool.ImportAutoCAD "C:/MyPath/a4lines.dxf", "C:/MyPath/a4lines.aedb.edb", "C:/MyPath/a4lines.xml"</pre>

## ImportGDSII

Imports a GDSII file into a new project.

UI Access	<b>File &gt; Import &gt; GDSII.</b>		
<b>Parameters</b>	Name	Type	Description
	<gdsiiFileName>	String	Full path of GDSII file.
	<outputPathName>	String	Full path of EDB file to create during import.
	<controlFileName>	String	Optional. Full path of XML control file. Pass empty string if none.
<b>Return Value</b>	<propertyFileName>	String	Optional. Full path to property mapping file. Pass empty string if none.
	None.		

<b>Python Syntax</b>	ImportGDSII(<gdsiiFileName>, <outputPathName>, <controlFileName>, <propertyFileName>)
<b>Python Example</b>	<pre> oDesktop.RestoreWindow() Set oTool = oDesktop.GetTool('ImportExport') oTool.ImportGDSII('C:/Files/test.gds', 'C:/Files/test.aedb.edb', 'C:/Files/test.xml', 'C:/Files/test.txt')</pre>

<b>VB Syntax</b>	ImportGDSII <gdsiiFileName>, <outputPathName>, <controlFileName>, <propertyFileName>
<b>VB Example</b>	<pre> Set oTool = oDesktop.GetTool("ImportExport") oTool.ImportGDSII "C:/Files/test.gds", "C:/Files/test.aedb.edb", _</pre>

	"C:/Files/test.xml", "C:/Files/test.txt"
--	--

## ImportODB

Imports an ODB++ file into a new project.

<b>UI Access</b>	<b>File &gt; Import &gt; ODB++.</b>												
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;odbFileName&gt;</td> <td>String</td> <td>Full path of ODB++ file.</td> </tr> <tr> <td>&lt;outputPathName&gt;</td> <td>String</td> <td>Full path of EDB file to create during import.</td> </tr> <tr> <td>&lt;controlFileName&gt;</td> <td>String</td> <td>Optional. Full path of XML control file. Pass empty string if none.</td> </tr> </table>	Name	Type	Description	<odbFileName>	String	Full path of ODB++ file.	<outputPathName>	String	Full path of EDB file to create during import.	<controlFileName>	String	Optional. Full path of XML control file. Pass empty string if none.
Name	Type	Description											
<odbFileName>	String	Full path of ODB++ file.											
<outputPathName>	String	Full path of EDB file to create during import.											
<controlFileName>	String	Optional. Full path of XML control file. Pass empty string if none.											
<b>Return Value</b>	None.												

<b>Python Syntax</b>	ImportODB(<odbFileName>, <outputPathName>, <controlFileName>)
<b>Python Example</b>	<pre> oDesktop.RestoreWindow()  Set oTool = oDesktop.GetTool('ImportExport')  oTool.ImportODB('C:/Files/test.odb', 'C:/Files/test.aedb', 'C:/Files/test.xml') </pre>

<b>VB Syntax</b>	ImportODB <odbFileName>, <outputPathName>, <controlFileName>
<b>VB Example</b>	<pre> Set oTool = oDesktop.GetTool("ImportExport")  oTool.ImportODB "C:/Files/test.odb", "C:/Files/test.aedb", _ </pre>

	"C:/Files/test.xml
--	--------------------

## LaunchJobMonitor

Use: For use in starting job monitoring. This brings up the **Monitor Job** dialog box.

<b>UI Access</b>	Launch Job Monitor		
<b>Parameters</b>	Name <projectPath>	Type String	Description Path to the project file to be monitored
<b>Return Value</b>	None		

<b>Python Syntax</b>	LaunchJobMonitor()
<b>Python Example</b>	oDesktop.LaunchJobMonitor("C:\\projects\\basic.aedt")

<b>VB Syntax</b>	LaunchJobMonitor()
<b>VB Example</b>	oDesktop.LaunchJobMonitor("C:\\projects\\basic.aedt")

## NewProject

Creates a new project. The new project becomes the active project.

<b>UI Access</b>	File > New.
<b>Parameters</b>	None.

<b>Return Value</b>	Object, the project that is added.
---------------------	------------------------------------

<b>Python Syntax</b>	NewProject()
<b>Python Example</b>	<code>oProject = oDesktop.NewProject()</code>

<b>VB Syntax</b>	NewProject
<b>VB Example</b>	<code>Set oProject = oDesktop.NewProject</code>

## OpenAndConvertProject

Opens a legacy project and converts or copies it to .aedt format.

<b>UI Access</b>	Click <b>File &gt; Open</b> , and choose a legacy project									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;itemPath&gt;</td> <td>String</td> <td>full project path of the legacy project</td> </tr> <tr> <td>&lt;legacyChoice&gt;</td> <td>Integer</td> <td>           0: show conversion dialog box, (same as <b>File &gt; Open</b> of a legacy file)            1: rename (changes extension to .aedt, the original file and results are renamed)            2: copy (creates new file with .aedt extension, and the original file and results remain available)         </td> </tr> </tbody> </table>	Name	Type	Description	<itemPath>	String	full project path of the legacy project	<legacyChoice>	Integer	0: show conversion dialog box, (same as <b>File &gt; Open</b> of a legacy file) 1: rename (changes extension to .aedt, the original file and results are renamed) 2: copy (creates new file with .aedt extension, and the original file and results remain available)
Name	Type	Description								
<itemPath>	String	full project path of the legacy project								
<legacyChoice>	Integer	0: show conversion dialog box, (same as <b>File &gt; Open</b> of a legacy file) 1: rename (changes extension to .aedt, the original file and results are renamed) 2: copy (creates new file with .aedt extension, and the original file and results remain available)								
<b>Return Value</b>	An object reference to the newly opened project which has the .aedt extension									

**Warning:** If project file/results with the same name and .aedt extension already exist in the same directory, they will be overwritten.

<b>Python Syntax</b>	<code>OpenAndConvertProject(filePath, legacyChoice)</code>
<b>Python Example</b>	<pre>oProject = oDesktop.OpenAndConvertProject("c:\files\optimtee.hfss", 1)</pre> <p><b>Note:</b> optimtee.hfss is gone after this code executes</p> <pre>oProject = oDesktop.OpenAndConvertProject("c:\files\optimtee.hfss", 2)</pre> <p><b>Note:</b> optimtee.hfss remains after this code executes</p>

<b>VB Syntax</b>	<code>OpenAndConvertProject(filePath, legacyChoice)</code>
<b>VB Example</b>	<pre>Set oProject = oDesktop.OpenAndConvertProject("c:\files\optimtee.hfss", 1)</pre> <p><b>Note:</b> optimtee.hfss is gone after this code executes</p>

## OpenMultipleProjects

Opens all files of a specified type in a specified directory.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;directory&gt;</code>	String	Path to the projects.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>OpenAndConvertProject(&lt;filePath&gt;, &lt;legacyChoice&gt;)</code>
<b>Python Example</b>	<code>oProject = oDesktop.OpenAndConvertProject("c:\files\optimtee.", "*.aedt")</code>

<b>VB Syntax</b>	<code>OpenAndConvertProject &lt;filePath&gt;, &lt; legacyChoice&gt;</code>
<b>VB Example</b>	<code>Set oProject = oDesktop.OpenAndConvertProject "c:\files\optimtee.", "*.aedt"</code>

## OpenProject

Opens a specified project.

<b>UI Access</b>	Click <b>File &gt; Open</b> .						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;FileName&gt;</code></td> <td>String</td> <td>Full path of the project to open.</td> </tr> </tbody> </table>	Name	Type	Description	<code>&lt;FileName&gt;</code>	String	Full path of the project to open.
Name	Type	Description					
<code>&lt;FileName&gt;</code>	String	Full path of the project to open.					
<b>Return Value</b>	An object reference to the newly opened project.						

<b>Python Syntax</b>	<code>OpenProject(&lt;FileName&gt;)</code>
<b>Python Example</b>	<code>oDesktop.OpenProject ("D:/Projects/Project1.aedt")</code>

<b>VB Syntax</b>	<code>OpenProject &lt;FileName&gt;</code>
<b>VB Example</b>	<code>oDesktop.OpenProject "D:/Projects/Project1.aedt"</code>

## OpenProjectWithConversion

**Note:**

This command is for internal Ansys use only.

<b>Python Syntax</b>	OpenProjectWithConversion()
<b>Python Example</b>	<code>oDesktop.OpenProjectWithConversion()</code>

## Paste (Project Object)

Pastes a design in the active project.

<b>UI Access</b>	Edit > Paste
<b>Parameters</b>	None
<b>Return Value</b>	None

<b>Python Syntax</b>	Paste()
<b>Python Example</b>	<code>oProject.Paste()</code>

<b>VB Syntax</b>	Paste
<b>VB Example</b>	<code>oProject.Paste</code>

## Paste (Project Object)

Pastes a design in the active project.

<b>UI Access</b>	Edit > Paste.
<b>Parameters</b>	None.
<b>Return Value</b>	None

<b>Python Syntax</b>	<code>Paste()</code>
<b>Python Example</b>	<code>oProject.Paste()</code>

<b>VB Syntax</b>	Paste
<b>VB Example</b>	<code>oProject.Paste</code>

## PauseRecording

Temporarily stop script recording.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>None</td> <td></td> <td></td> </tr> </tbody> </table>	Name	Type	Description	None		
Name	Type	Description					
None							

<b>Return Value</b>	None
---------------------	------

<b>Python Syntax</b>	PauseRecording()
<b>Python Example</b>	<code>oDesktop.PauseRecording()</code>

<b>VB Syntax</b>	PauseRecording
<b>VB Example</b>	<code>oDesktop.PauseRecording</code>

## PauseScript

Pause the execution of the script and pop up a message to the user. The script execution will not resume until the user chooses.

<b>UI Access</b>	Tools > Pause Script								
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;Message&gt;</td><td>String</td><td>Any Text.</td></tr></table>			Name	Type	Description	<Message>	String	Any Text.
Name	Type	Description							
<Message>	String	Any Text.							
<b>Return Value</b>	None								

<b>Python Syntax</b>	PauseScript (<Message>)
<b>Python Example</b>	<code>oDesktop.PauseScript ("Text to display in pop-up dialog box.")</code>

---

<b>VB Syntax</b>	PauseScript <Message>
<b>VB Example</b>	<code>oDesktop.PauseScript "Text to display in pop-up dialog box."</code>

## Print

Prints the contents of the active view window.

<b>UI Access</b>	<b>File &gt; Print.</b>
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	Print()
<b>Python Example</b>	<code>oDesktop.Print ()</code>

<b>VB Syntax</b>	Print
<b>VB Example</b>	<code>oDesktop.Print</code>

## QuitApplication

Exits the desktop.

<b>UI Access</b>	<b>File &gt; Exit.</b>
------------------	------------------------

<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	QuitApplication()
<b>Python Example</b>	<code>oDesktop.QuitApplication()</code>

<b>VB Syntax</b>	QuitApplication
<b>VB Example</b>	<code>oDesktop.QuitApplication</code>

## RefreshJobMonitor

For use in monitoring a job.

<b>UI Access</b>	Tools > Job Management > Monitor Jobs.
<b>Parameters</b>	None.
<b>Return Value</b>	A string specifying the job state.  The result can be any of the following strings: <ul style="list-style-type: none"><li>• "Monitor Not Visible"</li><li>• "Queued"</li><li>• "Running"</li></ul>

	<ul style="list-style-type: none"> <li>• "Shutting Down"</li> <li>• "Unknown"</li> <li>• "Completed"</li> <li>• "Not Monitoring"</li> <li>• "Starting Monitoring"</li> </ul>
--	--

<b>Python Syntax</b>	RefreshJobMonitor()
<b>Python Example</b>	<code>oDesktop.RefreshJobMonitor ()</code>

<b>VB Syntax</b>	RefreshJobMonitor
<b>VB Example</b>	<code>oDesktop.RefreshJobMonitor</code>

## ResetLogging

Redirects simulation log file to a specified directory and log level.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;logFile&gt;</code>	String	Path to log file.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ResetLogging(<logFile>, <logLevel>)
<b>Python Example</b>	<code>oDesktop.ResetLogging ("C:/Project1.aedtresults/", 1)</code>

<b>VB Syntax</b>	ResetLogging <logFile>, <logLevel>
<b>VB Examples</b>	<code>oDesktop.ResetLogging "C:/Project1.aedtresults/", 1</code>

## RestoreProjectArchive

Restores a previously archived project to a specified path.

<b>UI Access</b>	<b>File &gt; Restore Archive.</b>		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;ArchiveFilePath&gt;</code>	String	Path to archived file
	<code>&lt;ProjectFilePath&gt;</code>	String	Path to restore location
	<code>&lt;OverwriteExistingFiles&gt;</code>	Boolean	True to overwrite an existing file of the same name; else False.
	<code>&lt;OpenProjectAfterRestore&gt;</code>	Boolean	True to open the project after it is restored; else False.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	RestoreProjectArchive (<Archivefilepath>, <ProjectFilePath>, <OverwriteExistingFiles>, <OpenProjectAfterRestore>)
<b>Python Example</b>	<code>oDesktop.RestoreProjectArchive ("C:\Users\jdoe\Documents\OptimTee.aedtz", "C:\Documents\OptimTee.aedt", False, True)</code>

<b>VB Syntax</b>	RestoreProjectArchive <Archivefilepath>, <ProjectFilePath>, <OverwriteExistingFiles>, <OpenProjectAfterRestore>
<b>VB Example</b>	<code>oDesktop.RestoreProjectArchive "C:\Users\jdoe\Documents\OptimTee.aedtz", _</code> <code>"C:\Documents\OptimTee.aedt", false, true</code>

## RestoreWindow

Restores a minimized Desktop window.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	RestoreWindow()
<b>Python Example</b>	<code>oDesktop.RestoreWindow ()</code>

<b>VB Syntax</b>	RestoreWindow
<b>VB Example</b>	<code>oDesktop.RestoreWindow</code>

## ResumeRecording

Resume recording a script.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	None

<b>Python Syntax</b>	ResumeRecording()
<b>Python Example</b>	<code>oDesktop.ResumeRecording()</code>

<b>VB Syntax</b>	ResumeRecording
<b>VB Example</b>	<code>oDesktop.ResumeRecording</code>

## RunACTWizardScript

**Note:**

This command is for internal Ansys use only.

<b>Python Syntax</b>	RunACTWizardScript()
<b>Python Example</b>	<code>oDesktop.RunACTWizardScript()</code>

## RunProgram

Runs an external program.

<b>UI Access</b>	NA		
<b>Parameters</b>	Name	Type	Description
	<ProgName>	String	Name of the program to run.
	<ProgPath>	String	Location of the program. Pass in an empty string to use the system path.
	<WorkPath>	String	Working directory in which program will start.
	<ArgArray>	Array of Strings	Arguments to pass to the program. If no arguments, pass in None
<b>Return Value</b>	None		

<b>Python Syntax</b>	RunProgram (<ProgName>, <ProgPath>, <WorkPath>, <ArgArray>)
<b>Python Example</b>	<pre> oDesktop.RunProgram("winword.exe", "\"C:\\Program Files\\Microsoft Office\\Office10\", \"", None) </pre>

<b>VB Syntax</b>	RunProgram <ProgName>, <ProgPath>, <WorkPath>, <ArgArray>
<b>VB Example</b>	<pre> oDesktop.RunProgram "winword.exe", "\"C:\\Program Files\\Microsoft Office\\Office10\", \"", None </pre>

## RunScript

Launches another script from within the script currently being executed.

UI Access	Tools>Run Script		
Parameters	Name <SCriptPath>	Type String	Description <p>Name or full path of the script to execute.</p> <p>If the full path to the script is not specified, Twin Builder searches for the specified script in the following locations, in this order:</p> <ol style="list-style-type: none"><li>1. Personal library directory. This is the <b>PersonalLib</b> subdirectory in the project directory. The project directory can be specified in the <b>General Options</b> dialog box (click <b>Tools&gt;Options&gt;General Options</b> to open this dialog box) under the <b>Project Options</b> tab.</li><li>2. User library directory. This is the <b>userlib</b> subdirectory in the library directory. The library directory can be specified in the <b>General Options</b> dialog box (click <b>Tools&gt;Options&gt;General Options</b> to open this dialog box) under the <b>Project Options</b> tab.</li><li>3. System library directory. This is the <b>syslib</b> subdirectory in the library directory. The library directory can be specified in the <b>General Options</b> dialog box (click <b>Tools&gt;Options&gt;General Options</b> to open this dialog box) under the <b>Project Options</b> tab.</li><li>4. HFSS installation directory.</li></ol>

<b>Return Value</b>	Long the return code for the script method.
---------------------	--

<b>Python Syntax</b>	RunScript (<ScriptPath>)
<b>Python Example</b>	<code>oDesktop.RunScript ("C:/Project/test1.vbs")</code>

<b>VB Syntax</b>	RunScript <ScriptPath>
<b>VB Example</b>	<code>oDesktop.RunScript ("C:/Project/test1.vbs")</code>

## RunScriptWithArguments

Similar to RunScript, launch another script from within the currently executing script, but with arguments.

<b>UI Access</b>	NA		
<b>Parameters</b>	Name <ScriptPath>	Type String	Description  The name or full path of the script to execute. If the full path to the script is not specified, the software looks for the script in the following locations: <ul style="list-style-type: none"><li>• Personal library directory: "PersonalLib". The PersonalLib directory can be specified in Tools&gt;Options&gt;General Options on the 'Project Options' tab.</li><li>• User library directory: directory "userlib". The UserLib directory can be specified in Tools&gt;Options&gt;General Options on the 'Project Options'</li></ul>

			tab. <ul style="list-style-type: none"><li>• System library directory: directory "syslib". The SysLib directory can be specified in Tools&gt;Options&gt;General Options on the 'Project Options' tab.</li><li>• Software installation directory</li></ul>
	<Arguments>	String	The arguments to supply to the specified script.
<b>Return Value</b>	Long the return code for the script method.		

<b>Python Syntax</b>	RunScriptWithArguments (<ScriptPath>, <Arguments>)
<b>Python Example</b>	<pre>oDesktop.RunScriptWithArguments ("C:/Project/test2.py", "foo")</pre>

<b>VB Syntax</b>	RunScriptWithArguments <ScriptPath>, <Arguments>
<b>VB Example</b>	<pre>oDesktop.RunScriptWithArguments "C:/Project/test2.vbs", "foo"</pre>

## SelectScheduler

Selects the scheduler used for batch job submission. It tries non-graphical selection of the scheduler, attempting to get version information from the scheduler in order to check for successful selection. If unable to get the information, it displays the **Select Scheduler** dialog.

window and waits for the user to complete the settings.

UI Access	Tools > Job Management > Select Scheduler.		
<b>Parameters</b>	Name	Type	Description
	<option>	String	<p>One of the following options (not case sensitive):</p> <ul style="list-style-type: none"> <li>• Empty string for remote RSM service</li> <li>• "RSM" for local RSM</li> <li>• "Windows HPC" for Windows HPC</li> <li>• "LSF" for Load-Sharing Facility</li> <li>• "SGE" for Grid Engine (GE, OGE, SGE, UGE, etc.)</li> <li>• "PBS" for Portable Batch Scheduler/System (PBSPro, Torque, Maui, etc.)</li> <li>• "Ansys Cloud" for Ansys Cloud</li> </ul>
	<address (optional)>	String	String specifying the IP address or hostname of the head node or for the remote host running the RSM service.
	<username (optional)>	String	Username string to use for remote RSM service (or blank to use username stored in current submission host user settings). If the (non-blank) username doesn't match the username stored in current submission host user settings, then the Select Scheduler dialog is displayed to allow for password entry prior to job submission.
	<forcePasswordEntry (optional)>	String	Boolean used to force display of the Select Scheduler GUI to allow for password entry prior to job submission.
<b>Return Value</b>	The selected scheduler (if selection was successful, this string should match the input option string, although it could differ in upper/lowercase).		

<b>Python Syntax</b>	Select Scheduler(<option>, <address>, <username>, <forcePasswordEntry>)
<b>Python Example</b>	<pre>result = oDesktop.SelectScheduler("Windows HPC", "headnode.win.example.com")</pre>

<b>VB Syntax</b>	Select Scheduler <option>, <address>, <username>, <forcePasswordEntry>
<b>VB Example</b>	<pre>result = oDesktop.SelectScheduler "Windows HPC", "headnode.win.example.com"</pre>

## SetActiveProject

Specifies the name of the project that should become active in the desktop. Returns that project.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;ProjectName&gt;</td><td>String</td><td>The name of the project already in the Desktop that is to be activated.</td></tr></tbody></table>	Name	Type	Description	<ProjectName>	String	The name of the project already in the Desktop that is to be activated.
Name	Type	Description					
<ProjectName>	String	The name of the project already in the Desktop that is to be activated.					
<b>Return Value</b>	Object, the project that is activated.						

<b>Python Syntax</b>	SetActiveProject (<ProjectName>)
<b>Python Example</b>	<pre>oProject = oDesktop.SetActiveProject("Project1")</pre>

<b>VB Syntax</b>	SetActiveProject <ProjectName>
<b>VB Example</b>	Set oProject = oDesktop.SetActiveProject "Project1"

## SetActiveProjectByPath

Specifies the name of the project that should become active in the desktop. Returns that project. If a user has two projects open with the same name, the result of SetActiveProject is ambiguous (the first one listed in selected). This command permits unambiguous specification of the active project.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ProjectName&gt;</td> <td>String</td> <td>The full path name of the project already in the Desktop that is to be activated.</td> </tr> </tbody> </table>			Name	Type	Description	<ProjectName>	String	The full path name of the project already in the Desktop that is to be activated.
Name	Type	Description							
<ProjectName>	String	The full path name of the project already in the Desktop that is to be activated.							
<b>Return Value</b>	Object, the project that is activated.								

<b>Python Syntax</b>	SetActiveProjectByPath(<ProjectName>)
<b>Python Example</b>	oProject = oDesktop.SetActiveProjectByPath("c:\Projects\MyProject.aedt")

<b>VB Syntax</b>	SetActiveProjectByPath <ProjectName>
<b>VB Example</b>	Set oProject = oDesktop.SetActiveProjectByPath "c:\Projects\MyProject.aedt"

## SetCustomMenuSet

Sets the custom menu set for Electronics Desktop.

<b>UI Access</b>	Navigate to <b>Tools &gt; Options &gt; General Options &gt; General &gt; Desktop Configuration</b> . Select a configuration using the <b>Custom Menu Set</b> drop-down menu.								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;customMenuSet&gt;</td> <td>String</td> <td>           Name of desired menu set. Can be one of:  <ul style="list-style-type: none"> <li>• 'Default'</li> <li>• 'EM'</li> <li>• 'RF'</li> <li>• 'RF.0'</li> <li>• 'SI'</li> <li>• 'SI1.0'</li> <li>• 'SI2.0'</li> <li>• 'Twin Builder'</li> </ul> </td> </tr> </tbody> </table>	Name	Type	Description	<customMenuSet>	String	Name of desired menu set. Can be one of: <ul style="list-style-type: none"> <li>• 'Default'</li> <li>• 'EM'</li> <li>• 'RF'</li> <li>• 'RF.0'</li> <li>• 'SI'</li> <li>• 'SI1.0'</li> <li>• 'SI2.0'</li> <li>• 'Twin Builder'</li> </ul>		
Name	Type	Description							
<customMenuSet>	String	Name of desired menu set. Can be one of: <ul style="list-style-type: none"> <li>• 'Default'</li> <li>• 'EM'</li> <li>• 'RF'</li> <li>• 'RF.0'</li> <li>• 'SI'</li> <li>• 'SI1.0'</li> <li>• 'SI2.0'</li> <li>• 'Twin Builder'</li> </ul>							
<b>Return Value</b>	None								

<b>Python Syntax</b>	SetCustomMenuSet(<customMenuSet>)
<b>Python Example</b>	<code>oDesktop.SetCustomMenuSet ('Default')</code>

<b>VB Syntax</b>	SetCustomMenuSet(<customMenuSet>)
<b>VB Example</b>	<code>Set oProject = oDesktop.SetCustomMenuSet "EM"</code>

## SetDesktopConfiguration

Sets the desktop configuration.

<b>UI Access</b>	Navigate to <b>Tools &gt; Options &gt; General Options &gt; General &gt; Desktop Configuration</b> . Select a configuration using the <b>Set targeted configuration</b> drop-down menu.								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;configName&gt;</td> <td>String</td> <td>           Name of desired Desktop configuration. Can be one of:           <ul style="list-style-type: none"> <li>• 'All'</li> <li>• 'EM'</li> <li>• 'RF'</li> <li>• 'SI'</li> <li>• 'Twin Builder'</li> </ul> </td> </tr> </tbody> </table>			Name	Type	Description	<configName>	String	Name of desired Desktop configuration. Can be one of: <ul style="list-style-type: none"> <li>• 'All'</li> <li>• 'EM'</li> <li>• 'RF'</li> <li>• 'SI'</li> <li>• 'Twin Builder'</li> </ul>
Name	Type	Description							
<configName>	String	Name of desired Desktop configuration. Can be one of: <ul style="list-style-type: none"> <li>• 'All'</li> <li>• 'EM'</li> <li>• 'RF'</li> <li>• 'SI'</li> <li>• 'Twin Builder'</li> </ul>							
<b>Return Value</b>	None								

<b>Python Syntax</b>	SetDesktopConfiguration (<configName>)
<b>Python Example</b>	<code>oDesktop.SetDesktopConfiguration ('RF')</code>

<b>VB Syntax</b>	SetDesktopConfiguration(<configName>)
<b>VB Example</b>	<code>Set oProject = oDesktop.SetDesktopConfiguration "SI"</code>

## SetLibraryDirectory

Sets the library directory path. The specified directory must already exist and contain a syslib folder.

<b>UI Access</b>	NA		
<b>Parameters</b>	Name <code>&lt;DirectoryPath&gt;</code>	Type String	Description The path to the SysLib Directory
<b>Return Value</b>	None		

<b>Python Syntax</b>	<code>SetLibraryDirectory (&lt;DirectoryPath&gt;)</code>
<b>Python Example</b>	<code>oDesktop.SetLibraryDirectory("c:\libraries")</code>

<b>VB Syntax</b>	<code>SetLibraryDirectory &lt;DirectoryPath&gt;</code>
<b>VB Example</b>	<code>oDesktop.SetLibraryDirectory"c:\libraries"</code>

## SetProjectDirectory

Sets the project directory path.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <code>&lt;DirectoryPath&gt;</code>	Type String	Description The path to the project directory. This should be writeable by the user.

<b>Return Value</b>	None.
---------------------	-------

<b>Python Syntax</b>	SetProjectDirectory (<DirectoryPath>)
<b>Python Example</b>	<code>oDesktop.SetProjectDirectory("c:\projects")</code>

<b>VB Syntax</b>	SetProjectDirectory <DirectoryPath>
<b>VB Example</b>	<code>oDesktop.SetProjectDirectory "c:\projects"</code>

## SetSchematicEnvironment

Sets the schematic environment for Electronics Desktop.

<b>UI Access</b>	Navigate to <b>Tools &gt; Options &gt; General Options &gt; General &gt; Desktop Configuration</b> . Select a schematic environment using the <b>Custom Menu Set</b> drop-down menu.								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;schEnv&gt;</td> <td>Integer</td> <td> Desired schematic environment. Can be one of:  <ul style="list-style-type: none"> <li>• 0 (Circuit)</li> <li>• 1 (Twin Builder)</li> <li>• 2 (Maxwell Circuit)</li> </ul> </td> </tr> </tbody> </table>			Name	Type	Description	<schEnv>	Integer	Desired schematic environment. Can be one of: <ul style="list-style-type: none"> <li>• 0 (Circuit)</li> <li>• 1 (Twin Builder)</li> <li>• 2 (Maxwell Circuit)</li> </ul>
Name	Type	Description							
<schEnv>	Integer	Desired schematic environment. Can be one of: <ul style="list-style-type: none"> <li>• 0 (Circuit)</li> <li>• 1 (Twin Builder)</li> <li>• 2 (Maxwell Circuit)</li> </ul>							
<b>Return Value</b>	None								

<b>Python Syntax</b>	SetSchematicEnvironment(<schEnv>)
<b>Python Example</b>	<code>oDesktop.SetSchematicEnvironment(1)</code>

<b>VB Syntax</b>	SetSchematicEnvironment(<schEnv>)
<b>VB Example</b>	<code>Set oProject = oDesktop.SetSchematicEnvironment 2</code>

## SetTempDirectory

Sets the temp directory path. The directory will be automatically created if it does not already exist.

<b>UI Access</b>	N/A	
<b>Parameters</b>	Name <i>&lt;DirectoryPath&gt;</i>	Type String Description The path to the Temp directory. This should be writeable by the user.
<b>Return Value</b>	None.	

<b>Python Syntax</b>	SetTempDirectory (<DirectoryPath>)
<b>Python Example</b>	<code>oDesktop.SetTempDirectory ("c:\\tmp")</code>

<b>VB Syntax</b>	SetTempDirectory <DirectoryPath>
------------------	----------------------------------

<b>VB Example</b>	<code>oDesktop.SetTempDirectory "c:\tmp"</code>
-------------------	---

## ShowDockingWindow

Shows or hides a docking window.

<b>UI Access</b>	Right click docking window > Show/Hide.		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;windowName&gt;</code>	String	The window name (for example, "Message Manager", "Component Libraries", "Properties")
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>ShowDockingWindow (&lt;windowName&gt;, &lt;show&gt;)</code>
<b>Python Example</b>	<code>oDesktop.ShowDockingWindow ('Message Manager', False)</code>

<b>VB Syntax</b>	<code>ShowDockingWindow &lt;windowName&gt; &lt;show&gt;</code>
<b>VB Example</b>	<code>oDesktop.ShowDockingWindow "Message Manager" False</code>

## Sleep

Suspends execution of HFSS for the specified number of milliseconds, up to 60,000 milliseconds (1 minute).

<b>UI Access</b>	NA
------------------	----

Parameters	Name	Type	Description
	<TimeInMilliseconds>	Integer	The time that the execution should be suspended in milliseconds
Return Value	None		

Python Syntax	Sleep (<TimeInMilliseconds>)
Python Example	<code>oDesktop.Sleep (1000)</code>

VB Syntax	Sleep <TimeInMilliseconds>
VB Example	<code>oDesktop.Sleep 1000</code>

## SubmitJob

Submits a batch job to a scheduler. When submitting the same project file multiple times, you should have the script wait for each job (or jobs for multi-step) to finish, which can be done via the monitoring functions LaunchJobMonitor() and RefreshJobMonitor(), checking the result of RefreshJobMonitor() in a loop until it returns completed ("Monitor Not Visible") status.

UI Access	<b>Tools &gt; Job Management &gt; Submit Job.</b>		
Parameters	Name	Type	Description
	<settingsPath>	String	Path to the settings file (exported from the Submit Job GUI) to use for submission.
	<projectPath>	String	Path to the project file to use in the batch job. This could be an archive (.aedtz file) or an un-archived project.

	<code>&lt;design (optional)&gt;</code>	String	Name of the design to use for batch solve.
	<code>&lt;setup (optional)&gt;</code>	String	Name of the specific setup to solve.
<b>Return Value</b>	Array of job ID strings (empty if no jobs submitted).		

<b>Python Syntax</b>	<code>SubmitJob(&lt;settingsPath&gt;, &lt;projectPath&gt;, &lt;design&gt;, &lt;setup&gt;)</code>
<b>Python Example</b>	<pre>jobIDs = oDesktop.SubmitJob ("C:\\hpc-settings\\Submit_ Job_Settings.areg", "C:\\projects\\basic.aedt"))  moreIDs = oDesktop.SubmitJob ("C:\\hpc-settings\\Submit_ Job_Settings.areg", "C:\\projects\\basic.aedt", "Design1", "Setup1")</pre>

<b>VB Syntax</b>	<code>SubmitJob &lt;settingsPath&gt;, &lt;projectPath&gt;, &lt;design&gt;, &lt;setup&gt;</code>
<b>VB Example</b>	<pre>jobIDs = oDesktop.SubmitJob "C:\\hpc-settings\\Submit_ Job_Settings.areg", "C:\\projects\\basic.aedt"  moreIDs = oDesktop.SubmitJob "C:\\hpc-settings\\Submit_ Job_Settings.areg", "C:\\projects\\basic.aedt", "Design1", "Setup1"</pre>

## TileWindows

Arrange all open windows in a tiled format.

UI Access	From main menu, <b>Window &gt;Tile Horizontally</b> or <b>Window &gt;Tile Vertically</b> .		
Parameters	Name	Type	Description
	< <i>TilingFlag</i> >	Integer	<ul style="list-style-type: none"><li>• 0 – Tile vertically.</li><li>• 1 – Tile horizontally.</li></ul>
Return Value	None.		

Python Syntax	TileWindows(< <i>TilingFlag</i> >)
Python Example	<code>oDesktop.TileWindows (0)</code>

VB Syntax	TileWindows < <i>TilingFlag</i> >
VB Example	<code>oDesktop.TileWindows 0</code>

## Desktop Commands For Registry Values

The Ansys Registry is stored as XML format file. By default it is located at C:\Users-<UserName>\Documents\Ansoft\<AnsysProductName>\config\<PC\_NAME>\_user.XML. Most of the Ansys product configuration information is stored in this XML file. These methods allow you to change the product configuration in VB or Python.

For example, to set the DSO & HPC analysis setup for HFSS using a Python script:

1. Start Maxwell.
2. Go to the DSO and HPC options and create a setup named "test".
3. Export the setup to a file (for example, c:\temp\test.acf).

4. Copy the exported file to a target PC (for example, f:\temp\test.acf).
5. Run the following script:

```
#import the setup

oDesktop.SetRegistryFromFile("f:\\temp\\\\test.acf")

# Set Active Setup to "test"

oDesktop.SetRegistryString("Desktop/ActiveDSOConfigurations/Maxwell", "test")
```

See the following subtopics:

[DeleteRegistryEntry](#)

[DoesRegistryValueExist](#)

[GetRegistryInt](#)

[GetRegistryString](#)

[SetRegistryFromFile](#)

[SetRegistryInt](#)

[SetRegistryString](#)

## DoesRegistryValueExist

Determines whether a registry value exists.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<KeyName>	String	Full name of registry key, including path.
<b>Return Value</b>	Boolean:		

- |  |  |
|--|--|
|  | <ul style="list-style-type: none"><li>• <b>True</b> – Key exists.</li><li>• <b>False</b> – Key does not exist.</li></ul> |
|--|--|

<b>Python Syntax</b>	DoesRegistryValueExist(<KeyName>)
<b>Python Example</b>	<pre>Exist = oDesktop.DoesRegistryValueExist('Desktop/ActiveDSOConfigurations/Maxwell')</pre>

<b>VB Syntax</b>	DoesRegistryValueExist(<KeyName>)
<b>VB Example</b>	<pre>bExist = oDesktop.DoesRegistryValueExist("Desktop/ActiveDSOConfigurations/Maxwell")</pre>

## GetRegistryInt

Obtains registry key integer value.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;KeyName&gt;</td><td>String</td><td>Full name of registry key, including path.</td></tr></tbody></table>			Name	Type	Description	<KeyName>	String	Full name of registry key, including path.
Name	Type	Description							
<KeyName>	String	Full name of registry key, including path.							
<b>Return Value</b>	Integer if the integer value is found. Return as Bad-Argument-Value if registry key does not exist or it is not an integer value.								

<b>Python</b>	GetRegistryInt(<KeyName>)
---------------	---------------------------

<b>Syntax</b>	
<b>Python Example</b>	<pre>num = oDesktop.GetRegistryInt('Desktop/Settings/ProjectOptions/Maxwell/UpdateReportsDynamicallyOnEdits')</pre>

<b>VB Syntax</b>	<code>GetRegistryInt(&lt;KeyName&gt;)</code>
<b>VB Example</b>	<pre>num = oDesktop.GetRegistryInt("Desktop/Settings/ProjectOptions/Maxwell/UpdateReportsDynamicallyOnEdits")</pre>

## GetRegistryString

Obtains registry key string value.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;KeyName&gt;</code></td> <td>String</td> <td>Full name of registry key, including path.</td> </tr> </tbody> </table>			Name	Type	Description	<code>&lt;KeyName&gt;</code>	String	Full name of registry key, including path.
Name	Type	Description							
<code>&lt;KeyName&gt;</code>	String	Full name of registry key, including path.							
<b>Return Value</b>	String if the string value is found. Return as Bad-Argument-Value if registry key does not exist or it is not a string value.								

<b>Python Syntax</b>	<code>GetRegistryString(&lt;KeyName&gt;)</code>
<b>Python Example</b>	<pre>activeDSO = oDesktop.GetRegistryString('Desktop/ActiveDSOConfigurations/Maxwell')</pre>

<b>VB Syntax</b>	GetRegistryString(<KeyName>)
<b>VB Example</b>	activeDSO = oDesktop.GetRegistryString("Desktop/ActiveDSOConfigurations/Maxwell")

## SetRegistryFromFile

Configures registry by specifying an Analysis Configuration file which must have been exported from the HPC and Analysis panel.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;filePath&gt;</td><td>String</td><td>Full file path of registry file.</td></tr></table>	Name	Type	Description	<filePath>	String	Full file path of registry file.
Name	Type	Description					
<filePath>	String	Full file path of registry file.					
<b>Return Value</b>	Success if configuration is imported. Bad argument value if the file is not found or does not contain valid analysis configuration data.						

<b>Python Syntax</b>	SetRegistryFromFile(<filePath>)
<b>Python Example</b>	oDesktop.SetRegistryFromFile('c:/temp/test.acf')

<b>VB Syntax</b>	SetRegistryFromFile <filePath>
<b>VB Example</b>	oDesktop.SetRegistryFromFile "c:/temp/test.acf"

## SetRegistryInt

Sets registry key to an integer value.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<KeyName>	String	Full name of registry key, including path.
<b>Return Value</b>	Success if the key is defined as an integer. Bad argument value if a key is not defined, or if the value is a text string.		

<b>Python Syntax</b>	SetRegistryInt(<KeyName>, <int>)
<b>Python Example</b>	<code>oDesktop.SetRegistryInt ('Desktop/Settings/ProjectOptions/Maxwell/UpdateReportsDynamicallyOnEdits', 0)</code>

<b>VB Syntax</b>	SetRegistryInt <KeyName> <int>
<b>VB Example</b>	<code>oDesktop.SetRegistryInt "Desktop/Settings/ProjectOptions/Maxwell/UpdateReportsDynamicallyOnEdits", 0</code>

## SetRegistryString

Sets registry key to a string value.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<KeyName>	String	Full name of registry key, including path.
	<value>	String	String value to be assigned to registry key.

<b>Return Value</b>	Success if the key is defined as a text string. Bad argument value if the key is not defined or requires an integer value.
---------------------	--

<b>Python Syntax</b>	<code>SetRegistryString(&lt;KeyName&gt;, &lt;value&gt;)</code>
<b>Python Example</b>	<code>oDesktop.SetRegistryString ('Desktop/ActiveDSOConfigurations/Maxwell', 'Local')</code>

<b>VB Syntax</b>	<code>SetRegistryString &lt;KeyName&gt;, &lt;value&gt;</code>
<b>VB Example</b>	<code>oDesktop.SetRegistryString "Desktop/ActiveDSOConfigurations/Maxwell", "Local"</code>

## 4 - Running Instances Manager Script Commands

The Running Instances Manager is a scripting object that lets you identify and connect to all running instances of Electronics Desktop. oDesktop objects that are returned provide full scripting functionality. Running Instances Manager commands should be executed by the oDesktop object. For example:

```
Set oRunningInstances = oDesktop.GetRunningInstancesMgr()
```

[GetAllRunningInstances](#)

[GetRunningInstanceByProcessID](#)

[GetRunningInstanceByProject](#)

### GetAllRunningInstances

Returns a list of running instances of Ansys Electronics Desktop.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array containing list of Ansys Electronics Desktop instances.

<b>Python Syntax</b>	GetAllRunningInstances()
<b>Python Example</b>	obj = oRunningInstances.GetAllRunningInstances()

<b>VB Syntax</b>	GetAllRunningInstances
<b>VB Example</b>	set obj = oRunningInstances.GetAllRunningInstances()

## GetRunningInstanceByProcessID

Returns the instance of Ansys Electronics Desktop that is running a specified process.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;processID&gt;</td><td>Integer</td><td>Process ID</td></tr></table>			Name	Type	Description	<processID>	Integer	Process ID
Name	Type	Description							
<processID>	Integer	Process ID							
<b>Return Value</b>	String of the returned instance.								

<b>Python Syntax</b>	GetRunningInstanceByProcessID(<processID>)
<b>Python Example</b>	obj = oRunningInstances.GetRunningInstanceByProcessID(12345)

<b>VB Syntax</b>	GetRunningInstanceByProcessID <processID>
<b>VB Example</b>	set obj = oRunningInstances.GetRunningInstanceByProcessID(12345)

## GetRunningInstancesMgr

Returns the object of the Running Instances Manager.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>None</td><td></td><td></td></tr></table>			Name	Type	Description	None		
Name	Type	Description							
None									

<b>Return Value</b>	Object Running instances manager object
---------------------	--

<b>Python Syntax</b>	GetRunningInstancesMgr()
<b>Python Example</b>	<code>oRunningInstances = oDesktop.GetRunningInstancesMgr()</code>

<b>VB Syntax</b>	GetRunningInstancesMgr()
<b>VB Example</b>	<code>Set oRunningInstances = oDesktop.GetRunningInstancesMgr()</code>

This page intentionally  
left blank.

## 5 - Project Object Script Commands

Project commands should be executed by the oProject object. One example of accessing this object is:

```
Set oProject = oDesktop.GetActiveProject()
```

AddMaterial

[AnalyzeAll](#)

[ClearMessages](#)

[Close](#)

[CopyDesign](#)

[CutDesign](#)

[DeleteDesign](#)

[DeleteToolObject](#)

[GetActiveDesign](#)

[GetChildNames \[Project\]](#)

[GetChildObject \[Project\]](#)

[GetChildTypes \[Project\]](#)

[GetConfigurableData](#)

[GetDefinitionManager](#)

[GetDependentFiles](#)

[GetDesign](#)

[GetEDBHandle](#)

[GetLegacyName](#)  
[GetName \[Project\]](#)  
[GetObjPath \[Project\]](#)  
[GetPath](#)  
[GetPropEvaluatedValue](#)  
[GetPropNames \[Project\]](#)  
[GetPropSIValue](#)  
[GetPropValue \[Project\]](#)  
[GetProperties](#)  
[GetPropertyValue](#)  
[GetTopDesignList](#)  
[InsertDesign](#)  
[InsertDesignWithWorkflow](#)  
[InsertToolObject](#)  
[Paste \[Project Object\]](#)  
[Redo \[Project Level\]](#)  
[Rename](#)  
[RunToolkit](#)  
[Save](#)  
[SaveAs](#)

---

[SaveAsStandAloneProject](#)

[SaveProjectArchive](#)

[SetActiveDefinitionEditor](#)

[SetActiveDesign](#)

[SetPropValue \[Project\]](#)

[SimulateAll](#)

[Undo \[Project\]](#)

[UpdateDefinitions](#)

## AnalyzeAll [project]

Runs the project-level script command from the script, which simulates all solution setups and Optimetrics setups for all design instances in the project. The UI waits until simulation is finished before continuing with the script.

<b>UI Access</b>	Project > Analyze All.
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	AnalyzeAll()
<b>Python Example</b>	<code>oProject.AnalyzeAll()</code>

<b>VB Syntax</b>	AnalyzeAll
------------------	------------

**VB Example**

```
oProject.AnalyzeAll
```

## ClearMessages

Clears information in the **Messages** window.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	None.

**Python Syntax**

```
ClearMessages()
```

**Python Example**

```
oProject.ClearMessages()
```

<b>VB Syntax</b>	ClearMessages
<b>VB Example</b>	oProject.ClearMessages

## Close

Closes the active project.

**Warning:**

Unsaved changes will be lost.

---

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	Close()
<b>Python Example</b>	<code>oProject.Close()</code>

<b>VB Syntax</b>	Close
<b>VB Example</b>	<code>oProject.Close</code>

## CopyDesign

Copies a specified design.

<b>UI Access</b>	Edit > Copy.						
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><i>&lt;DesignName&gt;</i></td> <td>String</td> <td>Name of the design to copy from.</td> </tr> </table>	Name	Type	Description	<i>&lt;DesignName&gt;</i>	String	Name of the design to copy from.
Name	Type	Description					
<i>&lt;DesignName&gt;</i>	String	Name of the design to copy from.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	CopyDesign (<DesignName>)
<b>Python Example</b>	<code>oProject.CopyDesign ("HFSSDesign1")</code>

<b>VB Syntax</b>	CopyDesign < <i>DesignName</i> >
<b>VB Example</b>	<code>oProject.CopyDesign "HFSSDesign1"</code>

## CutDesign

Cuts a design from the active project. The design is stored in memory and can be pasted.

### Warning:

This is a legacy command that is no longer supported and should not be used as it may have unintended effects on solved designs.

<b>UI Access</b>	Edit > Cut.						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;<i>DesignName</i>&gt;</td><td>String</td><td>Name of the design.</td></tr></table>	Name	Type	Description	< <i>DesignName</i> >	String	Name of the design.
Name	Type	Description					
< <i>DesignName</i> >	String	Name of the design.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	CutDesign (< <i>DesignName</i> >)
<b>Python Example</b>	<code>oProject.CutDesign("SimplorerDesign1")</code>

<b>VB Syntax</b>	CutDesign < <i>DesignName</i> >
<b>VB Example</b>	<code>oProject.CutDesign "SimplorerDesign1"</code>

## DeleteDesign

Deletes a specified design in the project.

<b>UI Access</b>	Edit > Delete, or Delete in the ribbon.						
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;<i>DesignName</i>&gt;</td> <td>String</td> <td>Name of the design.</td> </tr> </table>	Name	Type	Description	< <i>DesignName</i> >	String	Name of the design.
Name	Type	Description					
< <i>DesignName</i> >	String	Name of the design.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	DeleteDesign (< <i>DesignName</i> >)
<b>Python Example</b>	<code>oProject.DeleteDesign ("MaxwellDesign2")</code>

<b>VB Syntax</b>	DeleteDesign < <i>DesignName</i> >
<b>VB Example</b>	<code>oProject.DeleteDesign "MaxwellDesign2"</code>

## DeleteToolObject

### Note:

This command is for internal Ansys use only.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;ObjectName&gt;</td><td>String</td><td>Name of the tool object.</td></tr></tbody></table>			Name	Type	Description	<ObjectName>	String	Name of the tool object.
Name	Type	Description							
<ObjectName>	String	Name of the tool object.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	DeleteToolObject(<ObjectName>)
<b>Python Example</b>	<code>oProject.DeleteToolObject ("object1")</code>

<b>VB Syntax</b>	DeleteToolObject <ObjectName>
<b>VB Example</b>	<code>oProject.DeleteToolObject "object1"</code>

## GetActiveDesign

Returns the design in the active project

**Note:** GetActiveDesign will return normally if there are no active objects.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Object of the active design.

<b>Python Syntax</b>	GetActiveDesign()
<b>Python Example</b>	<pre>oDesign = oProject.GetActiveDesign()</pre>

<b>VB Syntax</b>	GetActiveDesign
<b>VB Example</b>	<pre>Set oDesign = oProject.GetActiveDesign</pre>

## GetConfigurableData (Project)

**Note:**

This command is for internal Ansys use only.

<b>Python Syntax</b>	GetConfigurableData()
<b>Python Example</b>	<pre>oProject.GetConfigurableData()</pre>

## GetDefinitionManager

Gets the `DefinitionManager` object.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	<code>DefinitionManager</code> object.

<b>Python Syntax</b>	GetDefinitionManager()
<b>Python Example</b>	<code>oDefinitionManager = oProject.GetDefinitionManager()</code>

<b>VB Syntax</b>	GetDefinitionManager
<b>VB Example</b>	<code>Set oDefinitionManager = oProject.GetDefinitionManager</code>

## GetDependentFiles

Provides a list of the external files referenced in the project, including characteristic (for example, MDX) and coupled project files.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	List of referenced files.

<b>Python Syntax</b>	GetDependentFiles()
<b>Python Example</b>	<code>files = oProject.GetDependentFiles()</code>

<b>VB Syntax</b>	GetDependentFiles
<b>VB Example</b>	<code>files = oProject.GetDependentFiles</code>

## GetDesign

*Use:* Returns the specified design.

*Command:* None

*Syntax:* GetDesign <DesignName>

*Return Value:* The specified design.

*Parameters:* <DesignName>

Type: <string>

Name of the design to return.

*VB Example:* Set oDesign = oProject.GetDesign ("Maxwell13DDesign1")

## GetEDBHandle

Returns the EDB handle for the project.

### Important:

This script is for internal Ansys use only.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String indicating the EDB handle for the project.

### Python Syntax

GetEDBHandle()

**Python Example**

```
oProject.GetEDBHandle()
```

**VB Syntax**

```
GetEDBHandle
```

**VB Example**

```
oProject.GetEDBHandle
```

## GetLegacyName

Obtains the legacy name of a project.

**Note:**

This command is for internal Ansys use only.

**UI Access**

N/A

**Parameters**

None.

**Return Value**

String containing the legacy project name.

**Python Syntax**

```
GetLegacyName()
```

**Python Example**

```
oProject.GetLegacyName()
```

**VB Syntax**

```
GetLegacyName
```

<b>VB Example</b>	<code>oProject.GetLegacyName</code>
-------------------	-------------------------------------

## GetName [Project]

Obtains the project name

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String containing the project name, not including the path or extension.

<b>Python Syntax</b>	<code>GetName()</code>
<b>Python Example</b>	<code>oProject.GetName ()</code>

<b>VB Syntax</b>	<code>GetName</code>
<b>VB Example</b>	<code>oProject.GetName</code>

## GetPath

Returns the location of the project on disk.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.

<b>Return Value</b>	String containing the path to the project, not including the project name.
---------------------	--

<b>Python Syntax</b>	GetPath()
<b>Python Example</b>	<code>oProject.GetPath()</code>

<b>VB Syntax</b>	GetPath
<b>VB Example</b>	<code>oProject.GetPath</code>

## GetTopDesignList

Returns a list of top-level design names.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	List of strings containing name of top-level designs.

<b>Python Syntax</b>	GetTopDesignList()
<b>Python Example</b>	<code>oProject.GetTopDesignList()</code>

<b>VB Syntax</b>	GetTopDesignList
<b>VB Example</b>	<code>oProject.GetTopDesignList</code>

## InsertDesign

Inserts a new design in the project. For Maxwell scripts, the last argument will always be empty.

UI Access	Project > [Insert Maxwell 3D Design / Insert Maxwell 2D Design / Insert RMxprt Design].		
<b>Parameters</b>	Name	Type	Description
	<i>&lt;DesignType&gt;</i>	String	Design type."Maxwell 2D", "Maxwell 3D", or "RMxprt".
	<i>&lt;DesignName&gt;</i>	String	Design name.
	<i>&lt;SolutionType&gt;</i>	String	Dependent on design type.  For Maxwell 3D, can be: "Magnetostatic", "EddyCurrent", "Transient", "Electrostatic", "DCConduction", "ElectroDCConduction", or "ElectricTransient".  For Maxwell 2D, can be: "Magnetostatic", "EddyCurrent", "Transient", "Electrostatic", "ACConduction", or "DCConduction".  For RMxprt, can be: "Three-Phase Induction Motor", "Single-Phase Induction Motor", "Three-Phase Synchronous Machine", "Brushless Permanent-Magnet DC Motor", "Adjust-Speed Synchronous Machine", "Permanent-Magnet DC Motor", "Switched Reluctance Motor", "Line-Start Permanent-Magnet Synchronous Motor", "Universal Motor", "DC Machine", "Claw-Pole Alternator", "Three-Phase Non-Salient Synchronous Machine", or "Generic Rotating Machine".
	""	None	Empty argument.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	InsertDesign (<DesignType>, <DesignName>, <SolutionType>, "")
<b>Python Example</b>	<pre>oProject.InsertDesign('Maxwell 3D', 'MyDesign', 'ElectricTransient', '')</pre>

<b>VB Syntax</b>	InsertDesign <DesignType>, <DesignName>, <SolutionType>, ""
<b>VB Example</b>	<pre>oProject.InsertDesign "Maxwell 3D", "MyDesign", "ElectricTransient", ""</pre>

## InsertDesignWithWorkflow

Inserts a design with a named workflow and returns an IDispatch string.

<b>UI Access</b>	N/A																					
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;type&gt;</td><td>String</td><td>Type of design.</td></tr><tr><td>&lt;workflowName&gt;</td><td>String</td><td>Name of the workflow.</td></tr><tr><td>&lt;specName&gt;</td><td>String</td><td>Name of the spec.</td></tr><tr><td>&lt;fileName&gt;</td><td>String</td><td>Name of the file.</td></tr><tr><td>&lt;libLoc&gt;</td><td>String</td><td>Type of library, such as SysLib.</td></tr><tr><td>&lt;stationaryPath&gt;</td><td>String</td><td>Path.</td></tr></tbody></table>	Name	Type	Description	<type>	String	Type of design.	<workflowName>	String	Name of the workflow.	<specName>	String	Name of the spec.	<fileName>	String	Name of the file.	<libLoc>	String	Type of library, such as SysLib.	<stationaryPath>	String	Path.
Name	Type	Description																				
<type>	String	Type of design.																				
<workflowName>	String	Name of the workflow.																				
<specName>	String	Name of the spec.																				
<fileName>	String	Name of the file.																				
<libLoc>	String	Type of library, such as SysLib.																				
<stationaryPath>	String	Path.																				
<b>Return Value</b>	IDispatch string, such as 'IDispatch(IAltraSimScript)'																					

<b>Python Syntax</b>	InsertDesignWithWorkflow(<type>, <workflowName>, <specName>, <fileName>, <libLoc>, <stationaryPath>)
----------------------	--

<b>Python Example</b>	<pre><code>oProject.InsertDesignWithWorkflow ("Circuit Design", "Serial Design", "PCIe3 Stressed", "LongChannel", "SysLib", "C:\Program Files\AnsysEM\v232\Win64\syslib\MS - RT_duroid 6010 (Er=10.2) 0.010 inch, 0.5 oz copper.asty")</code></pre>
-----------------------	---

<b>VB Syntax</b>	InsertDesignWithWorkflow <type>, <workflowName>, <specName>, <fileName>, <libLoc>, <stationaryPath>
<b>VB Example</b>	<pre><code>oProject.InsertDesignWithWorkflow "Circuit Design", "Serial Design", _ "PCIe3 Stressed", "LongChannel", "SysLib", _ "C:\Program Files\AnsysEM\v232\Win64\syslib\MS - RT_duroid 6010" &amp; _ " (Er=10.2) 0.010 inch, 0.5 oz copper.asty"</code></pre>

## InsertToolObject

**Note:**

This command is for internal Ansys use only.

<b>Python Syntax</b>	InsertToolObject()
<b>Python Example</b>	oProject.InsertToolObject()

## Paste (Project Object)

Pastes a design in the active project.

<b>UI Access</b>	<b>Edit &gt; Paste.</b>
<b>Parameters</b>	None.
<b>Return Value</b>	None

<b>Python Syntax</b>	Paste()
<b>Python Example</b>	<code>oProject.Paste()</code>

<b>VB Syntax</b>	Paste
<b>VB Example</b>	<code>oProject.Paste</code>

## Redo [Project Level]

Reapplies the last project-level command.

<b>UI Access</b>	<b>Edit &gt; Redo.</b>
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	Redo()
<b>Python Example</b>	<code>oProject.Redo ()</code>

---

<b>VB Syntax</b>	Redo
<b>VB Example</b>	<code>oProject.Redo</code>

## Rename

Renames the project and saves it. Similar to [SaveAs\(\)](#).

<b>UI Access</b>	<b>Edit &gt; Rename.</b>									
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><code>&lt;NewName&gt;</code></td> <td>String</td> <td>Desired name of the project. The path is optional.</td> </tr> <tr> <td><code>&lt;OverWriteOk&gt;</code></td> <td>Boolean</td> <td> <ul style="list-style-type: none"> <li>True - overwrite the file on disk if it exists.</li> <li>False - prevent overwrite.</li> </ul> </td> </tr> </table>	Name	Type	Description	<code>&lt;NewName&gt;</code>	String	Desired name of the project. The path is optional.	<code>&lt;OverWriteOk&gt;</code>	Boolean	<ul style="list-style-type: none"> <li>True - overwrite the file on disk if it exists.</li> <li>False - prevent overwrite.</li> </ul>
Name	Type	Description								
<code>&lt;NewName&gt;</code>	String	Desired name of the project. The path is optional.								
<code>&lt;OverWriteOk&gt;</code>	Boolean	<ul style="list-style-type: none"> <li>True - overwrite the file on disk if it exists.</li> <li>False - prevent overwrite.</li> </ul>								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	<code>Rename(&lt;NewName&gt;,&lt;OverWriteOK&gt;)</code>
<b>Python Example</b>	<code>oProject.Rename ("c:\projects\MyProject.aedt", True)</code>

<b>VB Syntax</b>	<code>Rename &lt;NewName&gt;,&lt;OverWriteOK&gt;</code>
<b>VB Example</b>	<code>oProject.Rename "c:\projects\MyProject.aedt", true</code>

## RestoreProjectArchive

Restores a previously archived project to a specified path.

UI Access	File > Restore Archive.		
Parameters	Name	Type	Description
	<ArchiveFilePath>	String	Path to archived file
	<ProjectFilePath>	String	Path to restore location
	<OverwriteExistingFiles>	Boolean	True to overwrite an existing file of the same name; else False.
	<OpenProjectAfterRestore>	Boolean	True to open the project after it is restored; else False.
Return Value	None.		

Python Syntax	RestoreProjectArchive (<Archivefilepath>, <ProjectFilePath>, <OverwriteExistingFiles>, <OpenProjectAfterRestore>)
Python Example	<pre>oDesktop.RestoreProjectArchive("C:\Users\jdoe\Documents\OptimTee.aedtz", "C:\Documents\OptimTee.aedt", False, True)</pre>

VB Syntax	RestoreProjectArchive <Archivefilepath>, <ProjectFilePath>, <OverwriteExistingFiles>, <OpenProjectAfterRestore>
VB Example	<pre>oDesktop.RestoreProjectArchive "C:\Users\jdoe\Documents\OptimTee.aedtz", _ "C:\Documents\OptimTee.aedt", false, true</pre>

## Save

Saves the active project.

<b>UI Access</b>	<b>File &gt; Save.</b>
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	Save()
<b>Python Example</b>	<code>oProject.Save()</code>

<b>VB Syntax</b>	Save
<b>VB Example</b>	<code>oProject.Save</code>

## SaveAs

Saves the project under a new name. Requires a full path.

### Note:

This script takes two parameters for non-schematic/layout designs and four parameters for schematic/layout designs.

<b>UI Access</b>	<b>File &gt; Save As.</b>
------------------	---------------------------

Parameters	Name	Type	Description
	<NewName>	String	The desired name of the project, with directory and extension.
	<OverWriteOK>	Boolean	True to overwrite the file of the same name, if it exists. False to prevent over-write.
	<DefaultAction>	String	For Schematic/Layout projects only. Otherwise omit. See note below.  Valid actions: ef_overwrite , ef_copy_no_overwrite, ef_make_path_absolute, or empty string.
	<OverwriteActions>	Array	For Schematic/Layout projects only. Otherwise omit. See note below.  Structured array: Array("Name: <Action>", <FileName>, <FileName>, ...)  Valid actions: ef_overwrite , ef_copy_no_overwrite, ef_make_path_absolute, or empty string.
Return Value	None.		

Python Syntax	For non-Schematic/Layout project: SaveAs ( <NewName> <OverWriteOK>)  For Schematic/Layout project: SaveAs (<NewName> <OverWriteOK> <DefaultAction> <OverrideActions>)
Python Example	<pre>oProject.SaveAs ('D:/projects/project1.aedt', True) ----- oProject.SaveAs ('D:/Projects/Project1.aedt', True, 'ef_overwrite', ['NAME:OverrideActions', ['NAME:ef_copy_no_overwrite', ['NAME:Files', '\$PROJECTDIR/circuit_models.inc']], ['NAME:ef_make_path_absolute', ['NAME:Files', '\$PROJECTDIR/SL_6s.sp']]])</pre>

<b>VB Syntax</b>	<p>For non-Schematic/Layout project: SaveAs &lt;NewName&gt; &lt;OverWriteOK&gt;</p> <p>For Schematic/Layout project: SaveAs &lt;NewName&gt; &lt;OverWriteOK&gt; &lt;DefaultAction&gt; &lt;OverrideActions&gt;</p>
<b>VB Examples</b>	<pre> oProject.SaveAs "D:/projects/project1.aedt", true ---  oProject.SaveAs     "F:\Designer Projects\TA33097\HighSpeedChannel.aedt", true, "ef_overwrite", Array     ("NAME:OverrideActions",         Array("NAME:ef_copy_no_overwrite", Array("NAME:Files", "\$PROJECTDIR/circuit_mod-         els.inc")),         Array("NAME:ef_make_path_absolute", Array("NAME:Files", "\$PROJECTDIR\SL_6s.sp"))) </pre>

### Important:

The DefaultAction and OverrideActions strings correspond to the following actions:

- **ef\_overwrite** – Copy file to new project directory and overwrite.
- **ef\_copy\_no\_overwrite** – Copy file to new project directory and don't overwrite.
- **ef\_make\_path\_absolute** – Change reference to point to file in old project directory.
- **Empty String** – Do nothing.

The DefaultAction is applied to all files that are NOT explicitly listed in the OverrideActions array. Those in the OverrideActions array are separate arrays for actions that are different from the default action; those actions are applied to the files listed in the same array:

- If OverrideActions are not specified, DefaultAction is applied to ALL files in project directory.

## SaveAsStandAloneProject

Saves the project as a standalone copy.

**Note:**

This script is not supported when the application is being controlled by Ansys Workbench.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;projectName&gt;</i>	Type String	Description The desired name of the project, with directory and extension.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	SaveAsStandAloneProject( <i>&lt;projectName&gt;</i> )
<b>Python Example</b>	<code>oProject.SaveAsStandAloneProject('D:/projects/project1.aedt')</code>

<b>VB Syntax</b>	SaveAsStandAloneProject <i>&lt;projectName&gt;</i>
<b>VB Examples</b>	<code>oProject.SaveAsStandAloneProject "D:/projects/project1.aedt"</code>

## SaveProjectArchive

Saves the active project as an archive to the specified file path.

UI Access	File > Archive.		
Parameters	Name	Type	Description
	<archiveFilePath>	String	Path to archived file.
	<IncludeExternalFiles>	Boolean	True to include external files; False to exclude.
	<IncludeResultsFiles>	Boolean	True to include simulation files associated with the project; False to exclude.
	<AdditionalFiles>	Array	Additional specified files to include.
Return Value	<ArchiveNotes>		
	String describe the archive.		
Return Value		None.	

Python Syntax	SaveProjectArchive(<archivefilepath>, <IncludeExternalFiles>, <IncludeResultsFiles>, <AdditionalFiles>, <ArchiveNotes>)
Python Example	oProject.SaveProjectArchive("C:\\\\Users\\\\Documents\\\\Ansoft\\\\Project27.aedtz", True, False, [], "")

VB Syntax	SaveProjectArchive <archivefilepath>, <IncludeExternalFiles>, <IncludeResultsFiles>, <AdditionalFiles>, <ArchiveNotes>
VB Example	oProject.SaveProjectArchive "C:\\Documents\\OptimTee.aedtz", true, false, Array(), "My notes"

## SetActiveDefinitionEditor

Obtains a specified definition editor.

UI Access	N/A
-----------	-----

<b>Parameters</b>	Name	Type	Description
	<EditorName>	String	Name of the definition editor to set active, one of "SymbolEditor", "FootprintEditor".
		<DefinitionName>	String The combination name for the symbol or footprint, <libname>:<def-name>
<b>Return Value</b>	Object for the definition to be edited.		

<b>Python Syntax</b>	SetActiveDefinitionEditor(<EditorName>, <DefinitionName>)
<b>Python Example</b>	<pre> oProject.SetActiveDefinitionEditor("SymbolEditor", "Simpler Elements\Basic Elements\Circuit\Passive Elements:R") </pre>

<b>VB Syntax</b>	SetActiveDefinitionEditor <EditorName>, <DefinitionName>
<b>VB Example</b>	<pre> oProject.SetActiveDefinitionEditor "SymbolEditor", "Simpler Elements\Basic Elements\Circuit\Passive Elements:R" </pre>

## SetActiveDesign

Sets a design to be the active design.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;DesignName&gt;</td> <td>String</td> <td>Name of the design to set as the active design.</td> </tr> </tbody> </table>			Name	Type	Description	<DesignName>	String	Name of the design to set as the active design.
Name	Type	Description							
<DesignName>	String	Name of the design to set as the active design.							

<b>Return Value</b>	None.
---------------------	-------

<b>Python Syntax</b>	<code>SetActiveDesign (&lt;DesignName&gt;)</code>
<b>Python Example</b>	<code>oDesign = oProject.SetActiveDesign ("SimplorerDesign2")</code>

<b>VB Syntax</b>	<code>SetActiveDesign &lt;DesignName&gt;</code>
<b>VB Example</b>	<code>Set oDesign = oProject.SetActiveDesign "SimplorerDesign2"</code>

## SimulateAll

Simulates all solution setups and Optimetrics setups for all design instances in the project. Script processing only continues when all analyses are finished.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	<code>SimulateAll()</code>
<b>Python Example</b>	<code>oProject.SimulateAll ()</code>

<b>VB Syntax</b>	<code>SimulateAll</code>
------------------	--------------------------

**VB Example**

```
oProject.SimulateAll
```

## Undo [Project]

Cancels the last project-level command.

<b>UI Access</b>	Edit > Undo.
<b>Parameters</b>	None.
<b>Return Value</b>	None.

**Python Syntax**

```
Undo()
```

**Python Example**

```
oProject.Undo ()
```

**VB Syntax**

```
Undo
```

**VB Example**

```
oProject.Undo
```

## UpdateDefinitions

Updates all definitions. The **Messages** window reports when definitions are updated, or warns when definitions cannot be found.

**UI Access**

Tools > Project Tools > Update Definitions. Click **Select All**, then **Update**.

<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	UpdateDefinitions()
<b>Python Example</b>	<code>oProject.UpdateDefinitions ()</code>

<b>VB Syntax</b>	UpdateDefinitions
<b>VB Example</b>	<code>oProject.UpdateDefinitions</code>

This page intentionally  
left blank.

# 6 - Object Oriented Property Scripting

Scripting in AEDT has been markedly enhanced via the convenient use of Object-Oriented access to retrieve or modify properties of objects in AEDT. This feature allows for much less code to be written to access object properties and enables much more readable code for our users, avoiding complex array input.

The primary gains of the use of Object-Oriented scripting are the ease with which properties of various existing objects in an Ansys Electronics DesktopProject/Design can be read, modified, and set. Along with this ease of implementation comes much more 'readable' code to aid in others' interpretation of custom scripts.

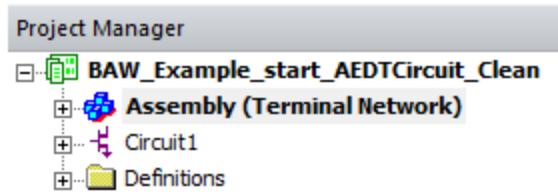
This App Note will discuss the following: The logic for syntax of access, Basic Attributes of Project and Design properties and examples of use.

## Object-Oriented Scripting

There are five basic functions that you use in Object Oriented scripting to retrieve and set properties:

1. GetChildNames()
2. GetChildObjects()
3. GetPropNames()
4. GetPropValue()
5. SetPropValue()

At a high level, use GetChildNames() to determine what object instances exist for a given object. An example is shown below to demonstrate for an AEDT Project shown that has two Designs.



If you open the Command Window that allows for executing python code, you first define the Project Object, oProject, and the Design Object, oDesign, as shown:

```
>>> oProject = oDesktop.GetActiveProject()
>>> oDesign = oProject.GetActiveDesign()
```

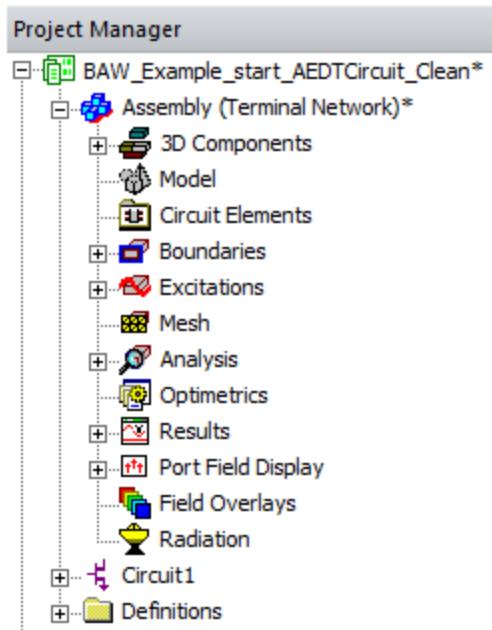
Once the objects have been defined, you can use GetChildNames() to learn what object instances exist for each. As an example, observe the Child Names of oProject, and you see a list of the Designs in the AEDT Project, per the GUI.

```
>>> oProject.GetChildNames()
['Assembly', 'Circuit1']
```

As another example, retrieve the names of the Object Instances available in oDesign, to see the various objects associated with a Design setup:

```
>>> oDesign.GetChildNames()
['Boundaries', 'Excitations', 'Circuit Elements', 'Model', 'Mesh', 'Analysis', 'Optimetrics', 'Port
Field Display', 'Field Overlays', 'Radiation', 'Results', '3D Modeler']
```

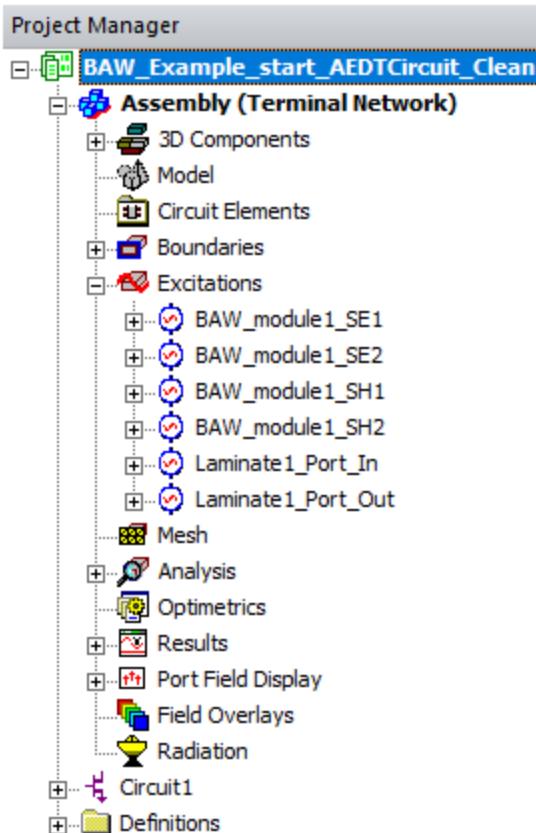
These names are what you would expect based on the Project Manager Layout:



In the **Project Manager** above, any object that has the '+' symbol is populated with children that you can query. Once you know the name of the object you want, you can instance it via the `GetChildObject()` command. This defines the instance to the desired object. As an example, set an instance for the Excitations:

```
>>> oExcitations = oDesign.GetChildObject('Excitations')
```

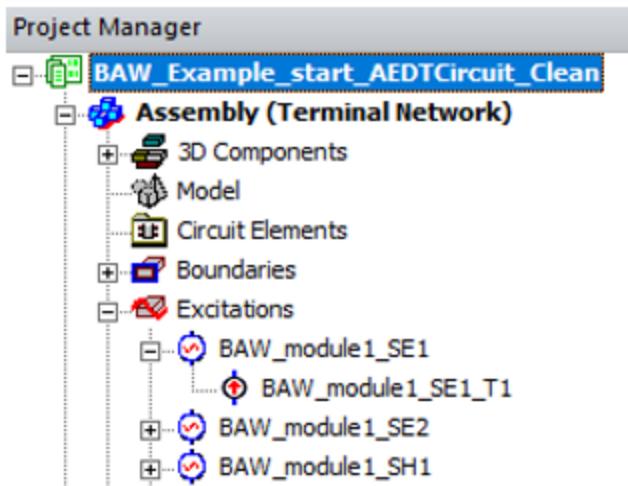
You now have an object, oExcitations defined to be the oDesign Child Object 'Excitations'. What does this mean? If you expand the Excitations dialogue in the Project Manager, you expect that the Child Names of this object would be the names of the Excitations as defined, in this case six ports:



```
>>> oExcitations.GetChildNames()
['Laminate1_Port_In', 'Laminate1_Port_Out', 'BAW_module1_SE1', 'BAW_module1_SE2', 'BAW_module1_SH1',
 'BAW module1 SH2']
```

There is clear logic to the Object Child Names as the children of the oExcitations object as the ports that have been defined in HFSS. Looking at the Project tree can help you to conceive and retrieve desired information.

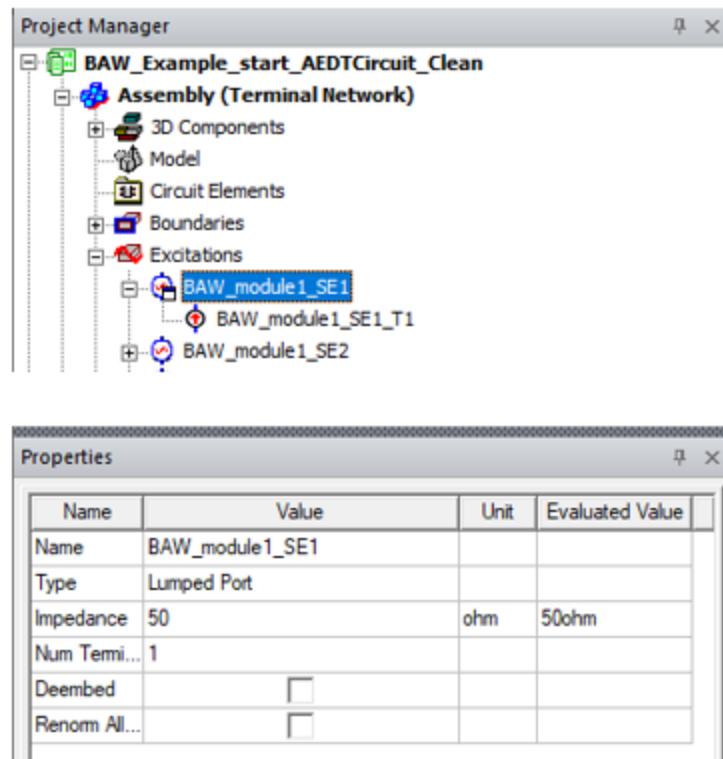
Expand the first port to see its expected Child Object, its Terminal, in the **Project Manager** Window:



Through scripting, first define the Port Object (in this example, oPort) using the 'GetChildObject()' command for the first port, 'BAW\_module1\_SE1.' Then determine its Object Child Name, the terminal definition:

```
>>> oPort = oExcitations.GetChildObject('BAW_module1_SE1')
>>> oPort.GetChildNames()
['BAW_module1_SE1_T1']
```

As the use and logic for GetChildNames() and GetChildObject() have been demonstrated, you can now explore the properties of each of these objects, if they exist. The function to determine what properties exist is GetPropNames(). Use this to determine what properties exist to be retrieved or modified for a given object. The properties available are readily identifiable in the **Property** window, by default located beneath the **Project Manager** window. For example, if you select a given port object, 'BAW\_module1\_SE1' the Property window populates as shown:



If you execute the GetPropNames() function on the previously defined object, oPort, you see the same Property Names as available in the **Properties** window:

```
>>> oPort.GetPropNames()
['Name', 'Type', 'Impedance', 'Num Terminals', 'Deembed', 'Renorm All Terminals']
```

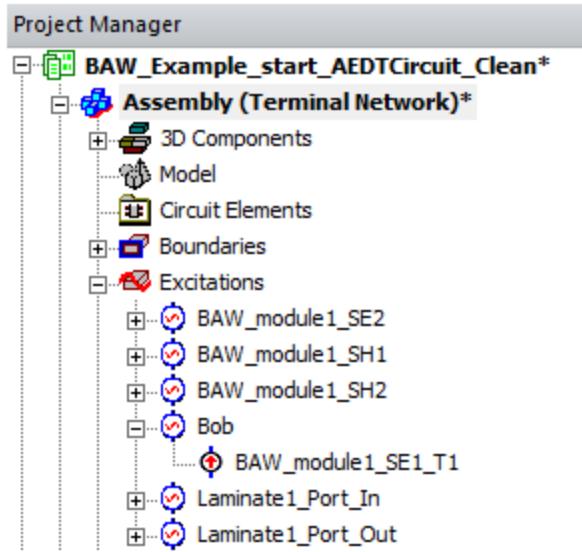
Once you identify the desired object and you know the desired property, you can access the value via GetPropValue(). For example, if you want to retrieve the name of the object oPort:

```
>>> oPort.GetPropValue('Name')
'BAW_module1_SE1'
```

To change the value of the property, use the SetPropValue() function. The arguments for this function are ('Property Name', 'New Value'). For example, to change the name of the port to 'Bob':

```
>>> oPort.SetPropValue('Name', 'Bob')
True
```

This function returns a Boolean 'True' if successful. The Project Manager window updates accordingly:



This approach for retrieving and setting properties is general and can be used for many aspects of an Ansys Electronics Desktop simulation. This Object-Oriented method of property identification and modification operates only on existing objects. Object-Oriented scripting cannot create new instances; you must revert to the functions in a given Module to do that. Not all Children of a given object may be accessible via the GetChildNames() command just yet. An example is given for Material property modification later in this App Note. However, if you need specific objects you can reference details in the Scripting Help or reach out to an Application Engineer.

## Material Properties and Examples

This section discusses the material properties and how to access and modify them. Because materials are globally defined, the objects are children of the Project, oProject, as shown below:

```
>>> oProject = oDesktop.GetActiveProject()
>>> oMaterials = oProject.GetChildObject('Materials')
>>> oMaterials.GetChildNames()
['vacuum', 'Cap_Mat', 'Outline_Mat', 'SolderMask_Mat', 'copper', 'pec']
```

All materials with a Project Definition, or assigned to an object, in the Project are accessible. For example, assume you want to see the conductivity of 'copper.' Follow the same flow as in the previous section:

```
>>> oCopper = oMaterials.GetChildObject('copper')
>>> oCopper.GetPropNames()
['Coordinate System Type', 'Coordinate System Type/Choices', 'Relative Permittivity Type', 'Relative Permittivity Type/Choices', 'Relative Permittivity', 'Relative Permeability Type', 'Relative Permeability Type/Choices', 'Relative Permeability', 'Bulk Conductivity Type', 'Bulk Conductivity Type/Choices', 'Bulk Conductivity', 'Dielectric Loss Tangent Type', 'Dielectric Loss Tangent Type/Choices', 'Dielectric Loss Tangent', 'Magnetic Loss Tangent Type', 'Magnetic Loss Tangent Type/Choices', 'Magnetic Loss Tangent', 'Electric Coercivity Type', 'Electric Coercivity Magnitude', 'Magnetic Coercivity Type', 'Magnetic Coercivity Magnitude', 'Thermal Conductivity Type', 'Thermal Conductivity Type/Choices', 'Thermal Conductivity', 'Magnetic Saturation Type', 'Magnetic Saturation', 'Lande G Factor Type', 'Lande G Factor', 'Delta H Type', 'Delta H', '- Measured Frequency Type', '- Measured Frequency', 'Core Loss Model', 'Core Loss Model/Choices', 'Mass Density Type', 'Mass Density', 'Composition', 'Composition/Choices', 'Specific Heat Type', 'Specific Heat', 'Young's Modulus Type', 'Young's Modulus Type/Choices', "Young's Modulus", "Poisson's Ratio Type", "Poisson's Ratio Type/Choices", "Poisson's Ratio", 'Thermal Expansion Coefficient Type', 'Thermal Expansion Coefficient Type/Choices', 'Thermal Expansion Coefficient', 'Magnetostriction Type', 'Inverse Magnetostriction Type', 'Thermal Material Type', 'Thermal Material Type/Choices', 'Solar Behavior Type', 'Solar Behavior Type/Choices', 'Solar Behavior']
```

The Material Property of interest is "Bulk Conductivity." So you create a "Cond" object to store the value and use the GetPropValue function to obtain it. Then name the Cond object to see the value:

```
>>> Cond = oCopper.GetPropValue('Bulk Conductivity')
>>> Cond
'58000000'
```

To change the conductivity, use the SetPropValue() function as shown below:

```
>>> oCopper.SetPropValue('Bulk Conductivity', '100')
True
>>> NewCond = oCopper.GetPropValue('Bulk Conductivity')
>>> NewCond
'100'
```

## Body Properties and Modification

The following example shows how to retrieve the properties of a Body in the model, in this case a Region object. Once you identify the desired property, you can modify it as needed.

```
>>> oModel = oDesign.GetChildObject('3D Modeler')
>>> oModel.GetChildNames()
['RadBox_Region_1']
>>> oRegion = oModel.GetChildObject('RadBox_Region_1')
>>> oRegion.GetPropNames()
['Name', 'Material', 'Solve Inside', 'Orientation', 'Orientation/Choices', 'Model', 'Group', 'Display
Wireframe', 'Material Appearance', 'Color', 'Color/Red', 'Color/Green', 'Color/Blue', 'Transparent']
```

## Retrieving Variables

Retrieving defined variables in a Design or Project is a common effort for automation. There are two types of variables, Design and Project. Project variables are preceded with a '\$' symbol and are retrieved in the Project object as it is globally defined to all Designs. Design variables do not have any preceding symbols and are retrieved in the Design object as their scope is limited to a given Design. The following example demonstrates the retrieval of Project Variable names and values, and then Design variable names and values.:

```
>>> oProjVar = oProject.GetChildObject("Variables")
>>> oProjVar.GetPropNames()
 ['$test']
>>> oProjVar.GetPropValue('$test')
 '0'
>>> oDesVar = oDesign.GetChildObject("Variables")
>>> oDesVar.GetPropNames()
 ['test']
>>> oDesVar.GetPropValue('test')
 '0'
```

## Retrieve Datasets and Values

The GetChildObject, GetChildTypes and GetChildNames functions operate on the oDesktop objects. This allows you to retrieve and view datasets and values. The dataset script wrapper store all values internally in SI units, and converts them back to user-supplied units when you request non-SI property values. For example, if you assigned a dataset to the example OptimTee project in HFSS, you could use these functions in the command window:

```
>>>oDesktop.GetChildTypes()
 ['Projects']

>>>oDesktop.GetChildNames()
 ['OptimTee']

>>>arrProjectNames = oDesktop.GetChildNames()

>>>tp = oDesktop.GetChildObject('OptimTee')

>>>tp.GetChildTypes()
```

```
['Design', 'Project Data']

>>>tp.GetChildNames('Project Data')

['Variables', 'Materials', 'Surface Materials', 'Datasets']

>>>ds = tp.GetChildObject('datasets')

>>>ds.GetChildNames()

['$ds1']

>>>ds1=ds.GetChildObject('$ds1')

>>>ds1.GetPropValue('[:, :, :]')

[[1.0, 4.0], [2.0, 5.0], [3.0, 6.0]]

>>>ds1.GetPropSIValue()

[[1.0, 4.0], [2.0, 5.0], [3.0, 6.0]]

>>>ds1.DimUnits

>>>ds1.DimUnits = ['mm', 'mm']

>>>ds1.DimUnits

['mm', 'mm']

>>>ds1.GetPropSIValue()

[[0.001, 0.004000000000000001], [0.002, 0.005000000000000001], [0.003000000000000001,
0.006000000000000001]]

>>>ds1.GetPropValue('[:, :, :]')

[[1.0, 4.0], [2.0, 5.0], [3.0, 6.0]]
```

## GetSolutionData API

Many users want to use scripts to extract solution data from Ansys Electronics Desktop for custom Post Processing. Scripting includes a new method to do this without having to export data to a file and then re-import it for use in a script. The new function is accessible via the “ReportSetup” Module. The function call is “GetSolutionDataPerVariation()”. A code snippet to extract Terminal S Parameter data is shown:

```
8 oModule = oDesign.GetModule("ReportSetup")
9 Results = oModule.GetSolutionDataPerVariation("Terminal.Solution.Data", "Setup1::Sweep1",
10 [
11   [
12     [
13       [
14         [
15           [
16             [
17               [
18                 [
19                   ## Get Dependent and Independent Variable data for Nominal Variation
20                   NominalData = Results[0]
21                   ## Get Independent Variable Data
22                   ## For second argument, if pass=True then data is in SI Units
23                   ## if pass=False then data is in default scale.units instead of SI
24                   SweepValues = NominalData.GetSweepValues("Freq", True)
25                   ## Get Dependent Variable DataValues
26                   ## Note: Can pass any 'Y Component' Name
27                   DataValues = NominalData.GetRealDataValues("dB(St(Terminal_1))")
```

The above code shows how you can extract the Dependent and Independent data to variables for easy manipulation. For more information on other functions available for this, see [GetSolutionDataPerVariation](#).

## Summary

Scripting has been advancing in Ansys Electronics Desktop to better allow you to customize and automate their repetitive or complex simulations. The ability to easily retrieve and set property values via the Object-Oriented scripting allows for ease or both writing and

reading. The ability to extract solution data within a script execution is a new functionality that markedly enables more advanced post processing.

Object oriented property scripting presents an easy to use, intuitive and object oriented representation of the data model. The framework supports query of objects and their properties including the edits of the data model in an object oriented fashion. With the new scripting framework, data exposure is intuitive and provides maximum coverage.

Each exposed script object supports the following COM functions:

- GetName
  - Return name of the object as text string
  - e.g. name of a design, solve setup, boundary, etc
- GetChildTypes
  - An object can have different types of children.
  - Return array of text string. Can be empty if the object's children are NOT categorized into different types.

For example, a design object has 3 children types. The following examples show how the commands run in the **Tools > Open Command Window** for IronPython.

```
>>> design.GetChildTypes()  
['Module', 'Editor', 'Design Data']
```

- GetChildNames
  - Input: [String – Type]. Default = “Module” and “Editor” for design script object. ‘All’ for other script objects.
  - Return an array of immediate children’s names, of a given type if specified

For example, a Mechanical design object has these children.

```
>>> design.GetChildNames()  
['Boundaries', 'Excitations', 'Optimetrics', 'Results', '3D Modeler']
```

Four of the children are of “Module” type

```
>>> design.GetChildNames("module")
['Boundaries', 'Excitations', 'Optimetrics', 'Results']
```

- **GetChildObject**
  - Input: String -- Object path. The path may include multiple generations.
  - Return the child object if found

For example,

```
>>> d = project.GetChildObject("hfss")
>>> d.GetChildObject("3d modeler").GetChildNames()
['Box1', 'Box1_1', 'Box1_1_1']
>>> project.GetChildObject("hfss/3d modeler").GetChildNames()
['Box1', 'Box1_1', 'Box1_1_1']
```

- **GetPropNames**
  - Input: [BOOL - IncludeReadOnly] -- default to true
  - Return an array of the object's properties

For example,

```
>>> geom = project.GetChildObject("hfss/3d modeler").GetChildObject("Box1")
>>> geom.GetPropNames()
['Name', 'Material', 'Material/SIValue', 'Material/EvaluatedValue', 'Solve Inside', 'Orientation',
 'Orientation/Choices', 'Model', 'Group', 'Display Wireframe', 'Material Appearance',
 'Color', 'Color/Red', 'Color/Green', 'Color/Blue', 'Transparent']
```

- GetPropValue
  - Input: String – Property Path. The path may include multiple generations.
  - Return the property value as VARIANT

For example,

```
>>> geom.GetPropValue("material")
'"vacuum"'
>>> geom.GetPropValue("xsize")
'3mm'
>>> op.GetPropValue("attach to original object")
False
```

- SetPropValue
  - Input: String – Property Path. The path may include multiple generations.
  - Input: String – Data. New value of the property.
  - Return -- True if property data is updated successfully. False if failed to assign the new value.

For example,

```
>>> geom.SetPropValue("model", False)
>>> boxcmd.SetPropValue("ysize", "4mm")
```

- GetPropEvaluatedValue (<PropName>)

For example,

```
oVar = oDesign.GetChildObject(" Variables/var")
oVar.GetPropEvaluatedValue()
```

- GetPropSIValue (<PropName>)

For example,

```
oCreateBox = oDesign.GetChildObject("3D Modeler/Box1/CreateBox:1")
oCreateBox.GetPropValue("xSize")
    return "length / 2"
oCreateBox.GetPropEvaluatedValue("xSize")
    return '0.4mm'
oCreateBox.GetPropSIValue("xSize")
    return 0.0004
```

## Additional Details Specific to AEDT Solvers

“3D Modeler” of 3D products and “Machine” of RMxprt are exposed as “Editor” type children of a design script object.

“Variables” and “Design Settings” are exposed as “Design Data” type children of a design script object.

The following “Module” types are exposed as “Module” type children of a design script object.

HFSS

- Boundaries, Excitations, Circuit Elements, Hybrid Regions, Analysis, Radiation, Field Overlays, Optimetrics, Results

HFSS 3D Layout

- Boundaries, Excitations, Circuit Elements, Analysis, Radiation, Field Overlays, Optimetrics, Results

Maxwell 3D/2D

- Boundaries, Excitations, Analysis, Field Overlays, Optimetrics, Results

RMxprt

- Analysis, Field Overlays, Optimetrics, Results

Q3D

- Boundaries, Nets, Analysis, Optimetrics,

Q2D

- Boundaries, Conductors, Analysis, Field Overlays, Optimetrics,

Icepak

- Thermal, Monitor, Mesh, Analysis, Field Overlays, Optimetrics, Results

Mechanical

- Boundaries, Excitations, Analysis, Field Overlays, Optimetrics, Results

Circuit

- Optimetrics, Results

Circuit Netlist

- Results

EMIT

- Coupling

Simplorer/Twin Builder

- Analysis, Optimetrics, Results

## Additional details on Boundaries/Excitations

Each design type presents its boundaries/excitations data in the project tree as different groups. For example, an HFSS design has Boundaries, Excitations, Circuit Elements and Hybrid Regions while a Icepak design has just a “Thermal” project tree folder.

These module script objects do not have properties

```
>>> project.GetChildObject("icepak/thermal").GetPropNames()  
[]
```

GetChildTypes of these module script objects returns the types of its immediate children

```
>>> d = p.GetChildObject("q2d")  
>>> d.GetChildObject("conductors").GetChildTypes()  
['NonIdealGround', 'SignalLine']
```

GetChildNames of these module object returns its immediate children

```
>>> p.GetChildObject("icepak/thermal").GetChildNames()  
['Source1', 'Resistance1', 'ConductingPlate1', 'Source2', 'Resistance2', 'ConductingPlate2',  
'Source3', 'Resistance3', 'ConductingPlate3']
```

GetChildNames can be invoked with a “type” and the returns will be filtered by that given type.

```
>>> p.GetChildObject("icepak/thermal").GetChildNames("resistance")  
['Resistance1', 'Resistance2', 'Resistance3']
```

Children of a module object are scriptable objects and have properties.

```
>>> port = p.GetChildObject("hfss/excitations/1")  
>>> port.GetPropNames(False)  
['Name', 'Deembed', 'Deembed Dist', 'Renorm All Terminals']
```

---

You can query/edit these properties

```
>>> port.GetPropValue("deembed")
False
>>> port.SetPropValue("deembed", True)
True
>>> port.GetPropValue("deembed")
True
```

A boundary/excitation script object can also have children. For example, HFSS terminal is a child of its port. Q3D source/sink can be children of a net.

```
>>> port.GetChildNames()
['Box1_T1']
>>> port.GetChildTypes()
['Terminal']
>>> p.GetChildObject("q3d/nets/s2").GetChildNames()
['Source2', 'Sink2']
>>> p.GetChildObject("q3d/nets/s2").GetChildTypes()
['Sink', 'Source']
```

## 3D component encapsulation

These script interfaces are compliant with encapsulation. For example,

- Design.GetChildObject("boundaries").GetChildNames() will not return component boundaries
- SetPropValues of component excitations can only be used to edit post processing settings such as 'Deembed', 'Deembed Dist' of a HFSS port.

## Additional details on Solve setup

All solve setups are children of the “Analysis” script object. This parent script object is also of the type “Module”.

```
>>> d = oDesktop.GetActiveProject().GetChildObject("hfss")
>>> d.GetChildNames()
['Boundaries', 'Excitations', 'Hybrid Regions', 'Circuit Elements', 'Analysis', 'Opti-
metrics', 'RadField', 'Results', '3D Modeler']
>>> setups = d.GetChildObject("analysis")
```

This module script object has no property

```
>>> setups.GetPropNames()
[]
```

The children of this module script object in a 3D design is not categorized into different types because the solve setup type is one-to-one to the solution type of a 3D design.

```
>>> setups.GetChildTypes()
[]
```

The children of this module script object in a 3DLayout and Simplorer/TwinBuilder design is categorized into different solve setup types, such as “Transient”, “AC” and “DC” in a Simplorer/TwinBuilder design and “HFSS”, “PlanarEM”, “SIwave” in a 3D layout design.

A solve setup script object can also have children. Children are typically frequency sweeps.

```
>>> setup.GetChildNames()
```

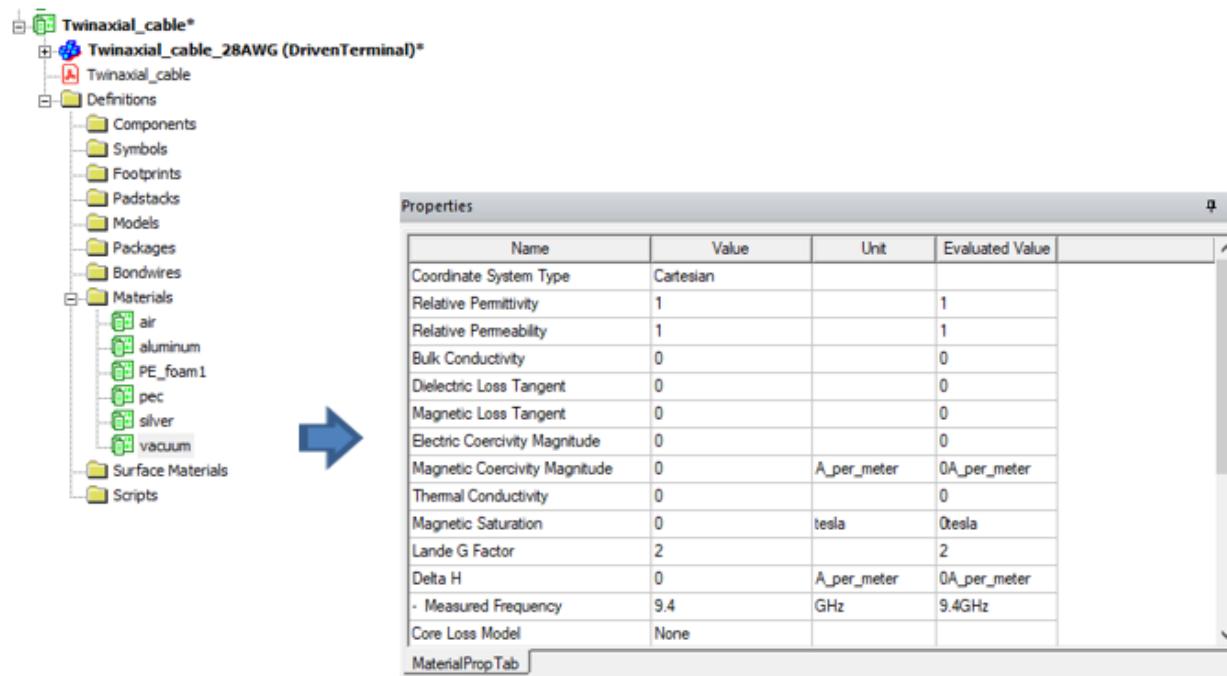
```
['Sweep', 'Sweep1', 'Sweep2']
>>> setup.GetChildTypes()
['Discrete', 'Interpolating']
>>> sweep1 = setup.GetChildObject("sweep")
```

### Related Topics

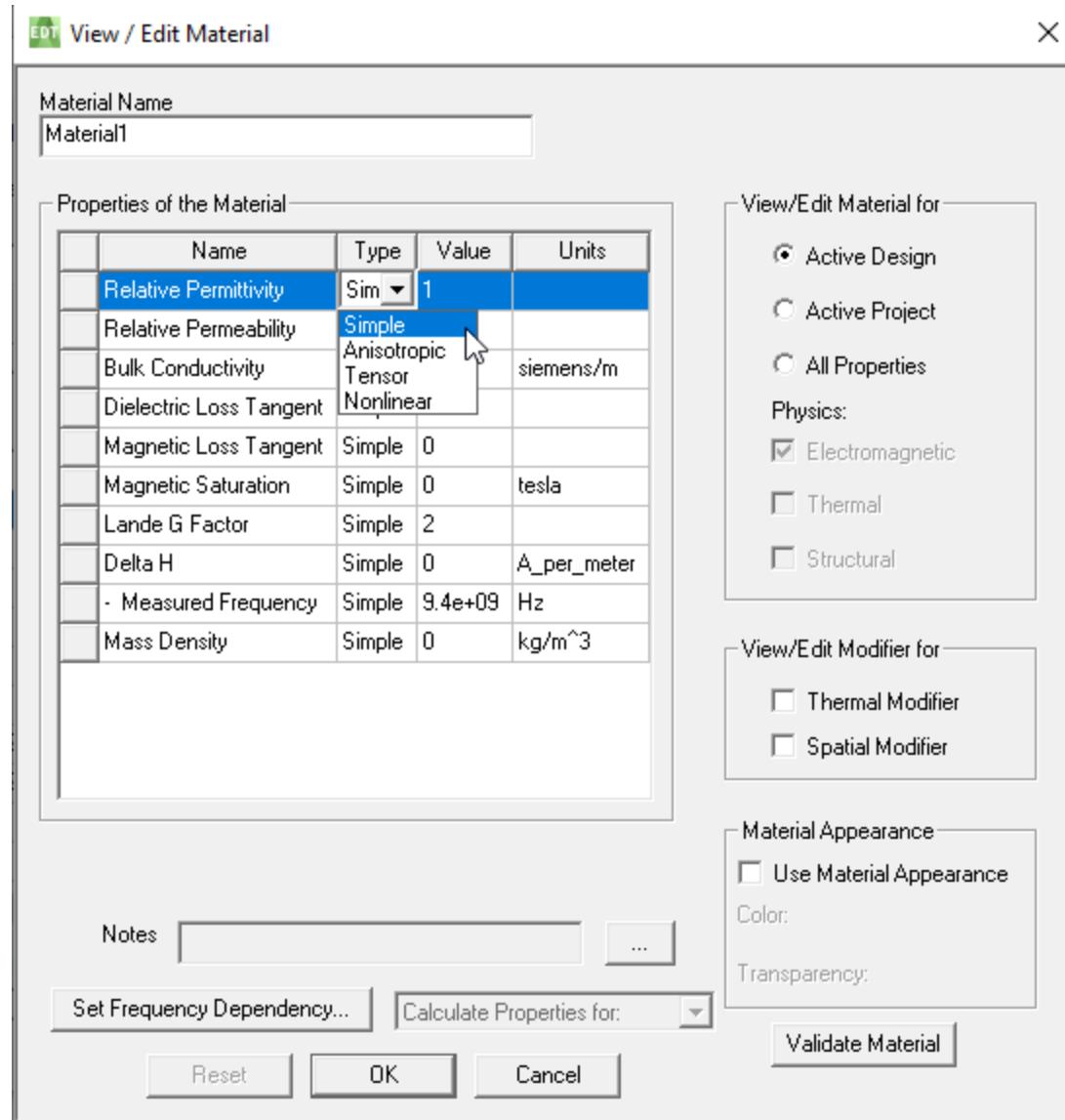
[ExampleGetLayeredImpedanceBoundaryPropertyNamesandValues.htm](#)

## Materials Scripting Support

Supported material properties are shown in a Property window for the material item.



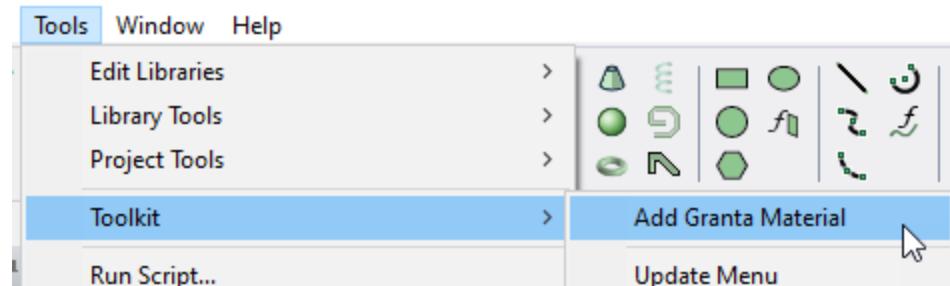
Some Material Properties like Relatively Permittivity may have values assigned as BH Curves or Tensors, as discussed in the Assigning Materials chapter of the online help.

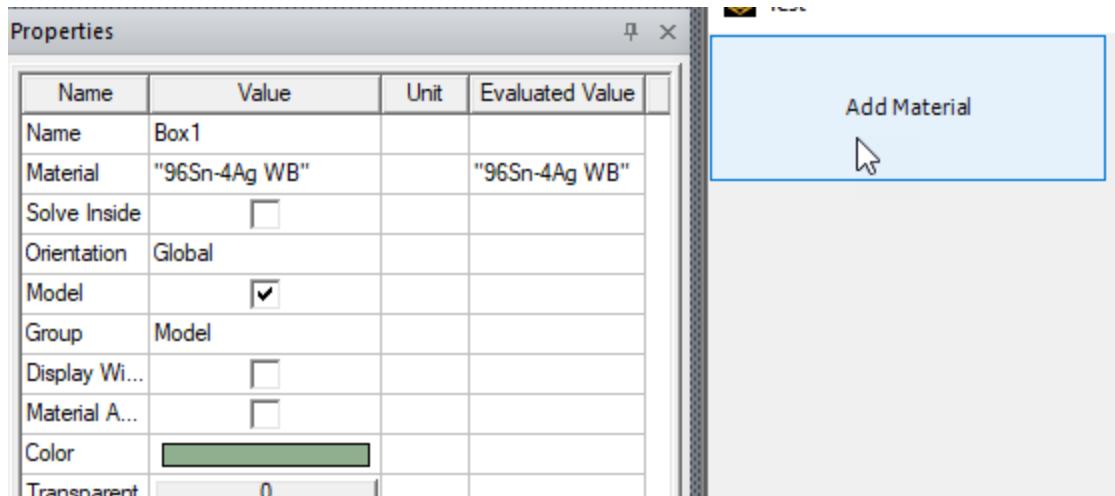


With this feature enabled you can then:

- Get/Set Simple material property
- Get/Set Anisotropic material property
- Get/Set Nonlinear material property
- Get/Set Vector material property
  - Components hide/shown as needed
- Get/Set Tensor material property
- Get/Set Choice material property
- [AddDefinitionFromBlock](#)
- [AddDefinitionFromLibFile](#)
- [GetExtendedDefinitionObject](#)

A new Toolkit allows you to select materials from the Granta materials gateway, such that project materials will automatically be added when you select a material from the gateway, and that the gateway itself is easily accessed from the materials.





What is not supported:

- Change of property type
- Custom material property, due to its complexity

## Object Oriented Scripting for Materials

Materials are Child objects of the Active Project. In the IronPython command window, you can execute GetPropNames() for a specified material as follows:

```
>>> omats = oDesktop.GetActiveProject().GetChildObject("Materials")
>>> omat = omats.GetChildObject("vacuum")
>>> omat.GetPropNames()
['Coordinate System Type', 'Coordinate System Type/Choices', 'Relative Permeability Type',
'Relative Permeability Type/Choices', 'Relative Permeability', 'Relative Permeability/SIValue',
```

```
'Relative Permeability/EvaluatedValue', 'Bulk Conductivity Type', 'Bulk Conductivity Type/Choices', 'Bulk Conductivity', 'Bulk Conductivity/SIValue', 'Bulk Conductivity/EvaluatedValue', 'Magnetic Coercivity Type', 'Magnetic Coercivity Magnitude', 'Magnetic Coercivity Magnitude/SIValue', 'Magnetic Coercivity Magnitude/EvaluatedValue', 'Composition', 'Composition/Choices', "Young's Modulus Type", "Young's Modulus Type/Choices", "Young's Modulus", "Young's Modulus/SIValue", "Young's Modulus/EvaluatedValue", "Poisson's Ratio Type", "Poisson's Ratio", "Poisson's Ratio Type/Choices", "Poisson's Ratio", "Poisson's Ratio/SIValue", "Poisson's Ratio/EvaluatedValue"]
```

### Examples showing change to material property type:

```
>>> omat.GetPropValue("Relative Permeability Type/Choices")
['Simple', 'Anisotropic', 'Tensor', 'Nonlinear']
>>> omat.GetPropValue("Relative Permeability Type")
'Nonlinear'
>>> omat.SetPropValue("Relative Permeability Type", "Simple")
True
>>> omat.GetPropValue("Relative Permeability Type")
'Simple'
>>> omat.SetPropValue("Relative Permeability", 10)
True
```

### Examples showing change to a vector component value

```
>>> omat.GetPropValue("Magnetic Coercivity Magnitude")
'0A_per_meter'
>>> omat.SetPropValue("Magnetic Coercivity Magnitude", "-1A_per_meter")
```

```
True
```

```
>>> omat.GetPropNames()
```

```
['Coordinate System Type', 'Coordinate System Type/Choices', 'Relative Permeability Type',  
'Relative Permeability Type/Choices', 'Relative Permeability', 'Relative Permeability/SIValue',  
'Relative Permeability/EvaluatedValue', 'Bulk Conductivity Type', 'Bulk Conductivity  
Type/Choices', 'Bulk Conductivity', 'Bulk Conductivity/SIValue', 'Bulk Con-  
ductivity/EvaluatedValue', 'Magnetic Coercivity Type', 'Magnetic Coercivity Magnitude', 'Mag-  
netic Coercivity Magnitude/SIValue', 'Magnetic Coercivity Magnitude/EvaluatedValue', 'Magnetic  
Coercivity Components', 'Magnetic Coercivity Components/Component1', 'Magnetic Coercivity Com-  
ponents/Component2', 'Magnetic Coercivity Components/Component3', 'Composition', 'Com-  
position/Choices', '- Stacking Factor Type', '- Stacking Factor', '- Stacking Factor/SIValue',  
'- Stacking Factor/EvaluatedValue', '- Stacking Direction', '- Stacking Direction/Choices',  
"Young's Modulus Type", "Young's Modulus Type/Choices", "Young's Modulus", "Young's Mod-  
ulus/SIValue", "Young's Modulus/EvaluatedValue", "Poisson's Ratio Type", "Poisson's Ratio  
Type/Choices", "Poisson's Ratio", "Poisson's Ratio/SIValue", "Poisson's Ratio/EvaluatedValue"]
```

```
>>> omat.SetPropValue("Magnetic Coercivity Components/Component2", 2)
```

```
True
```

```
>>> omat.GetPropValue("Magnetic Coercivity Components")
```

```
['Component1:=', '2', 'Component2:=', '2', 'Component3:=', '0']
```

## Change choice property value

```
>>> omat.GetPropValue("Composition/Choices")
```

```
['Solid', 'Lamination', 'Litz Wire']
```

```
>>> omat.SetPropValue("Composition", "Lamination")
```

```
True
```

```
>>> omat.GetPropNames()
```

```
['Coordinate System Type', 'Coordinate System Type/Choices', 'Relative Permeability Type',
'Relative Permeability Type/Choices', 'Relative Permeability', 'Relative Permeability/SIValue',
'Relative Permeability/EvaluatedValue', 'Bulk Conductivity Type', 'Bulk Conductivity
Type/Choices', 'Bulk Conductivity', 'Bulk Conductivity/SIValue', 'Bulk Con-
ductivity/EvaluatedValue', 'Magnetic Coercivity Type', 'Magnetic Coercivity Magnitude', 'Magnetic
Coercivity Magnitude/SIValue', 'Magnetic Coercivity Magnitude/EvaluatedValue', 'Magnetic Coer-
civity Components', 'Magnetic Coercivity Components/Component1', 'Magnetic Coercivity Com-
ponents/Component2', 'Magnetic Coercivity Components/Component3', 'Composition',
'Composition/Choices', '- Stacking Factor Type', '- Stacking Factor', '- Stacking Fact-
or/SIValue', '- Stacking Factor/EvaluatedValue', '- Stacking Direction', '- Stacking Dir-
ection/Choices', "Young's Modulus Type", "Young's Modulus Type/Choices", "Young's Modulus",
"Young's Modulus/SIValue", "Young's Modulus/EvaluatedValue", "Poisson's Ratio Type", "Poisson's
Ratio Type/Choices", "Poisson's Ratio", "Poisson's Ratio/SIValue", "Poisson's Ratio/E-
valuatedValue"]
```

```
>>> omat.SetPropValue("Magnetic Coercivity Components/Component2", 2)
```

```
True
```

```
>>> omat.GetPropValue("Magnetic Coercivity Components")
```

```
['Component1:=' , '2', 'Component2:=' , '2', 'Component3:=' , '0']
```

## Change choice property value

```
>>> omat.GetPropValue("Composition/Choices")
['Solid', 'Lamination', 'Litz Wire']
>>> omat.SetPropValue("Composition", "Lamination")
True
>>> omat.GetPropNames()
```

```
['Coordinate System Type', 'Coordinate System Type/Choices', 'Relative Permeability Type',
'Relative Permeability Type/Choices', 'Relative Permeability', 'Relative Permeability/SIValue',
'Relative Permeability/EvaluatedValue', 'Bulk Conductivity Type', 'Bulk Conductivity
Type/Choices', 'Bulk Conductivity', 'Bulk Conductivity/SIValue', 'Bulk Con-
ductivity/EvaluatedValue', 'Magnetic Coercivity Type', 'Magnetic Coercivity Magnitude', 'Mag-
netic Coercivity Magnitude/SIValue', 'Magnetic Coercivity Magnitude/EvaluatedValue', 'Magnetic Coercivity Components', 'Magnetic Coercivity Components/Component1', 'Magnetic Coercivity Com-
ponents/Component2', 'Magnetic Coercivity Components/Component3', 'Composition', 'Com-
position/Choices', '- Stacking Factor Type', '- Stacking Factor', '- Stacking Factor/SIValue',
'- Stacking Factor/EvaluatedValue', '- Stacking Direction', '- Stacking Direction/Choices',
"Young's Modulus Type", "Young's Modulus Type/Choices", "Young's Modulus", "Young's Mod-
ulus/SIValue", "Young's Modulus/EvaluatedValue", "Poisson's Ratio Type", "Poisson's Ratio
Type/Choices", "Poisson's Ratio", "Poisson's Ratio/SIValue", "Poisson's Ratio/EvaluatedValue"]
```

>>>

This page intentionally  
left blank.

## 7 - Property Script Commands

Property script commands allow you to navigate through all objects and properties in a project. You can get and set all properties for all objects in the Project tree with simple data types.

Property Object is the base class defined for all script objects that support the properties Get and Set.

GetName ()

- Returns the name of the object.

GetChildTypes ()

- An object may have different types of children. For example, a design may have variables, modules, and editors.
- Returns an array of text strings; may be empty if the children are not divided into different types.

GetChildNames (<type>)

- <type> – Child type name. By default, returns all children names for all types.
- Returns an array of immediate children names, belonging to a type if specified.

GetChildObject (<objPath>)

- <objPath> – A child object path; can contain multiple generations (for example, designObject/moduleObject/SetupObject).
- Returns a child property object if the object is found.

GetPropNames (<bIncludeReadOnly>)

- <bIncludeReadOnly> – Optional; defaults to true. True includes read-only properties; False excludes read-only properties.
- Returns an array of the object's property names.

GetPropValue (<propertyPath>)

- <propertyPath> – The property's path; may be a child object's path appended with a property name (for example, TeeModel/Offset/SIValue).
- Returns the property value if found. Otherwise causes script error.

```
SetPropValue(<propertyPath>, <data>)
```

- <propertyPath> – The property's path; may be a child object's path appended with a property name (for example, TeeModel/Offset/SIValue).
- <data> – New data; type depends on property type.
- Returns True if updated successfully; False if new data is invalid.

For a detailed summary of how Property script commands are used in a range of contexts, including Variable objects, see: [Object Script Property Function Summary](#). Additional examples for these commands are listed under [Project Objects](#), [Design Objects](#), [3D Modeler](#), [Optimetrics](#), Radiation Module and [Reporter](#).

**Note:**

Older property commands should be executed by the oProject object.

```
Set oProject = oDesktop.SetActiveProject ("Project1")  
oProject.CommandName <args>
```

**Some of the topics covered in this chapter are as follows:**

[Conventions Used in this Chapter](#)

[ChangeProperty](#)

[GetArrayVariables](#)

[GetProperties](#)

[GetProperty](#)

[GetVariables](#)

[GetVariable](#)

[SetProperty](#)

[SetVariable](#)

[Additional Property Scripting Example](#)

[Example Use of Record Script and Edit Properties](#)

## Object Script Property Function Summary

### Object Path

The Object path can be used to navigate through objects and properties in an Ansys EM project.

- An Object path consisted of one or multiple Object-ID-Nodes separated by "/" .
- Object-ID-Node; may exist in the following forms:
  - A simple object name or property name.
  - Type[Name] for object; Tab[name] for property.
  - Name[attr1="v1", attr2 = "v2", ...]. When more than one child object have the same name, use attributes to specify the difference.
  - ArrayName[index]. For example, in an Optimetric setup with multiple calculations, "Calculation[0]" could be used to identify the first calculation.
  - Name beginning with '@' character denoted as a property name, when an object has a child and property with the same name.

### Property Object

The Property Object is the base class defined for all script object that support property Get & Set.

- `GetName()`
  - Returns the name of the object.
- `GetChildTypes()`
  - An object may have different type of children. For example, a design may have variables, modules, and editors.
  - Returns array of text strings; may be empty if the children are *not* divided to different types.
- `GetChildNames(<type>)`
  - *<type>* – children type name; default returns all children names for all types.
  - Returns an array of immediate children names, belonging to the type if specified.
- `GetChildObject(<objPath>)`
  - *<objPath>* – A child object path. The path may include multiple generations, such as (designObject/moduleObj/SetupObject).
  - Returns a child property object if the object found.
- `GetPropEvaluatedValue(<propName>)`
  - Return the Evaluated-Value for Value-Property and Variable.
  - Return the Property-value as text string for other property types.
- `GetPropSIValue(<propName>)`
  - Return the SI-Value for Value-Property and Variable.
  - Return NAN for other property type if its value is cannot be converted to a double-floating point value.
- `GetPropNames(<bIncludeReadOnly>);`
  - *<bIncludeReadOnly>* – optional, default to true; True will include read-only properties, False will exclude read-only properties.
  - Returns an array of the object's property names.

- GetPropValue(<propertyPath>)
  - <propertyPath> – the property's full path. A property name or child object's path appended with a property name, like "TeeModel/Offset/SIValue"
  - Returns the property value if the property is found; otherwise causes script error.
- SetPropValue(<propertyPath>, <data>)
  - <propertyPath> – the property's full path. A property name or child object's path appended with a property name, like "TeeModel/Offset/Value"
  - <data> – new data, type is dependent on property type.
  - Returns True if property data is updated successfully; False if the new data is invalid.

## Project Object

Project Object inherited all functions defined in the Property Object. But it doesn't have property, GetPropValue() & SetPropValue() function can be used to set its child object's property.

- GetChildTypes() always return ["Design", "Variable"].
- GetChildNames(type)  
GetChildNames() & GetChildName("Design") will return all Design names of the project.  
GetChildNames("Variable") return all project variable names.
- GetChildObject(objPath)  

```
oDesign = oProject("TeeModel")  
  
oVariable = oProject.GetChildObject("VariableName")  
  
oReport = oProject.GetChildObject("TeeModel/Results/S Parameter Plot 1")
```
- GetPropNames(bnIncludeReadOnly) always return empty array since the project has no property.

- GetPropValue(propertyPath)  

```
oProject.GetPropValue("TeeModel/offset") //get the offset variable value in the TeeModel Design  
oProject.GetPropValue("TeeModel/Results/S Parameter Plot 1/Display Type") // Get the report display type.
```
- SetPropValue(propertyPath, newValue)  

```
oProject.SetPropValue("TeeModel/offset", "2mm") //Set the offset variable value to "2mm" in the TeeModel Design  
oProject.SetPropValue("TeeModel/Results/S Parameter Plot 1/Display Type", "Data Table") // Set the report display type to data table.
```

## Design Object

Design Object inherited all functions defined in the Property Object. But it doesn't have property, GetPropValue() & SetPropValue() function can be used to set its child object's property..

- GetChildTypes() always return ['Module', 'Editor', 'Variable'].  
GetChildNames(type) will return modules & editor child names.  
GetChildNames("Variable") will return all variable names  
GetChildNames("Module") will return all module names that support property-object-script like ['Optimetrics', 'RadField', 'Results']  
GetChildNames("Editor") will return a 3D editor name for all 3D Designs
- GetChildObject()  

```
oVariable = oDesign.GetChildObject("VariableName")  
oReport = oDesign.GetChildObject("Results/S Parameter Plot 1")  
oRptModule = oDesign.GetChildObject("ReportSetup")
```
- GetPropNames(bIncludeReadOnly) always return empty array since the design has no property.

- GetPropValue()  
oDesign.GetPropValue("offset/SIValue") //get the offset variable SI value in the Design  
oDesign.GetPropValue("Results/S Parameter Plot 1/Display Type") // Get the report display type
- SetPropValue()  
oDesign.SetPropValue("offset", "2mm") //Set the offset variable value to "2mm" in the Design  
oDesign.SetPropValue("Results/S Parameter Plot 1/Display Type", "Data Table") // Set the report display type to data table.

## 3D Modeler Object

GetChild commands returns the appropriate properties for modeler objects. For 3D Components and UDMs, these commands do not return parts, coordinate systems, plans, as top-level modeler children.

```
oModeler = oDesktop.GetActiveProject().GetActiveDesign().GetChildObject("3D Modeler")
oModeler.GetChildNames()
oModeler.GetChildNames("ModelParts")
oModeler.GetChildNames("AllParts")
oModeler.GetChildNames("NonModelParts")
oModeler.GetChildNames("Planes")
oModeler.GetChildNames("CoordinateSystems")
```

## Variable Object

Is a Property Object that has no child. It also provides quick function call to get/set it properties by adding functions with property name appended to Get\_ & Set\_ prefix. To find what functions it provided enter dir(oVar) the command window. It can accessed by the project or design object's GetChildObject(VariableName) function.

```
oProjVar = oProject.GetChildObject("$VarName")
oVar = oProject.GetChildObject("DesignName/VarName")
oVar = oDesign.GetChildObject("variableName")
oProject.GetChildNames("Variable") will return all project variable names.
oDesign.GetChildNames(Variable") will return all Design Variable names.
```

- GetChildTypes() always return empty array.
- GetChildNames() always return empty array , since variable has no child.
- GetChildObject(objPath) it has no child.
- GetPropNames(bIncludeReadOnly) ['EvaluatedValue', 'SIValue'] are read-only properties
  - Independent variable :["Value", 'EvaluatedValue', 'SIValue', 'Description', 'ReadOnly', 'Hidden', 'Sweep', 'Optimization/Included', 'Optimization/Min', 'Optimization/Max', 'Sensitivity/Included', 'Sensitivity/Min', 'Sensitivity/Max', 'Sensitivity/IDisp', 'Statistical', 'Statistical/Included', 'Tuning/Included', 'Tuning/Step', 'Tuning/Min', 'Tuning/Max'].
  - Dependent variable ["Value", 'EvaluatedValue', 'SIValue', 'Description', 'ReadOnly', 'Hidden', 'Sweep']
- GetPropValue(propName)  
oVar.GetPropValue() return the variable value as text string.  
oVar.GetPropValue("Value") return the variable value as text string.  
oVar.GetPropValue("SIValue") return the SI-value of variable as number.  
oVar.Get\_SIValue() also return the SI value.
- SetPropValue(propName, newValue)  
oVar.SetPropValue("Value", 888)  
oVar.SetPropValue("Sensitivity/Included", True)  
oVar.SetPropValue("Sensitivity/Max", '1.8pF'])  
oVar.Set\_Sensitivity\_Max( '1.8pF') also works as last call.  
oVar.SetPropValue("Sensitivity", ['Min:=' , '0.8pF', 'Max:=' , '1.8pF'])  
//set multiple attributes at one call:  
oVar.SetPropValue("@", ["Value:='288, 'Sensitivity', ['Included', True,'Min', '0.0']]])  
oVar.SetPropValue("", ["Value:='288, 'Sensitivity', ['Included', True,'Min', '0.0']]])

## Optimetrics Module Object:

Optimetrics Module Object inherited all functions defined in the Property Object. But it doesn't have property, GetPropValue() & SetPropValue() function can be used to set its child object's property..

- GetChildTypes() there are six type of children, they are ['OptiParametric', 'OptiOptimization', 'OptiSensitivity', 'OptiStatistical', 'OptiDesignExplorer', 'OptiDXDOE']. But the return array only included those that have setup defined, so it may be an empty array if no optimetrics setup is defined. The GetChildNames(type) function also recognized the type name without the prefix "Opti".
- GetChildNames(type)  
GetChildNames() will return all setup for all types.  
GetChildNames("OptiOptimization") & GetChildNames("Optimization") will return all Optimization setup.
- GetChildObject()  
oParamSetup = oOptModule.GetChildObject('ParametricSetup1') get the  
oOptSetup = oOptModule.GetChildObject('OptimizationSetup1')
- GetPropNames(bIncludeReadOnly) always return empty array since the it has no property.
- GetPropValue(propPath) may be used to get its child's property value  
oOptModule.GetPropValue("OptimizationSetup1\Optimizer") get the optimizer name for OptimizationSetup1
- SetPropValue(propPath, newValue) may be used to set its child's property value  
oOptModule.SetPropValue(ParametricSetup1\Enabled", False) //disable ParametricSetup1

## Optimetrics Setup Object

This is a new Object inherited all functions defined in the Property Object. But it doesn't have child. It is accessible through its parents.

```
oOptSetup = oOptModule.GetChildObject('OptimizationSetup1')
oOptSetup = oDesign.GetChildObject('Optimetrics\OptimizationSetup1')
oOptSetup = oProject.GetChildObject('TeeModel\Optimetrics\OptimizationSetup1')
```

- GetChildTypes() always return empty array.
- GetChildNames(type) always return empty array
- GetChildObject()
- GetPropNames(bIncludeReadOnly) will return the property names listed in the property window when the setup is selected.
- GetPropValue(propName)
  - oOptSetup.GetPropValue("Optimizer") return the selected optimizer name.
  - oOptSetup.GetPropValue("Optimizer/Choices") return all optimizer names.
- SetPropValue(propName, newValue)
  - oOptSetup.SetPropValue("Optimizer", "NotAnOptimzerName"); will return false.
  - oOptSetup.SetPropValue("Optimizer", "Quasi Newton"); return true, since "Quasi Newton" is one of the optimizer name returned as the Optimizer Choices.
- HasResult() return true if the setup is solved. Otherwise return false.
- Validate() return true if the setup is valid for analyze. Otherwise return false. Calling the SetPropValue() function to change the property may invalid the setup.

### **ReportSetup(Results) Module Object:**

ReportSetup module Object inherited all functions defined in the Property Object. But it doesn't have property, GetPropValue() & SetPropValue() function can be used to get/set its child object's property..

- GetChildTypes() always empty array.
- GetChildNames(type)
  - GetChildNames() return all report names
- •GetChildObject(objPath)
  - oRpt = oRptModule.GetChildObject("S Parameter Plot 1") return the report property object

```
oTrace = oRptModule.GetChildObject("S Parameter Plot 1/dB(S(Port1,Port1))") return the trace property object  
oAxisX = oRptModule.GetChildObject("S Parameter Plot 1/AxisX") return the axis X property object
```

- GetPropNames(bIncludeReadOnly) always return empty array since the it \has no property.
- GetPropValue()  
oRptModule.GetPropValue("S Parameter Plot 1/Display Type")
- SetPropValue()  
oRptModule.SetPropValue("S Parameter Plot 1/Display Type", "DataTable")

## ReportSetup(Results) Module Child Objects:

These are Property Objects. Its first level of child object is report. Report has trace, axis, header, Legend, and more children. Trace has curve as child etc.

Those child objects can be accessed by calling all levels of parent object's GetChildObject(path) function.

```
oRpt = oRptModule.GetChildObject(reportName)
```

```
oRpt = oDesign.GetChildObject("Results/reportName")
```

```
oTrace = oRpt.GetChildObject(traceName)
```

```
oTrace = oRptModule.GetChildObject(ReportName/TraceName)
```

- GetChildTypes() always return empty array.
- GetChildNames() get the object's child names. What will be returned will depended on the object instance.
- GetChildObject(objPath)
- GetPropNames(bIncludeReadOnly) will return the property names listed in the property window when the object is selected.
- GetPropValue(propName)  
oRpt.GetPropValue("Display Type") return the report's display type.  
oOptSetup.GetPropValue("Display Type/Choices") return all optimizer names.  
oTrace.GetPropValue("X Component")

- SetPropValue(propName, newValue)  
oTrace.SetPropValue("Primary sweep", "Freq")

### Radiation Module Object:

This inherited all functions defined in the Property Object. But it doesn't have property, GetPropValue() & SetPropValue() function can be used to set its child object's property.

- GetChildTypes() always return empty array, now its children
- GetChildNames(type)  
GetChildNames() return all setup names.
- GetChildObject(setupName) return the setup object as Property object.  
oOverlay= oRadModule.GetChildObject('Antenna Parameter Overlay1')  
oSphere = oRadModule.GetChildObject('Infinite Sphere1')
- GetPropNames() return empty array; it has no property.
- GetPropValue()  
oRadModule.GetPropValue('Line1/Num Points') //Get the Line1 setups' "Num Points" property value.
- SetPropValue()  
oRadModule.SetPropValue('Line1/Num Points', 100); Set the Line1 setups' "Num Points" property to 100.

### Radiation Module Child Objects:

These are Property Objects. It also provides quick function call to get/set its properties by adding functions with property name appended to Get\_ & Set\_ prefix. To find what functions it provides enter dir(oVar) the command window.

Those child objects can be access by call all levels of parent object's GetChildObject(path) function.

```
oRadSetup = oRadModule.GetChildObject(setupName)  
oRadSetup = oDesign.GetChildObject(RadField/setupName)
```

- GetChildTypes() always return empty array.
- GetChildNames() always return empty array , since Radiation setup has no child.
- GetChildObject(objPath) it has no child.
- GetPropNames(bIncludeReadOnly) will return the property names listed in the property window when the setup is selected.
- GetPropValue(propName)
  - oRadSetup.GetPropValue("Num Points") return the line setup's "Num Points" property value.
  - oRadSetup.Get\_NumPoints() will also get the same value.
- SetPropValue(propName, newValue)
  - oRadSetup.SetPropValue('Num Points', 888)
  - oRadSetup.Set\_NumPoints(888)

## Conventions Used in this Chapter

General Definitions:

<b>Property</b>	A single item that can be modified in the <b>Properties</b> window or in the modal <b>Properties</b> pop-up window.
<b>&lt;PropServer&gt;</b>	The item whose properties are being modified. This is usually a compound name, giving all information needed by the editor, design, or project in order to locate the item.
<b>&lt;PropTab&gt;</b>	Corresponds to one tab in the <b>Properties</b> window, the one under which properties are being edited.
<b>&lt;PropName&gt;</b>	The name of a single property.

The following tables list specific <PropServer> and <PropTab> values for different property types.

For Project Variables:

<b>&lt;PropServer&gt;</b>	"ProjectVariables"
<b>&lt;PropTab&gt;</b>	"ProjectVariableTab"

For Local Variables:

<b>&lt;PropServer&gt;</b>	"LocalVariables"
---------------------------	------------------

<b>&lt;PropTab&gt;</b>	"LocalVariableTab"
------------------------	--------------------

For Passed Parameters:

<b>&lt;PropServer&gt;</b>	"Instance:<Name of Circuit Instance>"
<b>&lt;PropTab&gt;</b>	"PassedParameter Tab"

For Definition Parameters:

<b>&lt;PropServer&gt;</b>	"DefinitionParameters"
<b>&lt;PropTab&gt;</b>	"DefinitionParameters"

For Modules and Editors:

<b>&lt;PropServer&gt;</b>	<ModuleName>:<ItemName> where <ItemName> is the boundary name, solution setup name, etc. For example, "BoundarySetup:PerfE1"
<b>&lt;PropTab&gt;</b>	Boundary Module: "HfssTab"  Mesh Operations Module: "MeshSetupTab"  Analysis Module: "HfssTab"  Optimetrics Module: "OptimetricsTab"  Solutions Module: <i>Does not support properties.</i>  Field Overlays Module: "FieldsPostProcessorTab"  Radiation Module: "RadFieldSetupTab"  Circuit Module: "CCircuitTab"  System Module: "SystemTab"  HFSS 3D Layout Module: "HFSS 3D LayoutTab"

	Nexxim Module: "NexximTab" Layout elements: "BaseElementTab" Schematic elements: "ComponentTab" Optimetrics Module: "OptimetricsTab"
--	---

For 3D Model Editor objects:

<b>&lt;PropServer&gt;</b>	Name of the object. For example, "Box1".
<b>&lt;PropTab&gt;</b>	"Geometry3DAttributeTab"

For 3D Model Editor operations:

<b>&lt;PropServer&gt;</b>	<ObjName>:<OperationName>:<int> where <int> is the operation's history index. For example, "Box2:CreateBox:2" refers to the second "CreateBox" operation in Box2's history.
<b>&lt;PropTab&gt;</b>	"Geometry3DCmdTab"

For Reporter operations on Report properties:

<b>&lt;PropServer&gt;</b>	<b>&lt;ReportSetup&gt;</b>
<b>&lt;ChangeProperty&gt;</b>	Array. For example, to set the company name in a plot header to "My Company":  Set oModule = oDesign.GetModule("ReportSetup")  oModule.ChangeProperty Array("NAME:AllTabs", _ Array("NAME:Header", _ Array("NAME:PropServers", _  "XY Plot1:Header"), Array("NAME:ChangedProps", _ Array("NAME:Company Name", "Value:=", "My Company"))))

For Maxwell Modules:

<b>&lt;PropServer&gt;</b>	<ModuleName>:<ItemName> where <ItemName> is the solution setup name.
<b>&lt;PropTab&gt;</b>	Boundary module: "MaxwellTab"

	Mesh Operations module: "MeshSetupTab" Analysis module: "MaxwellTab" Optimetrics module: "OptimetricsTab" Solutions module: <i>Does not support properties.</i> Field Overlays module: "FieldsPostProcessorTab" Radiation module: "RadFieldSetupTab"
--	---

**Note:**

For scripted property changes in the various modules and editors, refer to the chapters on the System, HFSS 3D Layout, and Nexxim tools, as well as the Layout and Schematic editors.

### ChangeProperty (HFSS and Maxwell)

**Use:** Changes to properties are scripted using the ChangeProperty command. This command can be executed by the oEditor to change editor properties, by the oDesign to change design level properties, and by the oProject to change project level properties. The command can be used to create, edit, and/or remove properties. In HFSS and Maxwell, only Variable and Separator properties can be deleted.

Use the script recording feature and edit a property, and then view the resulting script entry or use GetPropertyValue for the desired property to see the expected format.

**Command:** None

**Syntax:** ChangeProperty Array("Name:AllTabs", <PropTabArray>, <PropTabArray>, ...)

ChangeProperty(<modulename>:<setup name>:<sweep name>)

**Return Value:** None

**Parameters:** <PropTabArray>

```
Array("Name:<PropTab>",
    <PropServersArray>,
    <NewPropsArray>,
    <ChangedPropsArray>,
    <DeletedPropsArray>

<PropServersArray>
    Array("Name:PropServers", <PropServer>,
        <PropServer>, ...)

<NewPropsArray>
    Array("Name:NewProps", <PropdataArray>,
        <PropdataArray>, ...)

<ChangedPropsArray>
    Array("Name:ChangedProps", <PropdataArray>,
        <PropdataArray>, ...)

<DeletedPropsArray>
    Array("Name:DeletedProps", <PropName>,
        <PropName>, ...)
```

```
<PropdataArray>
    Array ("NAME:<PropName>",
        "PropType:=", <PropType>,
        "NewName:=", <string>,
        "Description:=", <string>,
        "NewRowPosition:=", <int>,
        "ReadOnly:=", <bool>,
        "Hidden:=", <bool>,
        <PropTypeSpecificArgs>)
```

```
<PropType>
```

Type: string

Identifies the type of property when a new property is added. In HFSS and Maxwell, only separator properties and variable properties can be added.

```
"SeparatorProp"
"VariableProp"
"TextProp"
"NumberProp"
"ValueProp"
"CheckboxProp"
```

```
"MenuProp"  
"PointProp"  
"VPointProp"  
"V3DPointProp"  
"ButtonProp"
```

#### NewName

Specify the new name of a property if the property's name is being edited. In HFSS and Maxwell, the name can only be changed for separators and variables.

#### Description

Specify a description of the property. In HFSS and Maxwell, the description can only be changed for separators and variables.

#### NewRowPosition

Used to reorder rows in the **Property** dialog box.

In HFSS, this only applies to the **Project>Project Variables** panel and the **Hfss>DesignProperties** panel.

In Maxwell, this only applies to the **Project>Project Variables** panel and the **Maxwell3D>DesignProperties**, **Maxwell2D>Design Properties**, or **RMxprt>Design Properties** panels.

Specify the new zero-based row index of the variable or separator.

#### ReadOnly

Used to mark a property as "read only" so it can not be modified. In HFSS and Maxwell, this flag can only be set for variables and separators.

#### **Hidden**

Used to hide a property so it can not be viewed outside of the **Property** dialog box. In HFSS and Maxwell, this flag can only be set for variables and separators.

```
<PropTypeSpecificArgs>

    SeparatorProp: no arguments

    TextProp: "Value:=", <string>

    NumberProp: "Value:=", <double>

    ValueProp: "Value:=", <value>

    CheckboxProp: "Value:=", <bool>

    MenuProp: "Value:=", <string>

    PointProp "X:=", <double>, "Y:=", <double>

    VPointProp: "X:=", <value>, "Y:=", <value>

    V3DPointProp: "X:=",<value>, "Y:=",<value>, "Z:=",<value>

    Material Button: "Material:=", <string>

    Color Button: "R:="",<int>, "G:="",<int>, "B:="",<int>

    Transparency Button: "Value:=", <double>
```

```
<PropTypeSpecificArgs> for VariableProps
```

Syntax:

```
"Value:=", <value>, <OptimizationFlagsArray>,  
<TuningFlagsArray>, <SensitivityFlagsArray>,  
<StatisticsFlagsArray>
```

Parameters:

```
<OptimizationFlagsArray>  
Array("NAME:Optimization",  
"Included:=", <bool>,  
"Min:=", <value>,  
"Max:=", <value>)
```

```
<Tuning flagsArray>  
Array("NAME:Tuning",  
"Included:=", <bool>,  
"Step:=", <value>,  
"Min:=", <value>,  
"Max:=", <value>)
```

```
<SensitivityFlagsArray>
```

```
Array("NAME:Sensitivity",
"Included:=", <bool>,
"Min:=", <value>,
"Max:=", <value>,
"IDisp:=", <value> )
```

```
<StatisticsFlagsArray>
Array("NAME:Statistical",
"Included:=", <bool>,
"Dist:=", <Distribution>,
"StdD:=", <value>,
"Min:=", <value>,
"Max:=", <value>,
"Tol:=", <string>)
```

```
<Distribution>
  Type: string
    Value should be "Gaussian" or "Uniform"
```

StdD

Standard deviation.

Min

Low cut-off for the distribution.

Max

High cut-off for the distribution.

Tol

Tolerance for uniform distributions. Format is "<int>%".

Example: "20%".

*VB Example:* Adding a new project level variable "\$width":

```
oProject.ChangeProperty Array("NAME:AllTabs",_
    Array("NAME:ProjectVariableTab",_
        Array("NAME:PropServers", "ProjectVariables"),_
        Array("NAME:NewProps",_
            Array("NAME:$width",_
                "PropType:=", "VariableProp",_
                "Value:=", "3mm",_
```

```
"Description:=", "my new variable")))
```

*VB Example:* Deleting the design level variable "height":

```
oDesign.ChangeProperty Array("NAME:AllTabs", _  
    Array("NAME:LocalVariableTab", _  
        Array("NAME:PropServers", "DefinitionParameters"), _  
        Array("NAME:DeletedProps", "height")))
```

*VB Example:* Changing a property's value. If the following command were executed, then the value of the property "XSize" of the PropServer "Box1:CreateBox:1" on the "Geometry3DCmdTab" tab would be changed. (oEditor is the Geometry3D editor in HFSS.)

```
oEditor.ChangeProperty Array("NAME:AllTabs", _  
    Array("NAME:Geometry3DCmdTab", _  
        Array("NAME:PropServers", "Box1:CreateBox:1"), _  
        Array("NAME:ChangedProps", _  
            Array("NAME:XSize", "Value:=", "1.4mil"))))
```

*VB Example:* Changing the Company Name, Design Name, the background color, and the Axis scaling in a Report.

```
Set oProject = oDesktop.SetActiveProject("wgcombiner")
Set oDesign = oProject.SetActiveDesign("HFSSDesign2")
Set oModule = oDesign.GetModule("ReportSetup")
oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Header", _
    Array("NAME:PropServers", "XY Plot1:Header"), _
    Array("NAME:ChangedProps", Array("NAME:Company Name", _
        "Value:=", "My Company"))))

oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Header", _
    Array("NAME:PropServers", "XY Plot1:Header"), _
    Array("NAME:ChangedProps", Array("NAME:Design Name", _
        "Value:=", "WG Combiner"))))

oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:General", _
    Array("NAME:PropServers", "XY Plot1:General"), _
    Array("NAME:ChangedProps", Array("NAME:Back Color", _
        "R:=", 128, "G:=", 255, "B:=", 255)))))

oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Axis", _
    Array("NAME:PropServers", "XY Plot1:AxisX"), _
    Array("NAME:ChangedProps", Array("NAME:Axis Scaling", _
        "Value:=", "Log"))))
```

## GetArrayVariables

Returns a list of array variables. To get a list of indexed project variables, execute with oProject. To get a list of indexed local variables, use oDesign.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array of strings containing names of variables.

<b>Python Syntax</b>	GetArrayVariables()
<b>Python Example</b>	<pre>oProject.GetArrayVariables() oDesign.GetArrayVariables()</pre>

<b>VB Syntax</b>	GetArrayVariables
<b>VB Example</b>	<pre>oProject.GetArrayVariables oDesign.GetArrayVariables</pre>

## GetProperties

Gets a list of all the properties belonging to a specific <PropServer> and <PropTab>. This can be executed by the oProject, oDesign, or oEditor variables.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<PropTab>	String	<p>One of the following, where tab titles are shown in parentheses:</p> <ul style="list-style-type: none"> <li>• PassedParameterTab ("Parameter Values")</li> <li>• DefinitionParameterTab (Parameter Defaults")</li> <li>• LocalVariableTab ("Variables" or "Local Variables")</li> <li>• ProjectVariableTab ("Project variables")</li> <li>• ConstantsTab ("Constants")</li> <li>• BaseElementTab ("Symbol" or "Footprint")</li> <li>• ComponentTab ("General")</li> <li>• Component("Component")</li> <li>• CustomTab ("Intrinsic Variables")</li> <li>• Quantities ("Quantities")</li> <li>• Signals ("Signals")</li> </ul>
<b>Return Value</b>	Array of strings containing the names of the appropriate properties.		

<b>Python Syntax</b>	GetProperties( <PropTab>, <PropServer> )
<b>Python Example</b>	<code>oEditor.GetProperties('PassedParameterTab', 'k')</code>

<b>VB Syntax</b>	GetProperties <PropTab>, <PropServer>
------------------	---------------------------------------

**VB Example**

```
oEditor.GetProperties "PassedParameterTab", "k"
```

## GetPropertyValue

Returns the value of a single property belonging to a specific <PropServer> and <PropTab>. This function is available with the Project, Design or Editor objects, including definition editors.

**Tip:**

Use the script recording feature and edit a property, and then view the resulting script to see the format for that property.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;PropTab&gt;</td> <td>String</td> <td>           One of the following, where tab titles are shown in parentheses:           <ul style="list-style-type: none"> <li>• PassedParameterTab ("Parameter Values")</li> <li>• DefinitionParameterTab (Parameter Defaults")</li> <li>• LocalVariableTab ("Variables" or "Local Variables")</li> <li>• ProjectVariableTab ("Project variables")</li> <li>• ConstantsTab ("Constants")</li> <li>• BaseElementTab ("Symbol" or "Footprint")</li> <li>• ComponentTab ("General")</li> <li>• Component("Component")</li> <li>• CustomTab ("Intrinsic Variables")</li> <li>• Quantities ("Quantities")</li> </ul> </td> </tr> </tbody> </table>	Name	Type	Description	<PropTab>	String	One of the following, where tab titles are shown in parentheses: <ul style="list-style-type: none"> <li>• PassedParameterTab ("Parameter Values")</li> <li>• DefinitionParameterTab (Parameter Defaults")</li> <li>• LocalVariableTab ("Variables" or "Local Variables")</li> <li>• ProjectVariableTab ("Project variables")</li> <li>• ConstantsTab ("Constants")</li> <li>• BaseElementTab ("Symbol" or "Footprint")</li> <li>• ComponentTab ("General")</li> <li>• Component("Component")</li> <li>• CustomTab ("Intrinsic Variables")</li> <li>• Quantities ("Quantities")</li> </ul>		
Name	Type	Description							
<PropTab>	String	One of the following, where tab titles are shown in parentheses: <ul style="list-style-type: none"> <li>• PassedParameterTab ("Parameter Values")</li> <li>• DefinitionParameterTab (Parameter Defaults")</li> <li>• LocalVariableTab ("Variables" or "Local Variables")</li> <li>• ProjectVariableTab ("Project variables")</li> <li>• ConstantsTab ("Constants")</li> <li>• BaseElementTab ("Symbol" or "Footprint")</li> <li>• ComponentTab ("General")</li> <li>• Component("Component")</li> <li>• CustomTab ("Intrinsic Variables")</li> <li>• Quantities ("Quantities")</li> </ul>							

		<ul style="list-style-type: none"> <li>• Signals ("Signals")</li> </ul>
<PropServer>	String	An object identifier, generally returned from another script method, such as CompInst@R;2;3
<PropName>	String	Name of the property.
<b>Return Value</b>	String value of the property.	

<b>Python Syntax</b>	GetPropertyValue (<PropTab>, <PropServer>, <PropName>)
<b>Python Example</b>	<pre>selectionArray = oEditor.GetSelections()  for k in selectionArray:     val = oEditor.GetPropertyValues("PassedParameterTab", k, "R")      ...</pre>

<b>VB Syntax</b>	GetPropertyValue (<PropTab>, <PropServer>, <PropName>)
<b>VB Example</b>	<pre>selectionArray = oEditor.GetSelections  for k in selectionArray:     val = oEditor.GetPropertyValues("PassedParameterTab", k, "R")      ...</pre>

## GetVariables

Returns a list of all defined variables. To get a list of project variables, execute this command using `oProject`. To get a list of local variables, use `oDesign`.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array of strings containing the variables.

<b>Python Syntax</b>	<code>GetVariables ()</code>
<b>Python Example</b>	<code>oProject.GetVariables()</code> <code>oDesign.GetVariables()</code>

<b>VB Syntax</b>	<code>GetVariables</code>
<b>VB Example</b>	<code>oProject.GetVariables</code> <code>oDesign.GetVariables</code>

## GetVariableValue

Gets the value of a single specified variable. To get the value of project variables, execute this command using `oProject`. To get the value of local variables, use `oDesign`.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;VarName&gt;</td> <td>String</td> <td>Name of the variable to access.</td> </tr> </tbody> </table>			Name	Type	Description	<VarName>	String	Name of the variable to access.
Name	Type	Description							
<VarName>	String	Name of the variable to access.							
<b>Return Value</b>	String represents the value of the variable.								

<b>Python Syntax</b>	GetVariableValue( <VarName> )
<b>Python Example</b>	<code>oProject.GetVariableValue("var_name")</code>

<b>VB Syntax</b>	GetVariableValue <VarName>
<b>VB Example</b>	<code>oProject.GetVariableValue "var_name"</code>

## SetPropertyValue

Sets the value of a single property belonging to a specific PropServer and PropTab. This function is available with the Project, Design or Editor objects, including definition editors. This is not supported for properties of the following types: ButtonProp, PointProp, V3DPointProp, and VPointProp. Only the ChangeProperty command can be used to modify these properties.

Use the script recording feature and edit a property, and then view the resulting script entry or use GetPropertyValue for the desired property to see the expected format.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;propTab&gt;</td> <td>String</td> <td>           One of the following, where tab titles are shown in parentheses:  <ul style="list-style-type: none"> <li>• PassedParameterTab ("Parameter Values")</li> </ul> </td> </tr> </tbody> </table>			Name	Type	Description	<propTab>	String	One of the following, where tab titles are shown in parentheses: <ul style="list-style-type: none"> <li>• PassedParameterTab ("Parameter Values")</li> </ul>
Name	Type	Description							
<propTab>	String	One of the following, where tab titles are shown in parentheses: <ul style="list-style-type: none"> <li>• PassedParameterTab ("Parameter Values")</li> </ul>							

			<ul style="list-style-type: none"> <li>• DefinitionParameterTab ("Parameter Defaults")</li> <li>• LocalVariableTab ("Variables" or "Local Variables")</li> <li>• ProjectVariableTab ("Project variables")</li> <li>• ConstantsTab ("Constants")</li> <li>• BaseElementTab ("Symbol" or "Footprint")</li> <li>• ComponentTab ("General")</li> <li>• Component("Component")</li> <li>• CustomTab ("Intrinsic Variables")</li> <li>• Quantities ("Quantities")</li> <li>• Signals ("Signals")</li> </ul>
	<i>&lt;propServer&gt;</i>	String	An object identifier, generally returned from another script method, such as ComplInst@R;2;3
	<i>&lt;propName&gt;</i>	String	Name of the property.
	<i>&lt;propValue&gt;</i>	String	The value for the property
<b>Return Value</b>	None.		

<b>Python Syntax</b>	SetPropertyValue(<propTab>, <propServer>, <propName>, <propValue>)
<b>Python Example</b>	<code>oEditor SetPropertyValue ("PassedParameterTab", "k", "R", "2200")</code>

---

<b>VB Syntax</b>	SetPropertyValue <propTab>, <propServer>, <propName>, <propValue>
<b>VB Example</b>	<code>oEditor SetPropertyValue "PassedParameterTab", "k", "R", "2200"</code>

## SetVariableValue

Sets the value of a variable. To set the value of a project variable, execute this command using `oProject`. To set the value of a local variable, use `oDesign`.

<b>UI Access</b>	N/A									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;VarName&gt;</td> <td>String</td> <td>Variable name.</td> </tr> <tr> <td>&lt;VarValue&gt;</td> <td>Value</td> <td>New value for the variable.</td> </tr> </tbody> </table>	Name	Type	Description	<VarName>	String	Variable name.	<VarValue>	Value	New value for the variable.
Name	Type	Description								
<VarName>	String	Variable name.								
<VarValue>	Value	New value for the variable.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	SetVariableValue (<VarName>, <VarValue>)
<b>Python Example</b>	<code>oProject.SetVariableValue ('\$Var1', '3mm')</code>

<b>VB Syntax</b>	SetVariableValue <VarName>, <VarValue>
<b>VB Example</b>	<code>oProject.SetVariableValue "\$Var1", "3mm"</code>

This page intentionally  
left blank.

# 8 - Dataset Script Commands

Dataset commands should be executed by the oProject object:

```
Set oProject = oDesktop.SetActiveProject("Project1")
oProject.CommandName <args>
```

[AddDataSet](#)

[DeleteDataSet](#)

[EditDataSet](#)

[ExportDataSet](#)

[HasDataSet](#)

[ImportDataSet](#)

## AddDataset

Adds a dataset. This can be executed by the oProject, or oDesign variables.

UI Access	Project > Datasets > Add.		
Parameters	Name	Type	Description
	< DatasetdataArray>	Array	Array("NAME:<DatasetName>", Array("NAME:Coordinates", <CoordinateArray>, <CoordinateArray>, ...))
	<DatasetName>	String	Name of the dataset.
<CoordinateArray> Array			Array("NAME:Coordinate", "X:=", <double>, "Y:=", <double>)
Return Value	None.		

<b>Python Syntax</b>	AddDataset < <i>DatasetdataArray</i> >
<b>Python Example</b>	<pre>oProject.AddDataset(     [         "NAME:\$ds1",         [             "NAME:Coordinates",             [                 "NAME:Coordinate",                 "X:=", 2,                 "Y:=", 4             ],             [                 "NAME:Coordinate",                 "X:=", 6,                 "Y:=", 8             ]         ]     ] )</pre>

```
)  
oDesign.AddDataset(  
[  
"NAME:$ds1",  
[  
"NAME:Coordinates",  
[  
"NAME:Coordinate",  
"X:=", 2,  
"Y:=", 4  
,  
[  
"NAME:Coordinate",  
"X:=", 6,  
"Y:=", 8  
,  
]  
]  
]  
)
```

**VB Syntax****AddDataset <DatasetdataArray>**

**VB Example**

```
oProject.AddDatasetArray("NAME:ds1",
    Array("NAME:Coordinates",
        Array("NAME:Coordinate", "X:=", 1, "Y:=", 2,
            Array("NAME:Coordinate", "X:=", 3, "Y:=", 4),
            Array("NAME:Coordinate", "X:=", 5, "Y:=", 7),
            Array("NAME:Coordinate", "X:=", 6, "Y:=", 20)))
oDesign.AddDatasetArray("NAME:ds1",
    Array("NAME:Coordinates",
        Array("NAME:Coordinate", "X:=", 1, "Y:=", 2,
            Array("NAME:Coordinate", "X:=", 3, "Y:=", 4),
            Array("NAME:Coordinate", "X:=", 5, "Y:=", 7),
            Array("NAME:Coordinate", "X:=", 6, "Y:=", 20))))
```

## DeleteDataset

Deletes a specified dataset. This can be executed by the oProject, or oDesign variables.

<b>UI Access</b>	<b>Project &gt; Datasets &gt; Remove.</b>		
<b>Parameters</b>	Name <i>&lt;DatasetName&gt;</i>	Type String	Description Name of the dataset found in the project.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	DeleteDataset (<DatasetName>)
<b>Python Example</b>	<pre>oProject.DeleteDataset ('\$ds1') oDesign.DeleteDataset ('\$ds1')</pre>

<b>VB Syntax</b>	DeleteDataset <DatasetName>
<b>VB Example</b>	<pre>oProject.DeleteDataset "\$ds1" oDesign.DeleteDataset "\$ds1"</pre>

## EditDataset

Modifies a dataset. This can be executed by the oProject, or oDesign variables.

<b>UI Access</b>	Project > Datasets > Edit.									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;OriginalName&gt;</td> <td>String</td> <td>Name of the original dataset.</td> </tr> <tr> <td>&lt;DatasetdataArray&gt;</td> <td>Array</td> <td>Data for the modified dataset.</td> </tr> </tbody> </table>	Name	Type	Description	<OriginalName>	String	Name of the original dataset.	<DatasetdataArray>	Array	Data for the modified dataset.
Name	Type	Description								
<OriginalName>	String	Name of the original dataset.								
<DatasetdataArray>	Array	Data for the modified dataset.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	EditDataset (<OriginalName> <DatasetdataArray>)
<b>Python Example</b>	<pre>oProject.EditDataset ("ds1" [ "NAME:ds2",   [ "NAME:Coordinates",</pre>

```
[  
    "NAME:Coordinate",  
    "X:=", 1, "Y:=", 2  
,  
    [  
        "NAME:Coordinate",  
        "X:=", 3, "Y:=", 4  
    ]  
]  
]  
)  
oDesign.EditDataset ("ds1"  
["NAME:ds2",  
    ["NAME:Coordinates",  
        [  
            "NAME:Coordinate",  
            "X:=", 1, "Y:=", 2  
,  
            [  
                "NAME:Coordinate",  
                "X:=", 1, "Y:=", 2  
            ]  
        ]  
    ]  
]
```

```

        "X:=", 3, "Y:=", 4
    ]
]
)

```

<b>VB Syntax</b>	EditDataset < <i>OriginalName</i> > < <i>DatasetdataArray</i> >
<b>VB Example</b>	<pre> oProject.EditDataset "ds1" Array("NAME:ds2",     Array("NAME:Coordinates",Array("NAME:Coordinate",         "X:=", 1, "Y:=", 2), Array("NAME:Coordinate",         "X:=", 3, "Y:=", 4))) oDesign.EditDataset "ds1" Array("NAME:ds2",     Array("NAME:Coordinates",Array("NAME:Coordinate",         "X:=", 1, "Y:=", 2), Array("NAME:Coordinate",         "X:=", 3, "Y:=", 4))) </pre>

## ImportDataset

Imports a dataset from a named file. This can be executed by the oProject, or oDesign variables. The name of the dataset is file-name+index number (e.g., dsdata1) unless the filename ends with a trailing number. When there is a trailing number at the end, we will remove the number and use first unused index. Alternatively, the name of the dataset can be explicitly defined by providing a string as an optional second argument.

**UI Access**

**Project > Datasets > Import.**

<b>Parameters</b>	Name	Type	Description
	<datasetFilePath>	String	The full path to the file containing the dataset values. *.tab files recommended (see <a href="#">note</a> below).
	<optionalDatasetName>	String	Optional. User-defined dataset name.
<b>Return Value</b>	None.		

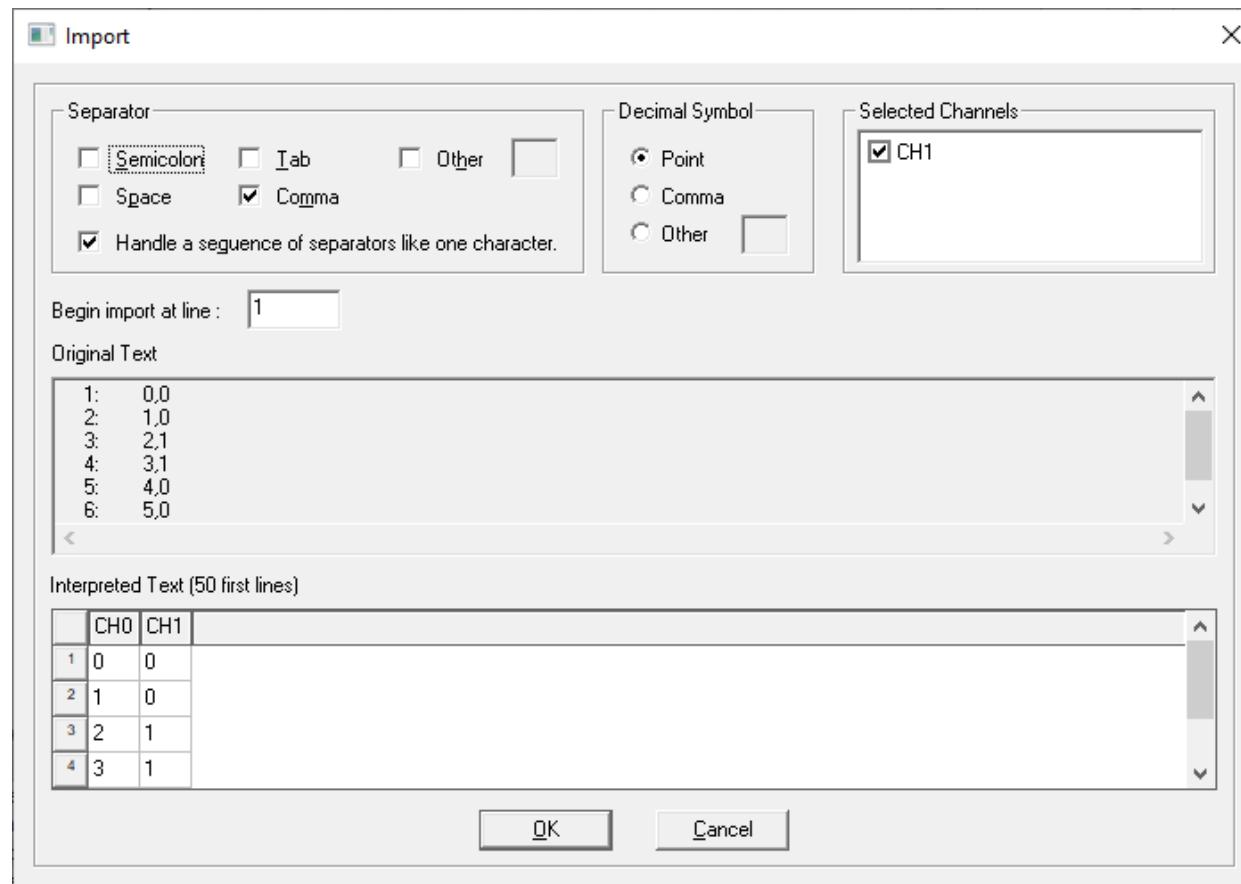
<b>Python Syntax</b>	ImportDataset (<datasetFilePath>,<optionalDatasetName>)
<b>Python Example</b>	<pre>oProject.ImportDataset ('e:\tmp\dsdata.tab') oDesign.ImportDataset ('e:\tmp\dsdata.tab') oProject.ImportDataset ('e:\tmp\dsdata.tab', 'MyDatasetName') oDesign.ImportDataset ('e:\tmp\dsdata.tab', 'MyDatasetName')</pre>

<b>VB Syntax</b>	ImportDataset <datasetFilePath>,<optionalDatasetName>
<b>VB Example</b>	<pre>oProject.ImportDataset "e:\tmp\dsdata.tab" oDesign.ImportDataset "e:\tmp\dsdata.tab" oProject.ImportDataset "e:\tmp\dsdata.tab", "MyDatasetName" oDesign.ImportDataset "e:\tmp\dsdata.tab", "MyDatasetName"</pre>

## Note About File Types:

Tab-delimited or space-delimited files with the extension \*.tab are the recommended file type. When using ImportDataset at the Design level, \*.tab is the only file type supported.

At the Project level, other file types are supported (for example, \*.csv). However, after calling the command, you must configure the file import format manually through the Electronics Desktop GUI by selecting **Project > Datasets** and clicking **Import**.



# 9 - Design Object Script Commands

Design object commands should be executed by the oDesign object.

```
oDesign.CommandName <args>
```

For example:

```
Set oDesign = oProject.SetActiveDesign("MaxwellDesign1")
```

## Conventions Used in this Chapter

<ModuleName> is a placeholder for any one of the following modules:

- [Analysis Module](#) – "AnalysisSetup"
- [Boundary Module](#) – "BoundarySetup"
- [Field Overlays Module](#) – "FieldsReporter"
- [Mesh Module](#) – "MeshSetup"
- [Optimetrics Module](#) – "Optimetrics"
- Radiation Module – "RadField"
- Reduce Matrix Module – "ReduceMatrix"
- [Reporter Module](#) – "ReportSetup"
- Simulation Setup Module – "SimSetup"
- [Solutions Module](#) – "Solutions"

[AnalyzeAll \(Maxwell\)](#)

[AnalyzeDistributed](#)

[ApplyMeshOps](#)

[ClearLinkedData](#)

[ConfigureFluentConductivityCoupling](#)

[ConstructVariationString](#)

[Create3DDesign](#)

[DeleteFieldVariation](#)

[DeleteFullVariation](#)

[DeleteLinkedDataVariation](#)

[EditNotes](#)

[EnableHarmonicForceCalculation](#)

[ExportConvergence](#)

[ExportElementBasedHarmonicForce](#)

[ExportHarmonicTransientForce](#)

ExportMatrixData

[ExportMeshStats](#)

[ExportProfile](#)

[ExportReport](#)

[GenerateMesh](#)

GetActiveEditor

[GetAllPorts](#)

[GetChildNames \[Design\]](#)

[GetChildObject \[Design\]](#)

[GetChildTypes \[Design\]](#)[GetConfigurableData](#)[GetData](#)[GetDesignName](#)[GetDesignType](#)[GetDesignValidationInfo](#)[GetGeometryMode](#)[GetManagedFilesPath](#)[GetModule](#)[GetName](#)[GetNominalVariation](#)[GetNoteText](#)[GetPostProcessingVariables](#)[GetPropNames \[Design\]](#)[GetPropValue \[Design\]](#)[GetSelections](#)[GetSolutionType](#)[InitializeSystemCoupling](#)[Is2D](#)[Is3D](#)[PasteDesign](#)

[Redo](#)

[RenameDesignInstance](#)

[RenameSource](#)

[RunToolkit](#)

[SetActiveEditor](#)

[SetConductivityThreshold](#)

[SetDesignSettings \(Maxwell\)](#)

[SetFastTransformationForLayoutComponent](#)

[SetObjectDeformation](#)

[SetObjectTemperature](#)

[SetPropValue \[Design\]](#)

[SetPropertyValue](#)

[SetShowLayoutForLayoutComponent](#)

[SetSolutionType \(Maxwell\)](#)

[Solve](#)

StartAnalysis

[StopSimLink](#)

[Undo](#)

ValidateCircuit

[ValidateDesign](#)

[ValidateLink](#)

## ApplyMeshOps

If any mesh operations were defined and not yet performed in the current variation for the specified solution setups, they will be applied to the current mesh. If necessary, an initial mesh will be computed first. No further analysis will be performed.

**Important:**

This is a legacy script. Use [GenerateMesh](#) instead.

<b>UI Access</b>	Maxwell > Analysis Setup > Generate Mesh.						
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;SetupNameArray&gt;</td> <td>Array</td> <td>Array containing the string names of setups.</td> </tr> </table>	Name	Type	Description	<SetupNameArray>	Array	Array containing the string names of setups.
Name	Type	Description					
<SetupNameArray>	Array	Array containing the string names of setups.					
<b>Return Value</b>	<p>Integer value:</p> <ul style="list-style-type: none"> <li>• 0 – Success</li> <li>• Else – Error</li> </ul>						

<b>Python Syntax</b>	ApplyMeshOps(<SetupNameArray>)
<b>Python Example</b>	<code>oDesign.ApplyMeshOps ( ['Setup1', 'Setup2'] )</code>

<b>VB Syntax</b>	ApplyMeshOps(<SetupNameArray>)
<b>VB Example</b>	<code>status = oDesign.ApplyMeshOps Array("Setup1", "Setup2")</code>

## Analyze

Solves a single solution setup and all of its frequency sweeps.

<b>UI Access</b>	Right-click a solution setup in the project tree, and select <b>Analyze</b> .						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;setup&gt;</td><td>String</td><td>Setup name.</td></tr></table>	Name	Type	Description	<setup>	String	Setup name.
Name	Type	Description					
<setup>	String	Setup name.					
<b>Return Value</b>	Integer value: <ul style="list-style-type: none"><li>• <b>0</b> – Success</li><li>• <b>Else</b> – Error</li></ul>						

<b>Python Syntax</b>	Analyze (<setup>)
<b>Python Example</b>	<code>oDesign.Analyze("Setup1")</code>

<b>VB Syntax</b>	Analyze <setup>
<b>VB Example</b>	<code>oDesign.Analyze "Setup1"</code>

## AnalyzeAll (Maxwell menu)

Analyzes the nominal problems and all the Optimetrics problems for all the setups under the selected design.

<b>UI Access</b>	Click <b>Maxwell &gt; Analyze All</b> , or right-click a design instance in the project tree and select <b>Analyze All</b> .
------------------	--

---

<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	AnalyzeAll ()
<b>Python Example</b>	<code>oDesign.AnalyzeAll()</code>

<b>VB Syntax</b>	AnalyzeAll
<b>VB Example</b>	<code>oDesign.AnalyzeAll</code>

## AnalyzeDistributed

Performs a distributed analysis.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;SetupName&gt;</code></td> <td>String</td> <td>String name of the setup to be analyzed.</td> </tr> </tbody> </table>	Name	Type	Description	<code>&lt;SetupName&gt;</code>	String	String name of the setup to be analyzed.
Name	Type	Description					
<code>&lt;SetupName&gt;</code>	String	String name of the setup to be analyzed.					
<b>Return Value</b>	<p>Integer value:</p> <ul style="list-style-type: none"> <li>• <b>0</b> – Success</li> <li>• <b>Else</b> – Error</li> </ul>						

<b>Python Syntax</b>	AnalyzeDistributed(<SetupName>)
<b>Python Example</b>	<code>oDesign.AnalyzeDistributed('Setup1')</code>

<b>VB Syntax</b>	AnalyzeDistributed <SetupName>
<b>VB Example</b>	<code>oDesign.AnalyzeDistributed "Setup1"</code>

## ClearLinkedData (Design)

Clear the linked data of all the solution setups. Similar to the ClearLinkedData command for the module level.

<b>UI Access</b>	[Product] > Analysis Setup > Clear Linked Data.
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	ClearLinkedData()
<b>Python Example</b>	<code>oDesign.ClearLinkedData ()</code>

<b>VB Syntax</b>	ClearLinkedData
<b>VB Example</b>	<code>oDesign.ClearLinkedData</code>

## ConfigureFluentConductivityCoupling

*Use:* Configures the directory for the conductivity coupling between a Maxwell 3D magnetostatic design and Ansys Fluent.

*Command:* Right-click on the solution setup under Analysis in the project tree and choose **Configure Fluent Conductivity Coupling**.

**Syntax:** ConfigureFluentConductivityCoupling <SolveSetupName>, <Pathofcouplingdirectory>

**Return Value:** None

**Parameters:** <SolveSetupName>

**Type:** <string>

Name of the solution setup.

<PathOfCouplingDirectory>

**Type:** string

Path of the coupling directory.

**Example:**

```
Set oProject = oDesktop.SetActiveProject("Project18")
Set oDesign = oProject.SetActiveDesign("Maxwell3DDesign1")
Set oModule = oDesign.GetModule("AnalysisSetup")
oModule.ConfigureFluentConductivityCoupling "Setup1",
"C:\Users\test\Documents\Ansys\Maxwell\"
```

## ConstructVariationString

Lists and orders the variables and values associated with a design variation.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<ArrayOfVariableNames>	Array of Strings	List of variable names.
	<ArrayOfVariableValuesIncludingUnits>	Array of Strings	List of variable values, including units, in the same order as the list of names.

<b>Return Value</b>	Returns variation string with the variables ordered to correspond to the order of variables in design variations. The values for the variables are inserted into the variation string. For an example of how <code>ConstructionVariationString</code> can be used, see the Python example.
---------------------	--

<b>Python Syntax</b>	<code>ConstructVariationString(&lt;ArrayOfVariableNames&gt;, &lt;ArrayOfVariableValuesIncludingUnits&gt;)</code>
<b>Python Example</b>	<pre>varStr = oDesign.ConstructVariationString(["xx", "yy"], ["2mm", "1mm"]) oDesign.ExportProfile("Setup1", varStr, "C:\profile.prof")</pre>

<b>VB Syntax</b>	<code>ConstructVariationString &lt;ArrayOfVariableNames&gt;, &lt;ArrayOfVariableValuesIncludingUnits&gt;</code>
<b>VB Example</b>	<pre>oDesign.ConstructVariationString Array("x_size", "y_size"), Array("2mm", "1mm")</pre>

## CreateParametricCircuit

**Note:**

This command is for internal Ansys use only.

<b>Python Syntax</b>	<code>CreateParametricCircuit()</code>
<b>Python Example</b>	<code>oDesign.CreateParametricCircuit()</code>

## Create3DDesign [Maxwell]

*Use:* Create a Maxwell 3D design from a Maxwell 2D design.

*Command:* Click **Maxwell2D>Create 3D Design**, or right-click a Maxwell 2D design instance in the project tree and select **Create 3D Design**.

*Syntax:* Create3DDesign <DesignSettingsArray>

*Return Value:* None

*Parameters:* <DesignSettingsArray>

**For a 2D design in "Cartesian, XY" geometry mode:**

```
Array ("NAME:Parameters",
"ZLength:=", <value>,
"ParametrizeZLength:=", <bool>,
```

**For a 2D design in "Cylindrical about Z" geometry mode:**

```
Array ("NAME:Parameters",
"SweepAngle:=", <value>,
"ParametrizeSweepAngle:=", <bool>,
"DraftAngle:=", <value>,
"DraftType:=", <string>, (one of: "Extended", "Round", "Natural")
"NumberOfSegments:=", <integer>)
```

*Example:*

**For a 2D design in "Cartesian, XY" geometry mode:**

```
oDesign.Create3DDesign Array("NAME:Parameters", "ZLength:=", "12mm", "PararmetrizeZLength:=", true)
```

### For a 2D design in "Cylindrical about Z" geometry mode:

```
oDesign.Create3DDesign Array("NAME:Parameters", "SweepAngle:=", "120deg", "PararmetrizeSweepAngle:=", true, "DraftAngle:=", "30deg", "DraftType:=", "Round", "NumberOfSegments:=", "4")
```

## DeleteFieldVariation

Deletes field variations, fields and meshes, or fields and meshes for specified variations.

UI Access	[Solver] > Results > Clean Up Solutions > [Fields Only / Fields and Meshes].		
Parameters	Name	Type	Description
	<variationKeys>	Array	Array containing either "All" string to select all variations, or strings of variations to include.
	<deleteMesh>	Boolean	When true, deletes mesh data.
Return Value	None.		

Python Syntax	DeleteFieldVariation(<variationKeys>, <deleteMesh>, <deleteLinkedData>)
Python Example	<pre>oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() Design.DeleteFieldVariation(['All'], True, False)</pre>

<b>VB Syntax</b>	DeleteFieldVariation <variationKeys>, <deleteMesh>, <deleteLinkedData>
<b>VB Example</b>	<pre>Set oProject = oDesktop.SetActiveProject "OptimTee" Set oDesign = oProject.SetActiveDesign "TeeModel" oDesign.DeleteFieldVariation Array("offset=" &amp; Chr(39) &amp; "0.0947046688081817in" &amp; Chr(39) &amp; ""), true, false</pre>

## DeleteFullVariation

Deletes either all solution data, or selected variation data.

<b>UI Access</b>	[Maxwell] > Results > Clean Up Solutions.									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;variationKeys&gt;</td> <td>Array</td> <td>Array containing either "All" string to select all variations, or strings of variations to include.</td> </tr> <tr> <td>&lt;deleteLinkedData&gt;</td> <td>Boolean</td> <td>When true, deletes linked data as well.</td> </tr> </tbody> </table>	Name	Type	Description	<variationKeys>	Array	Array containing either "All" string to select all variations, or strings of variations to include.	<deleteLinkedData>	Boolean	When true, deletes linked data as well.
Name	Type	Description								
<variationKeys>	Array	Array containing either "All" string to select all variations, or strings of variations to include.								
<deleteLinkedData>	Boolean	When true, deletes linked data as well.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	DeleteFullVariation(<variationKeys>, <deleteLinkedData>)
<b>Python Example</b>	<code>oDesign.DeleteFullVariation(['All']), false)</code>

<b>VB Syntax</b>	DeleteFullVariation <variationKeys>, <deleteLinkedData>
<b>VB Example</b>	<code>oDesign.DeleteFullVariation Array('All'), false</code>

## DeleteLinkedDataVariation

Deletes the linked data of specified variations.

<b>UI Access</b>	<b>[Maxwell] &gt; Results &gt; Clean Up Solutions.</b>		
<b>Parameters</b>	Name <i>&lt;DesignVariationKeys&gt;</i>	Type Array	Description Array containing strings of the variations keys whose linked data are going to be deleted.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	DeleteLinkedDataVariation(< <i>DesignVariationKeys</i> >)
<b>Python Example</b>	<code>oDesign.DeleteLinkedDataVariation(["current=\\"0.9mA\\\"", "current=\\"1.0mA\\\""])</code>

<b>VB Syntax</b>	DeleteLinkedDataVariation < <i>DesignVariationKeys</i> >
<b>VB Example</b>	<code>oDesign.DeleteLinkedDataVariation Array("current=\\"0.9mA\\\"", "current=\\"1.0mA\\\"")</code>

## DeleteVariation

Deletes a variation.

**Important:**

This script is obsolete. Use [DeleteFullVariation](#), [DeleteFieldVariation](#), or [DeleteLinkedDataVariation](#).

## EditNotes

Updates the notes for the design.

<b>UI Access</b>	<b>Maxwell &gt; Edit Notes.</b>		
<b>Parameters</b>	Name <i>&lt;DesignNotes&gt;</i>	Type String	Description New design notes that are applied, replacing any current notes.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EditNotes(<DesignNotes>)
<b>Python Example</b>	<code>oDesign.EditNotes("My design notes.")</code>

<b>VB Syntax</b>	EditNotes <DesignNotes>
<b>VB Example</b>	<code>oDesign.EditNotes "My design notes."</code>

## EnableHarmonicForceCalculation

*Use:* Sets the objects for harmonic force calculation. Used only for Maxwell designs opened in the Ansys Workbench environment.

*Command:* Click **Maxwell 3D>Enable Harmonic Force Calculation**, **Maxwell 2D>Enable Harmonic Force Calculation**, or right-click a design instance in the project tree and select **Enable Harmonic Force Calculation**. (Commands are present only for Maxwell designs opened in Ansys Workbench.)

*Syntax:* EnableHarmonicForceCalculation <ObjectArray>, <ForceType>, <NumberOfRepeatedWindows>, <UseNumberOfLastCycles>, <NumberOfLastCycles>, <StartTime>, <UseNumberOfCyclesForStopTime>, <NumberOfCyclesForStopTime>, <StopTime>, <CouplingSteps>, <ThirdParty>, <OutputDir>, <OutputFreqRangeType>, <OutputFreqRangeStop>, <OutputFreqRangeStart>, <OutputFreqRangeNum>

*Return Value:* None

*Parameters:* <ObjectArray>

*Type:* <string>

List of object names for force calculations.

For example: `Array("EnabledObjects:=", Array("Magnet", "Steel"))`

`<ForceType>`

*Type: Integer*

0, ObjectBased; 1, Element Based (Surface); 2, Element Based (Volumetric).

`<NumberOfRepeatedWindows>`

*Type: Integer*

The number of repeated windows. Can also be a variable that evaluates to an integer.

`<UseNumberOfLastCycles>`

*Type: Boolean*

Use number Of last cycles for force calculations.

`<NumberOfLastCycles>`

*Type: Integer*

Number of last cycles.

`<StartTime>`

*Type: double*

The start time of data collection.

`<UseNumberOfCyclesForStopTime>`

*Type: Boolean*

User option to use number of cycles or time to define the stop time. True if user wants to use the number of cycles for the stop time; otherwise, False.

<NumberOfCyclesForStopTime>

*Type: string*

The number of cycles after the start time to make the stop time.

<StopTime>

*Type: double*

The specific stop time. Should be greater than the <StartTime>.

<CouplingSteps>

*Type: string*

One of: "All", "Steps from end", "Range of steps"

<ThirdParty>

*Type: string*

Third-party output file type. One of: "LMS", "Romax", "UNV"

<OutputDir>

*Type: string*

The path to the output directory for third-party files.

<WindowFunctionType>

*Type: string*

One of: "Rectangular", "Tri", "Van Hann", "Hamming", "Blackman", "Lanczos", "Welch"

<OutputFreqRangeType>

*Type: string*

One of: "Use Range", "Use Number", "Use All"

<OutputFreqRangeStop>

*Type: double*

The output range stop frequency (with unit).

<OutputFreqRangeStart>

*Type: double*

The output range start frequency (with unit).

<OutputFreqRangeNum>

*Type: Integer*

Number count of frequencies. The number count starts from the start frequency from FFT results.

<CalculateForceType>

*Type: String*

"Harmonic Force"

*VB Example 1:* Sample script for setting EnableHarmonicForceCalculation options for a 2D Transient design with element based force type.

```
oDesign.EnableHarmonicForceCalculation Array(  
    "EnabledObjects:=", Array("Rectangle1", "Rectangle2"),  
    "ForceType:=", 2,  
    "NumberOfRepeatedWindows:=", 3,  
    "UseNumberOfLastCycles:=", true,  
    "NumberOfLastCycles:=", 1,
```

```
"StartTime:=", "0s",
"UseNumberOfCyclesForStopTime:=", true,
"NumberOfCyclesForStopTime:=", 1, "StopTime:=", "0.01s",
"WindowFunctionType:=", "Rectangular")
```

*VB Example 2:*

```
oDesign.EnableHarmonicForceCalculation Array("EnabledObjects:=", Array("Magnet", "Steel")),
"UseNumberOfLastCycles:=", False,
"NumberOfLastCycles:=", 2,
"StartTime:=", "0.002s",
"UseNumberOfCyclesForStopTime:=", False,
"NumberOfCyclesForStopTime:=", 2,
"StopTime:=", "0.003s"
"WindowFunctionType:=", "Rectangular"
```

*Example 3:* Sample Workbench script for setting EnableHarmonicForceCalculation options.

```
SetScriptVersion(Version="16.0")
system1 = GetSystem(Name="Maxwell2DDesign3")
Ansoft.EditSystem(System=system1)
string1 = Ansoft.ExecuteGenericDesktopCommand(
System=system1,
CommandClass="WB_ACTIVATE_GIVEN_SYSTEMID",
Argument="Maxwell",
```

```
ExecuteOnlyIfSystemIsAlreadyOpenInDesktop=True)

system1.SendAnsoftCommand(PyCommand="""oDesktop.SetActiveProject(\"test_project_from_
CADFEM\").SetActiveDesign(\"Maxwell2DDesign3\").EnableHarmonicForceCalculation(
[
\"EnabledObjects:=\", [\"IntegLine_0\", \"IntegLine_0_1\"],
\"UseNumberOfLastCycles:=\", False,
\"NumberOfLastCycles:=\", 2,
\"StartTime:=\", \"0.002s\",
\"UseNumberOfCyclesForStopTime:=\", False,
\"NumberOfCyclesForStopTime:=\", 2,
\"StopTime:=\", \"0.003s\"
\"WindowFunctionType:=\", \"Rectangular\"
])""")
```

*Python Example 4:* Sample script for setting WindowFunctionType options for a Transient design.

```
oDesign.EnableHarmonicForceCalculation(
[
    "EnabledObjects:=", ["Cylinder1", "Cylinder2", "Cylinder3"],
    "ForceType:=", 0,
    "NumberOfRepeatedSamples:=", "1",
    "UseNumberOfLastCycles:=", False,
    "NumberOfLastCycles:=", 1,
```

```
"StartTime:=", "0s",
"UseNumberOfCyclesForStopTime:=", False,
"NumberOfCyclesForStopTime:=", 1,
"StopTime:=", "0.01s",
"WindowFunctionType:=", "Rectangular",
])
```

*Python Example 5:* Sample script for setting OutputFreqRangeType to "Use Range".

```
oDesign.EnableHarmonicForceCalculation(
[
    "EnabledObjects:=", ["Cylinder1", "Cylinder2"],
    "ForceType:=", 0,
    "WindowFunctionType:=", "Rectangular",
    "NumberOfRepeatedSamples:=", "1",
    "UseNumberOfLastCycles:=", False,
    "NumberOfLastCycles:=", 1,
    "StartTime:=", "0s",
    "UseNumberOfCyclesForStopTime:=", False,
    "NumberOfCyclesForStopTime:=", 1,
    "StopTime:=", "0.01s",
    "OutputFreqRangeStart:=", "1Hz",
    "OutputFreqRangeStop:=", "3Hz",
```

```
"OutputFreqRangeType:=" , "Use Range"  
])
```

*Python Example 6:* Sample script for setting OutputFreqRangeType to "Use Number".

```
oDesign.EnableHarmonicForceCalculation(  
[  
    "EnabledObjects:=" , ["Cylinder1","Cylinder2"],  
    "ForceType:=" , 0,  
    "WindowFunctionType:=" , "Rectangular",  
    "NumberOfRepeatedSamples:=" , "1",  
    "UseNumberOfLastCycles:=", False,  
    "NumberOfLastCycles:=" , 1,  
    "StartTime:=" , "0s",  
    "UseNumberOfCyclesForStopTime:=", False,  
    "NumberOfCyclesForStopTime:=" , 1,  
    "StopTime:=" , "0.01s",  
    "OutputFreqRangeStart:=" , "1Hz",  
    "OutputFreqRangeNum:=" , "43",  
    "OutputFreqRangeType:=" , "Use Number"  
])
```

*Python Example 7:* Sample script for setting OutputFreqRangeType to "Use All".

```
oDesign.EnableHarmonicForceCalculation(  
    [  
        "EnabledObjects:=", ["Cylinder1", "Cylinder2"],  
        "ForceType:=", 0,  
        "WindowFunctionType:=", "Rectangular",  
        "NumberOfRepeatedSamples:=", "1",  
        "UseNumberOfLastCycles:=", False,  
        "NumberOfLastCycles:=", 1,  
        "StartTime:=", "0s",  
        "UseNumberOfCyclesForStopTime:=", False,  
        "NumberOfCyclesForStopTime:=", 1,  
        "StopTime:=", "0.01s",  
        "OutputFreqRangeType:=", "Use All"  
    ])
```

*Python Example 8: [Beta] Sample script for setting layout force nets and layers sets.*

```
oDesign.EnableHarmonicForceCalculation(  
    [  
        "ForceType:=", 0,  
        "WindowFunctionType:=", "Rectangular",  
        "NumberOfRepeatedSamples:=", "1",  
        "UseNumberOfLastCycles:=", True,
```

```
"NumberOfLastCycles:=" , 1,
"StartTime:=" , "0s",
"UseNumberOfCyclesForStopTime:=" , True,
"NumberOfCyclesForStopTime:=" , 1,
"StopTime:=" , "0.01s",
"OutputFreqRangeType:=" , "Use All",
"CaculateForceType:=" , "Harmonic Force",
[

"NAME:NetsAndLayersChoices",
[
"NAME:LC1_1",
[
"NAME:NetLayerSetMap",
[
"NAME:net1",
"LayerSet:=" , ["Top"]
],
[
"NAME:net2",
"LayerSet:=" , ["Top"]]
```

```

        ]
        ]
        ]
    ]
})  

...

```

## ExportConvergence

For a given variation, exports convergence data (max mag delta S, E, freq) to \*.conv file.

UI Access	Results > Solution Data. Select Convergence tab and click Export.		
Parameters	Name	Type	Description
	<Setup>	String	Setup name.
	<VariationKeys>	String	The variation. Pass empty string for the current nominal variation.
	<FilePath>	String	Full path to desired *.conv file location.
	<OverwritelfExists>	Boolean	<i>Optional.</i> If True, overwrites any existing file at the same path.
Return Value	None.		

Python Syntax	ExportConvergence(<Setup>, <VariationKeys>, <FilePath> <OverwritelfExists>)
Python Example	oDesign.ExportConvergence('Setup1', 'x_size = 2mm', 'c:\convergence.conv')

VB Syntax	ExportConvergence <Setup>, <VariationKeys>, <FilePath> <OverwritelfExists>
-----------	--

**VB Example**

```
oDesign.ExportConvergence "Setup1", "x_size
= 2mm", "c:\convergence.conv"
```

## ExportElementBasedHarmonicForce

Exports a element-based harmonic force data to a .csv file.

UI Access	Maxwell 3D > Export Transient / Harmonic Force		
<b>Parameters</b>	Name	Type	Description
	<outputDirectory>	String	The path for the output directory.
	<solveSetupName>	String	The name of the solution setup.
	<number1> ,<number2>		Possible values for <i>number1</i> and <i>number2</i> : <ul style="list-style-type: none"> <li>When <i>frequencyOption</i> is 1, <i>number1</i> is -1, <i>number2</i> is -1</li> <li>When <i>frequencyOption</i> is 2, <i>number1</i> is <i>startFrequency</i>&lt;double&gt;, <i>number2</i> is <i>endFrequency</i>&lt;double&gt;</li> <li>When <i>frequencyOption</i> is 3, <i>number1</i> is <i>number of frequencies</i>&lt;int&gt;, <i>number2</i> is -1</li> </ul>
<b>Return Value</b>	None.		

**Python Syntax**

```
ExportElementBasedHarmonicForce(<outputDirectory>, <solveSetupName>, <frequencyOption>, <number1>, <number2>)
```

**Python Example**

```
ExportElementBasedHarmonicForce("E:\\\\tmp", "Setup1", 1, -1, -1)
ExportElementBasedHarmonicForce("E:\\\\tmp", "Setup1", 2, 2.3, 10.5)
```

	<code>ExportElementBasedHarmonicForce("E:\\tmp", "Setup1", 3, 4, -1)</code>
--	---

<b>VB Syntax</b>	<code>ExportElementBasedHarmonicForce &lt;outputDirectory&gt;, &lt;solveSetupName&gt;, &lt;frequencyOption&gt;, &lt;number1&gt;, &lt;number2&gt;</code>
<b>VB Example</b>	<pre>ExportElementBasedHarmonicForce "E:\\tmp", "Setup1", 1, -1, -1 ExportElementBasedHarmonicForce "E:\\tmp", "Setup1", 2, 2.3, 10.5 ExportElementBasedHarmonicForce "E:\\tmp", "Setup1", 3, 4, -1</pre>

## ExportHarmonicTransientForce

Exports harmonic force solution data from transient designs for use by Ansys Motion and third-party software.

UI Access	<b>Maxwell 2D (or 3D)&gt;Export Harmonic/Transient Force...</b>		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;ExportType&gt;</code>	string	One of: <ul style="list-style-type: none"> <li>• UNV</li> <li>• Romax</li> <li>• LMS</li> <li>• Ansys Motion</li> <li>• Ansys Mechanical</li> </ul>
	<code>&lt;ParametricSetupName&gt;</code>	string	The name of the parametric setup whose solution data is to be exported
	<code>&lt;ExportDirectory&gt;</code>	string	The path for the exported files
	<code>&lt;SolverSetupName&gt;</code>	string	The name of the design setup whose solution data is to be exported. If no solution setup is specified, all solve setups under the specified parametric setup will be exported.

	<code>&lt;ExportTimeOption&gt;</code>	integer	Export time steps: 0 for All steps, 1 for Steps from end, 2 for Range of steps.
	<code>&lt;ExportBeginStep&gt;</code>	integer	Integer value for begin step, only valid for <code>ExportTimeOption</code> : Range of steps
	<code>&lt;ExportEndStep&gt;</code>	integer	Integer value for end step, only valid for <code>ExportTimeOption</code> : Range of steps
<b>Return Value</b>	None		

<b>Python Syntax</b>	<code>ExportHarmonicTransientForce (&lt;ExportType&gt;, &lt;ParametricSetupName&gt;, &lt;ExportDirectory&gt;, &lt;SolverSetupName&gt;, &lt;ExportTimeOption&gt;, &lt;ExportBeginStep&gt;, &lt;ExportEndStep&gt;)</code>
<b>Python Example</b>	<code>oDesign.ExportHarmonicTransientForce("Ansys Motion", " ParametricSetup1", "\$PROJECTDIR", "Setup1", 0, -1, -1)</code>

<b>VB Syntax</b>	<code>ExportHarmonicTransientForce &lt;ExportType&gt;, &lt;ParametricSetupName&gt;, &lt;ExportDirectory&gt;, &lt;SolverSetupName&gt;, &lt;ExportTimeOption&gt;, &lt;ExportBeginStep&gt;, &lt;ExportEndStep&gt;</code>
<b>VB Example</b>	<code>oDesign.ExportHarmonicTransientForce "Ansys Motion", "ParametricSetup1", "\$PROJECTDIR", "Setup1", 0, -1, -1</code>

## ExportMeshStats

Exports mesh statistics to a file.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <code>&lt;setupName&gt;</code>	Type String	Description Name of the solution setup.

	<code>&lt;variationString&gt;</code>	String	Name of the variation. An empty string is interpreted as the current nominal variation.
	<code>&lt;filePath&gt;</code>	String	Full file path, including extension *.ms.
	<code>&lt;overwritelfExists&gt;</code>	Boolean	(Optional). Defaults to True. If True, script overwrites any existing file of the same name. If False, no overwrite.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>ExportMeshStats(&lt;setupName&gt;, &lt;variationString&gt;, &lt;filePath&gt;, &lt;overwritelfExists&gt;)</code>
<b>Python Example</b>	<code>oDesign.ExportMeshStats ("Setup1", "radius=3mm", "C:\mydir\meshstats.ms", True)</code>

<b>VB Syntax</b>	<code>ExportMeshStats &lt;setupName&gt;, &lt;variationString&gt;, &lt;filePath&gt;, &lt;overwritelfExists&gt;</code>
<b>VB Example</b>	<code>oDesign.ExportMeshStats "Setup1", "radius=3mm", "C:\mydir\meshstats.ms", true</code>

## ExportProfile

Exports a solution profile to file.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;SetupName&gt;</code>	String	Text of the design's notes.
	<code>&lt;VariationString&gt;</code>	String	Variation name. Pass empty string for the current nominal variation.
	<code>&lt;filePath&gt;</code>	String	Full file path, including extension *.prof.
	<code>&lt;overwritelfExists&gt;</code>	Boolean	Optional. <ul style="list-style-type: none"> <li>• <b>True</b> - overwrites any existing file of the same name.</li> </ul>

			• <b>False</b> - no overwrite.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ExportProfile(<SetupName>, <VariationString>, <filePath>, <overwritelfExists>)
<b>Python Example</b>	<code>oDesign.ExportProfile('Setup1', 'radius=3mm', 'C:\Files\Profile.prof', True)</code>

<b>VB Syntax</b>	ExportProfile <SetupName>, <VariationString>, <filePath>, <overwritelfExists>
<b>VB Example</b>	<code>oDesign.ExportProfile "Setup1", "radius=3mm", "C:\Files\Profile.prof", True</code>

## GetConfigurableData

Obtains configurable data of a specified type

**Note:**

This command is for internal Ansys use only.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;type&gt;</td><td>String</td><td>Specified type.</td></tr></tbody></table>			Name	Type	Description	<type>	String	Specified type.
Name	Type	Description							
<type>	String	Specified type.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	GetConfigurableData(<type>)
<b>Python Example</b>	<code>oDesign.GetConfigurableData ("model")</code>

<b>VB Syntax</b>	GetConfigurableData <type>
<b>VB Example</b>	<code>oDesign.GetConfigurableData "model"</code>

## GetData

**Note:**

This command is for internal Ansys use only.

<b>Python Syntax</b>	GetData()
<b>Python Example</b>	<code>oDesign.GetData()</code>

## GetDesignValidationInfo

Copies a validation cache file that is generated by `oDesign.ValidateDesign()` or doing design validation from UI to a file. This file contains more information besides messages. It also has validation results of some validation check items.

If there is no validation cache file, a file won't be generated. This happens if `GetDesignValidationInfo` is executed and no `ValidateDesign` was performed.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>None</td><td></td><td></td></tr></tbody></table>	Name	Type	Description	None		
Name	Type	Description					
None							
<b>Return Value</b>	Integer 0 if invalid 1 if valid						

<b>Python Syntax</b>	GetDesignValidationInfo()
<b>Python Example</b>	<pre>oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() oDesign.GetDesignValidationInfo ()</pre>

## GetGeometryMode

Returns the geometry mode for the design. For Maxwell 2D only.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>None</td><td></td><td></td></tr></tbody></table>	Name	Type	Description	None		
Name	Type	Description					
None							
<b>Return Value</b>	String "XY" if design is "Cartesian, XY"						

	"about Z" if design is "Cylindrical about Z"
--	--

<b>Python Syntax</b>	GetGeometryMode()
<b>Python Example</b>	<pre> oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() oDesign.GetGeometryMode() </pre>

## GetManagedFilesPath

Get the path to the project's results folder.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String containing path where project results are located.

<b>Python Syntax</b>	oDesign.GetManagedFilesPath()
<b>Python Example</b>	<pre> oDesign.GetManagedFilesPath() </pre>

<b>VB Syntax</b>	oDesign.GetManagedFilesPath
<b>VB Example</b>	<pre> oDesign.GetManagedFilesPath </pre>

## GetModule

Returns the IDispatch for the specified module.

UI Access	N/A		
Parameters	Name <i>&lt;modulename&gt;</i>	Type String	Description One of the following: <ul style="list-style-type: none"><li>• <a href="#">Analysis Module</a> – "AnalysisSetup"</li><li>• <a href="#">Boundary Module</a> – "BoundarySetup"</li><li>• <a href="#">Field Overlays Module</a> – "FieldsReporter"</li><li>• <a href="#">Mesh Module</a> – "MeshSetup"</li><li>• <a href="#">Optimetrics Module</a> – "Optimetrics"</li><li>• Radiation Module – "RadField"</li><li>• Reduce Matrix Module – "ReduceMatrix"</li><li>• <a href="#">Reporter Module</a> – "ReportSetup"</li><li>• Simulation Setup Module – "SimSetup"</li><li>• <a href="#">Solutions Module</a> – "Solutions"</li></ul>
Return Value	Module IDispatch		

Python Syntax	GetModule ( <i>&lt;modulename&gt;</i> )
Python Example	<pre>oModule = oDesign.GetModule("SimSetup")</pre>

---

<b>VB Syntax</b>	GetModule < <i>modulename</i> >
<b>VB Example</b>	set oModule = oDesign.GetModule "SimSetup"

## GetName

Returns the name of the active design.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String indicating the name of the active design.

<b>Python Syntax</b>	GetName()
<b>Python Example</b>	design_name = oDesign.GetName()

<b>VB Syntax</b>	GetName
<b>VB Example</b>	design_name = oDesign.GetName

## GetNominalVariation

Returns the current nominal variation.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String containing current nominal variation.

<b>Python Syntax</b>	GetNominalVariation()
<b>Python Example</b>	<code>oDesign.GetNominalVariation()</code>

<b>VB Syntax</b>	GetNominalVariation
<b>VB Example</b>	<code>oDesign.GetNominalVariation</code>

## GetNoteText

Returns the text of the note attached to a design.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String: text of the design note.

<b>Python Syntax</b>	GetNoteText()
<b>Python Example</b>	<code>oDesign.GetNoteText()</code>

---

<b>VB Syntax</b>	GetNoteText
<b>VB Example</b>	<code>oDesign.GetNoteText</code>

## GetPostProcessingVariables

Returns the list of post-processing variables.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Returns array containing variables.

<b>Python Syntax</b>	GetPostProcessingVariables()
<b>Python Example</b>	<code>oDesign.GetPostProcessingVariables()</code>

<b>VB Syntax</b>	GetPostProcessingVariables
<b>VB Example</b>	<code>oDesign.GetPostProcessingVariables</code>

## GetSelections [Design]

This script serves no function at the Design level. See: GetSelections (Layout Editor), [GetSelections \(Model Editor\)](#), or [Get Selections \(Schematic Editor\)](#).

## GetSolutionType

Returns solution type of the design.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String containing the solution type. Possible values are: "SBR+", "HFSS [Modal   Terminal] [ Network   Composite]", "Transient [Network   Composite]", "Eigenmode", or "Characteristic".

<b>Python Syntax</b>	GetSolutionType()
<b>Python Example</b>	<code>oDesign.GetSolutionType()</code>

<b>VB Syntax</b>	GetSolutionType
<b>VB Example</b>	<code>oDesign.GetSolutionType</code>

## GetVariationVariableValue

Returns the value for a specified variation's variable.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;VariationString&gt;</code>	String	The name of the design variation.
	<code>&lt;VariableName&gt;</code>	String	The name of the variable.

<b>Return Value</b>	Returns a double precision value in SI units, interpreted to mean the value of the variable contained in the variation string.
---------------------	--

<b>Python Syntax</b>	<code>GetVariationVariableValue(&lt;VariationString&gt;, &lt;VariableName&gt;)</code>
<b>Python Example</b>	<code>oDesign.GetVariationVariableValue('x_size = 2mm y_size = 1mm', 'y_size')</code>

<b>VB Syntax</b>	<code>GetVariationVariableValue &lt;VariationString&gt;, &lt;VariableName&gt;</code>
<b>VB Example</b>	<code>varval = oDesign.GetVariationVariableValue "x_size = 2mm y_size = 1mm", "y_size"</code>

## InitializeSystemCoupling

Used for system coupling workflow.

**Note:**

This command is for internal Ansys use only.

If you have a `SystemCouplingSetupn` under Optimetrics and you right-click and select **Generation Configuration Files**, a system coupling configuration (.sp) file and .py file are generated automatically. The .py file contains this line `oDesign.InitializeSystemCoupling("Optimetrics:SystemCouplingSetupn")` and must be preserved.

<b>Python Syntax</b>	<code>InitializeSystemCoupling()</code>
<b>Python Example</b>	<code>oDesign.InitializeSystemCoupling()</code>

## Is2D

**Use:** Determines whether the current design is 2D.

**Command:** None

**Syntax:** Is2D

**Return Value:** Boolean

True if the design is 2D.

**Parameters:** None

**Example:** oDesign.Is2D

## Is3D

**Use:** Determines whether the current design is 3D.

**Command:** None

**Syntax:** Is3D

**Return Value:** Boolean

True if the design is 3D.

**Parameters:** None

**Example:** oDesign.Is3D

## Redo [Design]

Reapplies the last design-level command.

<b>UI Access</b>	Edit > Redo.
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	Redo()
<b>Python Example</b>	<code>oDesign.Redo()</code>

<b>VB Syntax</b>	Redo
<b>VB Example</b>	<code>oDesign.Redo</code>

## RenameDesignInstance

Renames a design instance.

<b>UI Access</b>	Right-click a design instance in the project tree, and then click <b>Rename</b> on the shortcut menu.											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>&lt;OldName&gt;</i></td> <td>String</td> <td>The current name of the design, which must be the design on which this command is invoked.</td> </tr> <tr> <td><i>&lt;NewName&gt;</i></td> <td>String</td> <td>The new name for the design.</td> </tr> </tbody> </table>			Name	Type	Description	<i>&lt;OldName&gt;</i>	String	The current name of the design, which must be the design on which this command is invoked.	<i>&lt;NewName&gt;</i>	String	The new name for the design.
Name	Type	Description										
<i>&lt;OldName&gt;</i>	String	The current name of the design, which must be the design on which this command is invoked.										
<i>&lt;NewName&gt;</i>	String	The new name for the design.										
<b>Return Value</b>	None.											

<b>Python Syntax</b>	RenameDesignInstance (<OldName>, <NewName>)
----------------------	---

**Python Example**

```
oDesign.RenameDesignInstance("Design1", "Design2")
```

**VB Syntax**

```
RenameDesignInstance <OldName>, <NewName>
```

**VB Example**

```
oDesign.RenameDesignInstance "Design1", "Design2"
```

## ResetToTimeZero

Resets a simulation to time zero.

<b>UI Access</b>	<b>CleanStop</b> when running Electronics Desktop in Batchmode.		
<b>Parameters</b>	Name <i>&lt;setupName&gt;</i>	Type String	Description Name of the simulation setup to be reset.
<b>Return Value</b>	None.		

**Python Syntax**

```
ResetToTimeZero(<setupName>)
```

**Python Example**

```
oModule.ResetToTimeZero("Setup1")
```

**VB Syntax**

```
ResetToTimeZero <setupName>
```

**VB Example**

```
oModule.ResetToTimeZero "Setup1"
```

## SetActiveEditor

Sets the active editor.

<b>UI Access</b>	N/A.								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;EditorName&gt;</td> <td>String</td> <td>Text of the design's notes.</td> </tr> </tbody> </table>			Name	Type	Description	<EditorName>	String	Text of the design's notes.
Name	Type	Description							
<EditorName>	String	Text of the design's notes.							
<b>Return Value</b>	Editor object.								

<b>Python Syntax</b>	SetActiveEditor(<EditorName>)
<b>Python Example</b>	<code>oDesign.SetActiveEditor('3D Modeler')</code>

<b>VB Syntax</b>	SetActiveEditor <EditorName>
<b>VB Example</b>	<code>oDesign.SetActiveEditor "3D Modeler"</code>

## SetConductivityThreshold

**Use:** Sets the thresholds for a perfect conductor and an insulator/conductor.

**Command:** Maxwell>Set Material Thresholds or right click on design instance in project tree and choose "Set Material Thresholds"

**Syntax:** SetConductivityThreshold <PerfectConductorThreshold> <Insulator/ConductorThreshold>

**Return Value:** None

**Parameters:** <PerfectConductorThreshold>

Type: <double>

Perfect conductor threshold

<Insulator/ConductorThreshold>

Type: <double>

Insulator/conductor threshold

**Example:** oDesign.SetConductivityThreshold 1E+030, 1.00

## SetDesignSettings [Maxwell]

**Use:** Sets design settings for the active design. Settings vary with solution type.

**Command:** Click **Maxwell3d>Design Settings**, **Maxwell2D>Design Settings**, or right-click a design instance in the project tree and select **Design Settings**.

**Syntax:** SetDesignSettings <DesignSettingsArray>

**Return Value:** None.

**Parameters:** <DesignSettingsArray>

**For 3D Transient solution type:**

```
Array("NAME:Design Settings Data",
      "Allow Material Override:=", <bool>,
      "Perform Minimal validation:=", <bool>,
      "PreservTranSolnAfterDatasetEdit:=", <bool>,
      "ComputeTransientInductance:=", <bool>,
      "ComputeIncrementalMatrix:=", <bool>,
      "PerfectConductorThreshold:=", <real>,
      "InsulatorThreshold:=", <real>,
      "UseSkewModel:=", <bool>,
      "EnableTranTranLinkWithSimplorer:=", <bool>,
```

```
"SolveFraction:=" , <bool>, //True" enables solve using circumferential fraction. Set to "False" by default.  
"FractionNumber:=" , <int>, //Fraction number to generate the partial model. (Used when "Solve Fraction" is "True".) Set to 1 by default.  
"HalfPeriodicField:=" , <bool>, //True" enables half-periodic field. (Used when "Solve Fraction" is "True".) Set to "False" by default.  
"HalfAxialGeometry:=" , <bool>, //True" cuts the original geometry in half in the axial direction. (Used when "Solve Fraction" is "True".) Set to "False" by default.  
"Multiplier:=" , <int>), //Symmetry multiplier value used when "Solve Fraction" is "False".  
"SkipMeshChecks:=" , <bool>  
Array("NAME:Model Validation Settings",  
"EntityCheckLevel:=" , "Strict", //One of "Strict", "None", "WarningOnly", "Basic"  
"IgnoreUnclassifiedObjects:=" , <bool>,  
"SkipIntersectionChecks:=" , <bool>
```

**Example:**

```
oDesign.SetDesignSettings(  
[  
    "NAME:Design Settings Data",  
    "Allow Material Override:=" , True,  
    "Perform Minimal validation:=" , False,  
    "EnabledObjects:=" , [],  
    "PreserveTranSolnAfterDatasetEdit:=" , True,  
    "ComputeTransientInductance:=" , True,
```

```
"ComputeIncrementalMatrix:=", False,  
"PerfectConductorThreshold:=", 1E+30,  
"InsulatorThreshold:=" , 2500000,  
"UseSkewModel:=" , False,  
"EnableTranTranLinkWithSimplorer:=", False,  
"SolveFraction:=" , True,  
"FractionNumber:=" , "4",  
"HalfPeriodicField:=" , True,  
"HalfAxialGeometry:=" , True  
"SkipMeshChecks:=" , False  
,  
[  
    "NAME:Model Validation Settings",  
    "EntityCheckLevel:=" , "Strict",  
    "IgnoreUnclassifiedObjects:=", True,  
    "SkipIntersectionChecks:=", False  
)
```

**For 3D Magnetostatic solution type:**

```
Array("NAME:Design Settings Data",  
    "Allow Material Override:=", <bool>,
```

```
"ComputeIncrementalMatrix:=", <bool>,
"PerfectConductorThreshold:=", <real>,
"InsulatorThreshold:=", <real>,
"SkipMeshChecks:=" , <bool>)
```

**For 3D Eddy Current solution type:**

```
Array("NAME:Design Settings Data",
"Allow Material Override:=", <bool>,
"PerfectConductorThreshold:=", <real>,
"InsulatorThreshold:=", <real>,
"SymmetryMultiplier=", <real>,
"SkipMeshChecks:=" , <bool>)
```

**For 3D Electrostatic solution type:**

```
Array("NAME:Design Settings Data",
"Allow Material Override:=", <bool>,
"PerfectConductorThreshold:=", <real>,
"InsulatorThreshold:=", <real>,
"Multiplier:=", <int>,
"SkipMeshChecks:=" , <bool>)
```

**For 3D DC Conduction solution type:**

```
Array("NAME:Design Settings Data",
"Allow Material Override:=", <bool>,
```

```
"PerfectConductorThreshold:=", <real>,
```

```
"InsulatorThreshold:=", <real>,
```

```
"SkipMeshChecks:=", <bool>)
```

**For 3D Electric Transient solution type:**

```
Array("NAME:Design Settings Data",
```

```
"Allow Material Override:=", <bool>,
```

```
"PreservTranSolnAfterDatasetEdit:=", <bool>,
```

```
"SkipMeshChecks:=", <bool>)
```

**For 3D AC Conduction solution type:**

```
Array("NAME:Design Settings Data",
```

```
"Allow Material Override:=", <bool>,
```

```
"PerfectConductorThreshold:=", <real>,
```

```
"InsulatorThreshold:=", <real>,
```

```
"SkipMeshChecks:=", <bool>)
```

**For 2D Transient solution type:**

```
Array("NAME:Design Settings Data",
```

```
"PreserveTranSolnAfterDatasetEdit:=", <bool>,
```

```
"ComputeTransientInductance:=", <bool>,
```

```
"PerfectConductorThreshold:=", <real>,
```

```
"InsulatorThreshold:=", <real>,
```

```
"ModelDepth:=", <real>,
"UseSkewModel:=", <bool>,
"SkewModelType:=", <string>
    //possible values are: "Continuous", "Step", "V-Shape", "User Defined"
"SkewPart:=", <string>
    //possible values are: "Stator", "Rotor"
"SkewAngle:=", <real>,
    //for "Continuous", "Step", "V-Shape" model types only
"SkewAngleUnit:=", <string>,
    //for "User Defined" model type only; possible values are: "rad", "degsec", "degmin", "deg"
"NumberOfSlices:=", <int>,
//SkewSliceTable array below is for skew model type "User Defined" only.
Array("NAME:SkewSliceTable",
    Array("NAME:OneSliceInfo", "SkewAngle:=", <real>, "SliceLength:=", <real>,
        ...
        Array("NAME:OneSliceInfo", "SkewAngle:=", <real>, "SliceLength:=", <real>, ))
)

"EnableTranTranLinkWithSimplorer:=", <bool>,
"BackgroundMaterialName:=", <string>
"Multiplier:=", <int>

For 2D Magnetostatic solution type:
```

```
Array("NAME:Design Settings Data",
"PerfectConductorThreshold:=", <real>,
"InsulatorThreshold:=", <real>,
"ModelDepth:=", <real>,
"ComputeIncrementalMatrix:=", <bool>,
"BackgroundMaterialName:=", <string>)
```

**For 2D Eddy Current, Electrostatic, DC Conduction, and AC Conduction solution types:**

```
Array("NAME:Design Settings Data",
"PerfectConductorThreshold:=", <real>,
"InsulatorThreshold:=", <real>,
"SymmetryMultiplier:=", <int>,
"ModelDepth:=", <real>,
"BackgroundMaterialName:=", <string>)
```

**For all solution types:**

```
Array("NAME:Validation Options",
"EntityCheckLevel:=", "Strict",
//possible values are: Strict, Basic, Warning Only, None
"IgnoreUnclassifiedObjects:=", <bool>,
"SkipIntersectionChecks:=", <bool>),
"Design Validation Settings:=", "Perform full validations"
```

```
//possible values are: Perform full validations, Perform minimal validations
```

**Example:** oDesign.SetDesignSettings()

```
[  
  "NAME:Design Settings Data",  
  "Perform Minimal validation:=", False,  
  "EnabledObjects:=", [],  
  "PreserveTranSolnAfterDatasetEdit:=", False,  
  "ComputeTransientInductance:=", False,  
  "ComputeIncrementalMatrix:=", False,  
  "PerfectConductorThreshold:=", 1E+30,  
  "InsulatorThreshold:=", 1,  
  "ModelDepth:=", "1mm",  
  "UseSkewModel:=", True,  
  "SkewModelType:=", 0,  
  "SkewAngle:=", "6deg",  
  "NumberOfSlices:=", "5",  
  "EnableTranTranLinkWithSimplorer:=", False,  
  "BackgroundMaterialName:=", "vacuum",  
  "Multiplier:=", "fractions"  
,  
[
```

```
"NAME:Model Validation Settings",
"EntityCheckLevel:=", "Strict",
"IgnoreUnclassifiedObjects:=", True,
"SkipIntersectionChecks:=", False
])
```

***Python Examples: 2D Transient solution type***

```
oDesign.SetDesignSettings(
[
    "NAME:Design Settings Data",
    "Perform Minimal validation:=", False,
    "EnabledObjects:=", [],
    "PreserveTranSolnAfterDatasetEdit:=", False,
    "ComputeTransientInductance:=", False,
    "ComputeIncrementalMatrix:=", False,
    "PerfectConductorThreshold:=", 1E+30,
    "InsulatorThreshold:=", 1,
    "ModelDepth:=", "1meter",
    "UseSkewModel:=", True,
    "SkewType:=", "Continuous",
```

```
"SkewPart:=" , "Stator",
"SkewAngle:=" , "5deg",
"NumberOfSlices:=" , "3",
"EnableTranTranLinkWithSimplorer:=", False,
"BackgroundMaterialName:=", "vacuum",
"SolveFraction:=" , False,
"Multiplier:=" , "1"

]

oDesign.SetDesignSettings(
[

...
"SkewType:=" , "Step",
"SkewPart:=" , "Stator",
"SkewAngle:=" , "0.6rad",
"NumberOfSlices:=" , "2",
...

]
oDesign.SetDesignSettings(
[

...
"SkewType:=" , "V-Shape",
```

```
"SkewPart:=" , "Rotor",
"SkewAngle:=" , "5deg",
"NumberOfSlices:=" , "7",
...
]

oDesign.SetDesignSettings(
[

  .
  .

"SkewType:=" , "User Defined",
"SkewPart:=" , "Rotor",
"SkewAngleUnit:=" , "deg",
"NumberOfSlices:=" , "4",
[
"NAME:SkewSliceTable",
[
"NAME:OneSliceInfo",
"SkewAngle:=" , 0.6,
"SliceLength:=" , 0.1
],
[

```

```
"NAME:OneSliceInfo",
"SkewAngle:=" , -0.9,
"SliceLength:=" , 0.2
],
[
"NAME:OneSliceInfo",
"SkewAngle:=" , 4,
"SliceLength:=" , 0.3
],
[
"NAME:OneSliceInfo",
"SkewAngle:=" , 18,
"SliceLength:=" , 0.4
]
],
...
]
```

**VBExamples: 2D Transient solution type**

```
oDesign.SetDesignSettings
Array("NAME:Design Settings Data",
"Perform Minimal validation:=", false,
"EnabledObjects:=", Array(),
"PreserveTranSolnAfterDatasetEdit:=", false, "ComputeTransientInductance:=", false,
```

```
"ComputeIncrementalMatrix:=", false,
"PerfectConductorThreshold:=", 1E+30,
"InsulatorThreshold:=", 1,
"ModelDepth:=", "1meter",
"UseSkewModel:=", true,
"SkewType:=", "Continuous",
"SkewPart:=", "Rotor",
"SkewAngle:=", "3rad",
"NumberOfSlices:=", "3",
"EnableTranTranLinkWithSimplorer:=", false, "BackgroundMaterialName:=", "vacuum",
"SolveFraction:=", false,
"Multiplier:=", "1"),
Array("NAME:Model Validation Settings",
"EntityCheckLevel:=", "Strict",
"IgnoreUnclassifiedObjects:=", false,
"SkipIntersectionChecks:=", false)

oDesign.SetDesignSettings
Array(
...
"ModelDepth:=", "1meter",
"UseSkewModel:=", true,
"SkewType:=", "Step",
"SkewPart:=", "Stator",
"SkewAngle:=", "3rad",
"NumberOfSlices:=", "4",
...
)
```

```
oDesign.SetDesignSettings
Array(
...
"ModelDepth:=", "1meter",
"UseSkewModel:=", true,
"SkewType:=", "V-Shape",
"SkewPart:=", "Stator",
"SkewAngle:=", "3rad",
"NumberOfSlices:=", "5",
...
)

oDesign.SetDesignSettings
Array(
...
"ModelDepth:=", "1meter", "UseSkewModel:=", true,
"SkewType:=", "User Defined",
"SkewPart:=", "Rotor",
"SkewAngleUnit:=", "deg",
"NumberOfSlices:=", "5",
Array("NAME:SkewSliceTable",
Array("NAME:OneSliceInfo", "SkewAngle:=", "10", "SliceLength:=", "0.3"),
Array("NAME:OneSliceInfo", "SkewAngle:=", "-2.7", "SliceLength:=", "0.2"),
Array("NAME:OneSliceInfo", "SkewAngle:=", "-1.8", "SliceLength:=", "0.25"),
Array("NAME:OneSliceInfo", "SkewAngle:=", "-0.3", "SliceLength:=", "0.15"),
Array("NAME:OneSliceInfo", "SkewAngle:=", "3", "SliceLength:=", "0.1"))
...
)
```

## SetObjectDeformation

Sets the deformation of objects when calling Ansys Electronics Desktop from Ansys Workbench

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;ObjectArray&gt;</i>	Type Array	Description Objects to set.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	SetObjectDeformation( <i>&lt;ObjectArray&gt;</i> )
<b>Python Example</b>	<code>oDesign.SetObjectDeformation(["EnabledObjects:=", , ["Cylinder1", "Sphere1"]])</code>

<b>VB Syntax</b>	SetObjectDeformation <ObjectArray>
<b>VB Example</b>	<code>oDesign.SetObjectDeformation Array("EnabledObjects:=", , Array("Cylinder1", "Sphere1"))</code>

## SetObjectTemperature

Sets the temperature of objects.

<b>UI Access</b>	Maxwell > Set Object Temperature		
<b>Parameters</b>	Name <i>&lt;TemperatureSettings&gt;</i>	Type Array	Description ["NAME": <SettingName>, "IncludeTemperatureDependence:=", <boolean, True to include temperature dependence. False to not include.>,

			"EnableFeedback:=", <boolean, True to enable feedback. False to not enable feedback.>, "Temperatures:=", <Temperatures>]
	<Temperatures>	Array	Objects and temperatures.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	oDesign.SetObjectTemperature(<TemperatureSettings>)
<b>Python Example</b>	<pre>oDesign.SetObjectTemperature(     [         "NAME:TemperatureSettings",         "IncludeTemperatureDependence:=", True,         "EnableFeedback:=", False,         "Temperatures:=", ["RegularPolyhedron1", "22cel", "Polygon1", "22cel"]     ])</pre>

<b>VB Syntax</b>	oDesign.SetObjectTemperature <TemperatureSettings>
<b>VB Example</b>	<pre>oDesign.SetObjectTemperature Array(     "NAME:TemperatureSettings", "IncludeTemperatureDependence:=",     true, "EnableFeedback:=", true, "Temperatures:=", Array("RegularPolyhedron1",     "22cel", "Polygon1", "22cel"))</pre>

## SetSolutionType (Maxwell)

**Use:** Sets the solution type for the design.

**Command:** Click **Maxwell3D>Solution Type**, **Maxwell2D>Solution Type**, or right-click a design instance in the project tree and select **Solution Type**.

**Syntax:** SetSolutionType <SolutionType>, <GeomMode>

**Return Value:** None

**Parameters:** <SolutionType>

Type: <string>

Possible values for 3D designs are: "Magnetostatic", "EddyCurrent", "Transient", "Electrostatic", "DCConduction", "ElectroDCConduction", "ElectricTransient"

Possible values for 2D designs are: "Magnetostatic", "EddyCurrent", "Transient", "Electrostatic", "DCConduction", "ACConduction"

<GeomMode>

Type: <string>

Possible values for 2D geometry mode: "XY", "about Z"

**Example:** oDesign.SetSolutionType "ElectroDCConduction" oDesign.SetSolutionType "Electrostatic", "XY" oDesign.SetSolutionType "EddyCurrent", "about Z"

## Solve

Performs one or more simulation. The next script command will not be executed until the simulation(s) are complete.

**UI Access**

Select solution setup(s). Right-click and select **Analyze**.

<b>Parameters</b>	<table border="1"> <tr> <th>Name</th><th>Type</th><th>Description</th></tr> <tr> <td><code>&lt;SimulationNames&gt;</code></td><td>Array</td><td>Array containing string simulation names.</td></tr> </table>	Name	Type	Description	<code>&lt;SimulationNames&gt;</code>	Array	Array containing string simulation names.
Name	Type	Description					
<code>&lt;SimulationNames&gt;</code>	Array	Array containing string simulation names.					
<b>Return Value</b>	<p>Integer:</p> <ul style="list-style-type: none"> <li>• <b>0</b> – Simulation(s) completed.</li> <li>• <b>1</b> – Simulation error.</li> <li>• <b>-1</b> – Command execution error.</li> </ul>						

<b>Python Syntax</b>	<code>Solve &lt;SimulationNames&gt;</code>
<b>Python Example</b>	<code>oDesign.Solve(['Setup1','Setup2','Setup3'])</code>

<b>VB Syntax</b>	<code>Solve &lt;SimulationNames&gt;</code>
<b>VB Example</b>	<code>oDesign.Solve Array("Setup1", "Setup2", "Setup3")</code>

## StopSimLink

Stops linked simulation.

<b>UI Access</b>	N/A											
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td><code>&lt;simId&gt;</code></td> <td>Integer</td> <td>ID of specified simulation.</td> </tr> <tr> <td><code>&lt;abort&gt;</code></td> <td>Boolean</td> <td>Whether to stop the running simulation:           <ul style="list-style-type: none"> <li>• <b>True</b> - abort the running simulation.</li> <li>• <b>False</b> - do not abort running simulation.</li> </ul> </td> </tr> </table>			Name	Type	Description	<code>&lt;simId&gt;</code>	Integer	ID of specified simulation.	<code>&lt;abort&gt;</code>	Boolean	Whether to stop the running simulation: <ul style="list-style-type: none"> <li>• <b>True</b> - abort the running simulation.</li> <li>• <b>False</b> - do not abort running simulation.</li> </ul>
Name	Type	Description										
<code>&lt;simId&gt;</code>	Integer	ID of specified simulation.										
<code>&lt;abort&gt;</code>	Boolean	Whether to stop the running simulation: <ul style="list-style-type: none"> <li>• <b>True</b> - abort the running simulation.</li> <li>• <b>False</b> - do not abort running simulation.</li> </ul>										
<b>Return Value</b>	None.											

<b>Python Syntax</b>	StopSimLink(<simId>, <abort>)
<b>Python Example</b>	<code>oDesign.StopSimLink(18, True)</code>

<b>VB Syntax</b>	StopSimLink <simId>, <abort>
<b>VB Example</b>	<code>oDesign.StopSimLink 18, true</code>

## RunToolkit

Runs a Python toolkit script, applying it to the active design. The toolkit script itself may have prerequisites, such as sweeps defined, or specific model characteristics, such as port definitions.

### Important:

Full scripting for cable modeling is not supported and no arguments are allowed in the script. The **RunToolkit** command will not automatically create a cable bundle, but will simply open the **Cable Modeling** dialog box. You will have to manually input your parameters.

<b>UI Access</b>	<b>[Maxwell] &gt; Toolkit &gt; [Script Name].</b>		
<b>Parameters</b>	Name	Type	Description
	<LibraryType>	String	Name of the library in which the script is located.
	<ToolkitName>	String	Name of the Python script.
	<ToolkitArg>	Array	Structured array containing arguments required by the toolkit script.

<b>Return Value</b>	User-defined solution(s) and/or output(s).
---------------------	--

<b>Python Syntax</b>	RunToolkit(<LibraryType>, <ToolkitName>, <ToolkitArg>)
<b>Python Example</b>	<pre>oDesign.RunToolkit("SysLib", "Cable Modeling/Automotive Cable Bundle", []) oDesign.RunToolkit("SysLib", "Cable Modeling/Oil-Gas Cable Bundle", [])</pre>

<b>VB Syntax</b>	RunToolkit <LibraryType>, <ToolkitName>, <ToolkitArg>
<b>VB Example</b>	<pre>oDesign.RunToolkit "SysLib", "Cable Modeling/Automotive Cable Bundle", Array() oDesign.RunToolkit "SysLib", "Cable Modeling/Oil-Gas Cable Bundle", Array()</pre>

## Undo [Design]

Cancels the last design-level command.

<b>UI Access</b>	Edit > Undo
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	Undo()
<b>Python Example</b>	<code>oDesign.Undo()</code>

<b>VB Syntax</b>	Undo
<b>VB Example</b>	<code>oDesign.Undo</code>

## ValidateDesign

Returns whether a design is valid.

<b>UI Access</b>	<b>Maxwell &gt; Validation Check.</b>
<b>Parameters</b>	None.
<b>Return Value</b>	Integer: <ul style="list-style-type: none"><li>• <b>1</b> – Validation passed.</li><li>• <b>0</b> – Validation failed.</li></ul>

<b>Python Syntax</b>	<code>ValidateDesign()</code>
<b>Python Example</b>	<code>oDesign.ValidateDesign()</code>

<b>VB Syntax</b>	<code>ValidateDesign</code>
<b>VB Example</b>	<code>oDesign.ValidateDesign</code>

## ValidateLink

**Note:**

This command is for internal Ansys use only.

<b>Python Syntax</b>	ValidateLink()
<b>Python Example</b>	oDesign.ValidateLink()

This page intentionally  
left blank.

# 10 - 3D Modeler Editor Script Commands

3D Modeler commands should be executed by the "3D Modeler" editor:

```
Set oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.<CommandName>
```

## Conventions Used in this Chapter:

### <AttributesArray>

<AttributesArray> takes the following structure:

```
Array("NAME:Attributes",
      "Name:=", <string>,
      "Flags:=", <string>,
      "Color:=", <string>,
      "Transparency:=", <value>,
      "PartCoordinateSystem:=", <string>,
      "UDMId:=", <string>,
      "MaterialValue:=", <string>,
      "SurfaceMaterialValue:=", <string>,
      "Solveinside:=", <boolean>,
      "ShellElement:=", <boolean>,
      "ShellElementThickness:=", <string>,
      "IsMaterialEditable:=", <boolean>,
```

```
"UseMaterialAppearance:=", <boolean>,  
"IsLightweight:=", <boolean>)
```

Where:

- **Flags** – Takes a string containing "NonModel" and/or "Wireframe", separated by the # character. For example, "NonModel#Wireframe".
- **Color** – Takes a string containing an RGB triple, formatted as "<RGB>". For example, "(255 255 255)".
- **Transparency** – Takes a value between 0 and 1.
- **PartCoordinateSystem** – Orientation of the primitive. The name of one of the defined coordinate systems should be specified.
- **UDMId** – Takes a string containing an ID.
- **MaterialValue** – Takes a string of the material name.
- **SurfaceMaterialValue** – Takes a string of the surface material name.
- **Solveinside** – Takes a boolean value.
- **ShellElement** – Takes a boolean value specifying whether or not a shell element is present.
- **ShellElementThickness** – Takes a string containing the shell element thickness. If element is not present, pass empty string.
- **IsMaterialEditable** – Takes a boolean value.
- **IsLightweight** – Takes a boolean value.

## <SelectionsArray>

<SelectionsArray> typically takes the following structure:

```
Array ("NAME:Selections",  
"Selections:=", <string>)
```

Where:

- **Selections** – Takes a comma-separated list of parts on which to perform the action. For example, "Rect1, Rect2, Rect3".

In some cases, <SelectionsArray> takes additional parameters:

```
Array ("NAME:Selections",
      "AllowRegionDependentPartSelectionForPMLCreation:=", <boolean>,
      "AllowRegionSelectionForPMLCreation:=", <boolean>,
      "Selections:=", <string>,
      "NewPartsModelFlag:=", <string>,
      "UseCurrentCS:=", <boolean>)
```

Where:

- **AllowRegionDependentPartSelectionForPMLCreation** – Takes a boolean value. See individual script for whether this parameter is required.
- **AllowRegionSelectionForPMLCreation** – Takes a boolean value. See individual script for whether this parameter is required.
- **Selections** – Takes a comma-separated list of parts on which to perform the action. For example, "Rect1, Rect2, Rect3".
- **NewPartsModelFlag** – Takes either string "Model" or string "Nonmodel". See individual script for whether this parameter is required.
- **UseCurrentCS** – Takes a boolean value. See individual script for whether this parameter is required. Use [GetActiveCoordinateSystem](#) to determine the current CS.

#### Note:

*Selections* is the only parameter required in *all* 3D Modeler Editor scripts. See individual scripts for additional required parameters.

## Organization

3D Modeler editor scripts are organized into the following categories:

[Draw Menu Commands](#)

[Edit Menu Commands](#)

[Modeler Menu Commands](#)

[Cable Modeling Commands](#)

[Other oEditor Commands](#)

## **Draw Menu Commands**

[Create3DComponent](#)

[CreateBondwire](#)

[CreateBox](#)

[CreateCircle](#)

[CreateCone](#)

[CreateCutplane](#)

[CreateCylinder](#)

[CreateEllipse](#)

[CreateEquationCurve](#)

[CreateEquationSurface](#)

[CreateHelix](#)

[CreatePoint](#)

[CreatePolyline](#)

[CreateRectangle](#)  
[CreateRegion](#)  
[CreateRegularPolygon](#)  
[CreateRegularPolyhedron](#)  
[CreateSphere](#)  
[CreateSpiral](#)  
[CreateTorus](#)  
[CreateUserDefinedModel](#)  
[CreateUserDefinedPart](#)  
[Edit3DComponent](#)  
[EditPolyline](#)  
[Get3DComponentDefinitionNames](#)  
[Get3DComponentInstanceNames](#)  
[Get3DComponentMaterialNames](#)  
[Get3DComponentMaterialProperties](#)  
[Get3DComponentParameters](#)  
[Insert3DComponent](#)  
[InsertPolylineSegment](#)  
[SweepAlongPath](#)  
[SweepAlongVector](#)  
[SweepAroundAxis](#)

---

[SweepFacesAlongNormal](#)[SweepFacesAlongNormalWithAttributes](#)[UpdateComponentDefinition](#)

## Create3DComponent

Create a 3D component.

UI Access	Draw>3D Component Library>Create 3D Component		
Parameters	Name	Type	Description
	<NAME>	string	CreateData
	<ComponentName>	string	The name of the component
	<Company>	string	Company name
	<Company URL>	string	Company website address
	<Model Number>	string	Component model designation
	<Help URL>	string	Help link address
	<Version>	string	Component version
	<Notes>	string	Notes about the component
	<IconType>	string	
	<Owner>	string	Owner name
	<Email>	string	Email address
	<Date>	string	Component creation date
	<HasLabel>	bool	
	<IsEncrypted>	bool	Is the component encrypted?
	<AllowEdit>	bool	Permit component to be edited?
	<SecurityMessage>	string	Security information
	<Password>	string	Enter password
	<EditPassword>	string	
	<PasswordType>	string	

---

	<code>&lt;HideContents&gt;</code>	bool	Hide selected contents?
	<code>&lt;ReplaceNames&gt;</code>	bool	Allow replacement of object and material names?
	<code>&lt;ComponentOutline&gt;</code>	string	"None" or "Bounding Box"
	<code>&lt;IncludedParts&gt;</code>	list	Array of included parts
	<code>&lt;HiddenParts&gt;</code>	list	Array of hidden parts
	<code>&lt;IncludedCS&gt;</code>	list	Array of included coordinate system(s)
	<code>&lt;ReferenceCS&gt;</code>	string	Reference coordinate system
	<code>&lt;IncludedParameters&gt;</code>	list	Array of included parameters
	<code>&lt;ParameterDescription&gt;</code>	list	Array of parameter descriptions
	<code>&lt;IsLicensed&gt;</code>	bool	Is the component licensed?
	<code>&lt;LicensingDIIName&gt;</code>	string	License name
	<code>&lt;VendorComponentIdentifier&gt;</code>	string	Vendor component Identifier
	<code>&lt;PublicKeyFile&gt;</code>	string	Public Key filename - including path to the key
	<code>&lt;NAME&gt;</code>	string	
	<code>&lt;Excitations&gt;</code>	list	Array of excitations
	<code>&lt;Filename&gt;</code>	string	Path and name of component being saved.
	<code>&lt;NAME&gt;</code>	string	
	<code>&lt;ImageFile&gt;</code>	string	
<b>Return Value</b>	None		

<b>Python Syntax</b>	Create3DComponent ( <i>ComponentParameters</i> )
<b>Python Example</b>	<pre> oEditor.Create3DComponent ( [     "NAME:CreateData",     "ComponentName:=" , "CoilWindingsNew",     "Company:=" , "",</pre>

```
"Company URL:="           , "",  
"Model Number:="          , "",  
"Help URL:="              , "",  
"Version:="                , "1.0",  
"Notes:="                 , "",  
"IconType:="              , "",  
"Owner:="                  , "J Doe",  
"Email:="                  , "jdoe@",  
"Date:="                   , "4:40:58 PM Oct 14, 2019",  
"HasLabel:="               , False,  
"IsEncrypted:="            , False,  
"AllowEdit:="              , False,  
"SecurityMessage:="        , "",  
"Password:="               , "",  
"EditPassword:="            , "",  
"PasswordType:="           , "UnknownPassword",  
"HideContents:="            , True,  
"ReplaceNames:="            , True,  
"ComponentOutline:="        , "None",  
"IncludedParts:="           , ["Coil_1st_Section1",  
                           "Coil_2nd_Section1"],  
"HiddenParts:="             , [],  
"IncludedCS:="              , [],  
"ReferenceCS:="             , "Global",  
"IncludedParameters:="      , ["nCond", "wCurrent", "wPhase"],  
"ParameterDescription:="    , ["nCond:=", "", "wCurrent:=", "",  
                           "wPhase:=", ""],  
"IsLicensed:="              , False,  
"LicensingDllName:="        , "",  
"VendorComponentIdentifier:=" , "",  
"PublicKeyFile:="            , ""  
],  
[  
"NAME:DesignData",
```

```
lWindingsNew.a3dcomp",
[
    "NAME:ImageFile",
    "ImageFile:="           , ""
])
```

<b>VB Syntax</b>	<b>Create3DComponent (ComponentParameters)</b>
<b>VB Example</b>	<pre>oEditor.Create3DComponent Array("NAME:CreateData", "ComponentName:=", "CoilWindings", "Company:=", _ "", "Company URL:=", "", "Model Number:=", "", "Help URL:=", "", "Version:=", _ "1.0", "Notes:=", "", "IconType:=", "", "Owner:=", "J Doe", "Email:=", "jdoe@", "Date:=", - "4:33:50 PM Oct 14, 2019", "HasLabel:=", false, "IsEncrypted:=", false, "AllowEdit:=", - false, "SecurityMessage:=", "", "Password:=", "", "EditPassword:=", "", "PasswordType:=", - "UnknownPassword", "HideContents:=", true, "ReplaceNames:=", true, "ComponentOutline:=", - "None", "IncludedParts:=", Array("Coil_1st_Section1", "Coil_2nd_Section1"), "HiddenParts:=", Array(), "IncludedCS:=", Array(), "ReferenceCS:=", _ "Global", "IncludedParameters:=", Array("nCond", "wCurrent", "wPhase"), "ParameterDescription:=", Array("nCond:=", _ "", "wCurrent:=", "", "wPhase:=", ""), "IsLicensed:=", false, "LicensingDllName:=", _ "", "VendorComponentIdentifier:=", "", "PublicKeyFile:=", ""), Array("NAME:DesignData",</pre>

```
"Excitations:=", Array( _  
"CoilTerminal1", "CoilTerminal2", "Winding1", "Winding2")), _  
"C:/work/3Dcomponents/CoilWindings.a3dcomp,Array("NAME:ImageFile", "ImageFile:=", "")
```

## CreateBondwire

Creates a bondwire.

UI Access	Draw > Bondwire.		
	Name	Type	Description
Parameters	<Parameters>	Array	<p>Structured array.</p> <pre>         Array("NAME:BondwireParameters",               "WireType:=", &lt;string("JEDEC_4Points", "JEDEC_5Points", or "LOW")&gt;,               "WireDiameter:=", &lt;string&gt;,               "NumSides:=", &lt;value&gt;,               "XPadPos:=", &lt;value&gt;,               "YPadPos:=", &lt;value&gt;,               "ZPadPos:=", &lt;value&gt;,               "XDir:=", &lt;value&gt;,               "YDir:=", &lt;value&gt;,               "ZDir:=", &lt;value&gt;,               "Distance:=", &lt;value&gt;,             </pre>

			<pre>"h1:=", &lt;value&gt;, "h2:=", &lt;value&gt;, "alpha:=", &lt;value&gt;, "beta:=", &lt;value&gt;, "WhichAxis:=", &lt;string("X", "Y", or "Z")&gt;, "ReverseDirection:=", &lt;boolean&gt;)</pre>
	<AttributesArray>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateBondwire(<Parameters>, <Attributes>)
<b>Python Example</b>	<pre>oEditor.CreateBondwire( ["NAME:BondwireParameters",  "WireType:=" , "JEDEC_4Points",  "WireDiameter:=" , "0.025mm",  "NumSides:=" , "6",  "XPadPos:=" , "1.6mm",  "YPadPos:=" , "-0.2mm",  "ZPadPos:=" , "0mm",  "XDir:=" , "-2.2mm",  "YDir:=" , "-1.4mm",  "ZDir:=" , "0mm",</pre>

```
    "Distance:="           , "2.60768096208106mm",
    "h1:="                 , "0.2mm",
    "h2:="                 , "0mm",
    "alpha:="               , "80deg",
    "beta:="               , "0",
    "WhichAxis:="          , "Z",
    "ReverseDirection:="   , True
  ],
["NAME:Attributes",
  "Name:="                , "Bondwire1",
  "Flags:="               , "",
  "Color:="               , "(143 175 143)",
  "Transparency:="         , 0,
  "PartCoordinateSystem:=" , "Global",
  "UDMId:="               , "",
  "MaterialValue:="        , "\\"vacuum\\\"",
  "SurfaceMaterialValue:=" , "\\"\\\"",
  "SolveInside:="          , True,
  "ShellElement:="         , False,
  "ShellElementThickness:=" , "0mm",
```

```

    "IsMaterialEditable:=" , True,
    "UseMaterialAppearance:=", False,
    "IsLightweight:=" , False
]
)

```

VB Syntax	CreateBondwire <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateBondwire Array("NAME:BondwireParameters", "WireType:=", "LOW", "WireDia- meter:=", _ "0.025mm", "NumSides:=", "6", "XPadPos:=", "6mm", "YPadPos:=", "-2.5mm", "ZPadPos:=", _ "0mm", "XDir:=", "-10mm", "YDir:=", "4mm", "ZDir:=", "0mm", "Distance:=", _ "10.770329614269mm", "h1:=", "0.2mm", "h2:=", "0mm", "alpha:=", "80deg", "beta:=", _ "80deg", "WhichAxis:=", "Z", "ReverseDirection:=", false), Array("NAME:Attributes", "Name:=", _ "Bondwire2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", _ "Global", "UDMID:=", "", "MaterialValue:=", "" &amp; Chr(34) &amp; "copper" &amp; Chr(34) &amp; "", "SurfaceMaterialValue:=", _ "" &amp; Chr(34) &amp; "" &amp; Chr(34) &amp; "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa- terialAppearance:=", _ false, "IsLightweight:=", false) </pre>

## CreateBox

Creates a box.

UI Access	Draw > Box.		
<b>Parameters</b>	Name <i>&lt;Parameters&gt;</i>	Type Array	Description Structured array.  Array ("NAME:BoxParameters", "XPosition:=", <string>, "YPosition:=", <string>, "ZPosition:=", <string>, "XSize:="     , <string>, "YSize:="     , <string>, "ZSize:="     , <string>)
	<i>&lt;AttributesArray&gt;</i>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateBox(<Parameters>, <Attributes>)
<b>Python Example</b>	<pre>oEditor.CreateBox(                   [ "NAME:BoxParameters",                   "XPosition:="                   , "0.5mm",</pre>

```
"YPosition:="           , "-6.5mm",
"ZPosition:="          , "0mm",
"XSize:="              , "2mm",
"YSIZE:="              , "1.5mm",
"ZSize:="              , "1.5mm"
],
["NAME:Attributes",
 "Name:="                , "Box3",
 "Flags:="               , "",
 "Color:="               , "(143 175 143)",
 "Transparency:="        , 0,
 "PartCoordinateSystem:=" , "Global",
 "UDMId:="               , "",
 "MaterialValue:="       , "\"copper\"",
 "SurfaceMaterialValue:=" , "\"\"",
 "SolveInside:="          , False,
 "ShellElement:="         , False,
 "ShellElementThickness:=" , "0mm",
 "IsMaterialEditable:="   , True,
 "UseMaterialAppearance:=" , False,
 "IsLightweight:="         , False
```

	]	)
--	---	---

<b>VB Syntax</b>	CreateBox <Parameters>, <Attributes>
<b>VB Example</b>	<pre> oEditor.CreateBox Array("NAME:BoxParameters", "XPosition:=", "0mm", "YPosition:=", _ "4mm", "ZPosition:=", "0mm", "XSize:=", "-4mm", "YSize:=", "1mm", "ZSize:=", _ "2mm"), Array("NAME:Attributes", "Name:=", "Box4", "Flags:=", "", "Color:=", _ "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", "Global", "UDMID:=", _ "", "MaterialValue:=", "" &amp; Chr(34) &amp; "copper" &amp; Chr(34) &amp; "", "Sur- faceMaterialValue:=", _ "" &amp; Chr(34) &amp; "" &amp; Chr(34) &amp; "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, _ "UseMaterialAppearance:=", false, "IsLightweight:=", false) </pre>

## CreateCircle

Creates a circle.

UI Access	Draw > Circle.		
Parameters	Name	Type	Description
	<Parameters>	Array	<p>Structured array.</p> <p>Array ("NAME:CircleParameters",</p>

			<pre>"IsCovered:=", &lt;boolean&gt;, "XCenter:=", &lt;value&gt;, "YCenter:=", &lt;value&gt;, "ZCenter:=", &lt;value&gt;, "Radius:=", &lt;value&gt;, "WhichAxis:=", &lt;string&gt; "NumSegments:=", &lt;string containing integer&gt;)</pre>
	<code>&lt;AttributesArray&gt;</code>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>CreateCircle(&lt;Parameters&gt;, &lt;Attributes&gt;)</code>
<b>Python Example</b>	<pre>oEditor.CreateCircle(     ["NAME:CircleParameters",         "IsCovered:=" , True,         "XCenter:=" , "5.5mm",         "YCenter:=" , "-3mm",         "ZCenter:=" , "0mm",         "Radius:=" , "0.707106781186548mm",         "WhichAxis:=" , "Z",         "NumSegments:=" , "0"     ],     ] ,</pre>

```
[ "NAME:Attributes",
  "Name:=" , "Circle1",
  "Flags:=" , "",
  "Color:=" , "(143 175 143)",
  "Transparency:=" , 0,
  "PartCoordinateSystem:=", "Global",
  "UDMId:=" , "",
  "MaterialValue:=" , "\"copper\"",
  "SurfaceMaterialValue:=" , "\"\"",
  "SolveInside:=" , False,
  "ShellElement:=" , False,
  "ShellElementThickness:=" , "0mm",
  "IsMaterialEditable:=" , True,
  "UseMaterialAppearance:=" , False,
  "IsLightweight:=" , False
])
```

<b>VB Syntax</b>	CreateCircle <Parameters>, <Attributes>
<b>VB Example</b>	oEditor.CreateCircle Array("NAME:CircleParameters", "IsCovered:=", true, "XCenter:=", _

```

e    "7mm", "YCenter:=", "-6mm", "ZCenter:=", "0mm", "Radius:=", "0.5mm", "WhichAxis:=", _
     "Z", "NumSegments:=", "0"), Array("NAME:Attributes", "Name:=", "Circle2", "Flags:=", _
     "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", _
     "Global", "UDMID:=", "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "Sur-
     faceMaterialValue:=", _
     "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _
     false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa-
     terialAppearance:=", _
     false, "IsLightweight:=", false)

```

## CreateCone

Creates a cone.

UI Access	Draw > Cone.								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <p>Array("NAME:ConeParameters",</p> <p>"XCenter:=", &lt;string&gt;,</p> <p>"YCenter:=", &lt;string&gt;,</p> <p>"ZCenter:=", &lt;string&gt;,</p> <p>"WhichAxis:=", &lt;string&gt;,</p> <p>"Height:=", &lt;string&gt;,</p> <p>"BottomRadius:=", &lt;string&gt;,</p> <p>"TopRadius:=", &lt;string&gt;)</p> </td></tr> </tbody> </table>	Name	Type	Description	<Parameters>	Array	<p>Structured array.</p> <p>Array("NAME:ConeParameters",</p> <p>"XCenter:=", &lt;string&gt;,</p> <p>"YCenter:=", &lt;string&gt;,</p> <p>"ZCenter:=", &lt;string&gt;,</p> <p>"WhichAxis:=", &lt;string&gt;,</p> <p>"Height:=", &lt;string&gt;,</p> <p>"BottomRadius:=", &lt;string&gt;,</p> <p>"TopRadius:=", &lt;string&gt;)</p>		
Name	Type	Description							
<Parameters>	Array	<p>Structured array.</p> <p>Array("NAME:ConeParameters",</p> <p>"XCenter:=", &lt;string&gt;,</p> <p>"YCenter:=", &lt;string&gt;,</p> <p>"ZCenter:=", &lt;string&gt;,</p> <p>"WhichAxis:=", &lt;string&gt;,</p> <p>"Height:=", &lt;string&gt;,</p> <p>"BottomRadius:=", &lt;string&gt;,</p> <p>"TopRadius:=", &lt;string&gt;)</p>							

	<code>&lt;AttributesArray&gt;</code>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>CreateCone(&lt;Parameters&gt;, &lt;Attributes&gt;)</code>
<b>Python Example</b>	<pre> oEditor.CreateCone(     [         ["NAME:ConeParameters",             "XCenter:=" , "3mm",             "YCenter:=" , "-4.5mm",             "ZCenter:=" , "0mm",             "WhichAxis:=" , "Z",             "Height:=" , "2.5mm",             "BottomRadius:=" , "2.82842712474619mm",             "TopRadius:=" , "2.23606797749979mm"         ],         ["NAME:Attributes",             "Name:=" , "Cone1",             "Flags:=" , "",             "Color:=" , "(143 175 143)",             "Transparency:=" , 0,             "PartCoordinateSystem:=" , "Global",         ]     ] ) </pre>

```

    "UDMID:="           , "",
    "MaterialValue:="   , "\\"copper\\",
    "SurfaceMaterialValue:=", "\\\",
    "SolveInside:="      , False,
    "ShellElement:="     , False,
    "ShellElementThickness:=", "0mm",
    "IsMaterialEditable:=" , True,
    "UseMaterialAppearance:=" , False,
    "IsLightweight:="     , False
)

```

VB Syntax	CreateCone <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateCone Array("NAME:ConeParameters", "XCenter:=", "3mm", "YCenter:=", _  "-3mm", "ZCenter:=", "0mm", "WhichAxis:=", "Z", "Height:=", "4mm", "BottomRadius:=", _  "1.11803398874989mm", "TopRadius:=", "2.06155281280883mm"), Array("NAME:Attributes",  "Name:=", _  "Cone1", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", _  "Global", "UDMID:=", "", "MaterialValue:=", "" &amp; Chr(34) &amp; "copper" &amp; Chr(34) &amp; "",  "SurfaceMaterialValue:=", _  "" &amp; Chr(34) &amp; "" &amp; Chr(34) &amp; "", "SolveInside:=", false, "ShellElement:=", _  false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa- </pre>

	<code>terialAppearance:=", _ false, "IsLightweight:=", false)</code>
--	--

## CreateCutplane

Creates a cutplane.

UI Access	Draw > Plane.		
<b>Parameters</b>	Name <code>&lt;Parameters&gt;</code>	Type Array	Description Structured array.  <code>Array ("NAME:PlaneParameters",         "PlaneBaseX:=", &lt;string&gt;,         "PlaneBaseY:=", &lt;string&gt;,         "PlaneBaseZ:=", &lt;string&gt;,         "PlaneNormalX:=", &lt;string&gt;,         "PlaneNormalY:=", &lt;string&gt;),         "PlaneNormalZ:=", &lt;string&gt;)</code>
	<code>&lt;AttributesArray&gt;</code>	Array	See: <a href="#">AttributesArray</a> . CreateCutplane only takes the Name and Color attributes.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>CreateCutplane(&lt;Parameters&gt;, &lt;Attributes&gt;)</code>
<b>Python Example</b>	<code>oEditor.CreateCutplane(</code>

```
[ "NAME:PlaneParameters",
  "PlaneBaseX:="           , "-0.6mm",
  "PlaneBaseY:="           , "-0.8mm",
  "PlaneBaseZ:="           , "0mm",
  "PlaneNormalX:="         , "1.2mm",
  "PlaneNormalY:="         , "0.2mm",
  "PlaneNormalZ:="         , "0mm"
],
[ "NAME:Attributes",
  "Name:="                 , "Plane1",
  "Color:="                , "(143 175 143)"
])
```

<b>VB Syntax</b>	CreateCutplane <Parameters>, <Attributes>
<b>VB Example</b>	<pre>oEditor.CreateCutplane Array("NAME:PlaneParameters", "PlaneBaseX:=", "0.6mm", "PlaneBaseY:=", _ "1.2mm", "PlaneBaseZ:=", "0mm", "PlaneNormalX:=", "0.4mm", "PlaneNormalY:=", _ "0.6mm", "PlaneNormalZ:=", "0mm"), Array("NAME:Attributes", "Name:=", "Plane2", "Col- or:=", _ "(143 175 143)")</pre>

## CreateCylinder

Creates a cylinder.

UI Access	Draw > Cylinder.		
<b>Parameters</b>	Name <i>&lt;Parameters&gt;</i>	Type Array	Description Structured array.  Array ("NAME:CylinderParameters", "XCenter:=", <string>, "YCenter:=", <string>, "ZCenter:=", <string>, "Radius:=", <string>, "Height:=", <string>, "WhichAxis:=", <string>, "NumSides:=", <string containing integer>)
	<i>&lt;AttributesArray&gt;</i>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateCylinder(<Parameters>, <Attributes>)
<b>Python Example</b>	<pre>oEditor.CreateCylinder(     [ "NAME:CylinderParameters",         "XCenter:=" , "6mm",</pre>

```
"YCenter:="           , "-4.5mm",
"ZCenter:="           , "0mm",
"Radius:="            , "0.5mm",
"Height:="             , "4.5mm",
"WhichAxis:="          , "Z",
"NumSides:="            , "0"
],
["NAME:Attributes",
 "Name:="                , "Cylinder1",
 "Flags:="               , "",
 "Color:="                , "(143 175 143)",
 "Transparency:="         , 0,
 "PartCoordinateSystem:=", "Global",
 "UDMID:="               , "",
 "MaterialValue:="        , "\"copper\",
 "SurfaceMaterialValue:=", "\\"",
 "SolveInside:="           , False,
 "ShellElement:="          , False,
 "ShellElementThickness:=", "0mm",
 "IsMaterialEditable:="   , True,
 "UseMaterialAppearance:=", False,
```

	"IsLightweight:=" , False ])
--	---------------------------------

VB Syntax	CreateCylinder <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateCylinder Array("NAME:CylinderParameters", "XCenter:=", "1.5mm", "YCenter:=", _ "6.5mm", "ZCenter:=", "0mm", "Radius:=", "1mm", "Height:=", "1mm", "WhichAxis:=", _ "Z", "NumSides:=", "0"), Array("NAME:Attributes", "Name:=", "Cylinder2", "Flags:=", _ "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", _ "Global", "UDMID:=", "", "MaterialValue:=", "" &amp; Chr(34) &amp; "copper" &amp; Chr(34) &amp; "", "Sur-_ faceMaterialValue:=", _ "" &amp; Chr(34) &amp; "" &amp; Chr(34) &amp; "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa-_ terialAppearance:=", _ false, "IsLightweight:=", false) </pre>

## CreateEllipse

Creates an ellipse.

UI Access	Draw > Ellipse.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array.

		<pre>Array ("NAME:EllipseParameters",       "IsCovered:=", &lt;string&gt;,       "XCenter:=", &lt;string&gt;,       "YCenter:=", &lt;string&gt;,       "ZCenter:=", &lt;string&gt;,       "MajRadius:=", &lt;string&gt;,       "Ratio:=", &lt;string&gt;,       "WhichAxis:=", &lt;string&gt;,       "NumSegments:=", &lt;string&gt;)</pre>
	<AttributesArray>	Array Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.	

<b>Python Syntax</b>	CreateEllipse(<Parameters>, <Attributes>)
<b>Python Example</b>	<pre>oEditor.CreateEllipse(     ["NAME:EllipseParameters",      "IsCovered:="           , True,      "XCenter:="              , "0.6mm",      "YCenter:="              , "-0.6mm",      "ZCenter:="              , "0mm",      "MajRadius:="            , "0.2mm",      "Ratio:="                 , "7",</pre>

```
    "WhichAxis:="           , "Z",
    "NumSegments:="         , "0"
],
[ "NAME:Attributes",
  "Name:="                , "Ellipse1",
  "Flags:="               , "",
  "Color:="               , "(143 175 143)",
  "Transparency:="        , 0,
  "PartCoordinateSystem:=", "Global",
  "UDMID:="               , "",
  "MaterialValue:="       , "\"copper\"",
  "SurfaceMaterialValue:=", "\\"",
  "SolveInside:="          , False,
  "ShellElement:="         , False,
  "ShellElementThickness:=", "0mm",
  "IsMaterialEditable:="   , True,
  "UseMaterialAppearance:=", False,
  "IsLightweight:="        , False
])
```

<b>VB Syntax</b>	CreateEllipse < <i>Parameters</i> >, < <i>Attributes</i> >
<b>VB Example</b>	<pre> oEditor.CreateEllipse Array("NAME:EllipseParameters", "IsCovered:=", true, "XCenter:=", _ "-0.4mm", "YCenter:=", "-3.2mm", "ZCenter:=", "0mm", "MajRadius:=", "0.4mm", "Ratio:=", _ "0.5", "WhichAxis:=", "Z", "NumSegments:=", "0"), Array("NAME:Attributes", "Name:=", _ "Ellipse2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", _ "Global", "UDMID:=", "", "MaterialValue:=", "" &amp; Chr(34) &amp; "copper" &amp; Chr(34) &amp; "", "SurfaceMaterialValue:=", _ "" &amp; Chr(34) &amp; "" &amp; Chr(34) &amp; "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa- terialAppearance:=", _ false, "IsLightweight:=", false) </pre>

## CreateEquationCurve

Creates an equation-based curve.

UI Access	Draw > Equation-Based Curve.		
Parameters	Name	Type	Description
	< <i>Parameters</i> >	Array	<p>Structured array.</p> <p>Array ("NAME:EquationBasedCurveParameters",  "XtFunction:=", &lt;string&gt;,  "YtFunction:=", &lt;string&gt;,</p>

			<pre>"ZtFunction:=", &lt;string&gt;, "tStart:=", &lt;string&gt;, "tEnd:=", &lt;string&gt;, "NumOfPointsOnCurve:=", &lt;string&gt;, "Version:=", &lt;integer&gt;), &lt;polylineArray(optional)&gt;</pre>
	<code>&lt;AttributesArray&gt;</code>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateEquationCurve(<Parameters>, <Attributes>)
<b>Python Example</b>	<pre>oEditor.CreateEquationCurve (     [ "NAME:EquationBasedCurveParameters",         "XtFunction:=" , "1",         "YtFunction:=" , "3",         "ZtFunction:=" , "32",         "tStart:=" , "1",         "tEnd:=" , "3",         "NumOfPointsOnCurve:=" , "0",         "Version:=" , 1,     [ "NAME:PolylineXSection",</pre>

```
"XSectionType:=" , "None",
"XSectionOrient:=" , "Auto",
"XSectionWidth:=" , "0",
"XSectionTopWidth:=" , "0",
"XSectionHeight:=" , "0",
"XSectionNumSegments:=" , "0",
"XSectionBendType:=" , "Corner"
]
],
[ "NAME:Attributes",
  "Name:=" , "EquationCurve1",
  "Flags:=" , "",
  "Color:=" , "(143 175 143)",
  "Transparency:=" , 0,
  "PartCoordinateSystem:=" , "Global",
  "UDMId:=" , "",
  "MaterialValue:=" , "\"copper\"",
  "SurfaceMaterialValue:=" , "\"\"",
  "SolveInside:=" , False,
  "ShellElement:=" , False,
  "ShellElementThickness:=" , "0mm",
```

```

    "IsMaterialEditable:=" , True,
    "UseMaterialAppearance:=", False,
    "IsLightweight:=" , False
]
)

```

VB Syntax	CreateEquationCurve <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateEquationCurve Array("NAME:EquationBasedCurveParameters", "XtFunction:=", _ "1", "YtFunction:=", "3", "ZtFunction:=", "32", "tStart:=", "1", "tEnd:=", "3", "NumOfPointsOnCurve:=", _ "0", "Version:=", 1, Array("NAME:PolylineXSection", "XSectionType:=", "None", "XSectionOrient:=", _ "Auto", "XSectionWidth:=", "0", "XSectionTopWidth:=", "0", "XSectionHeight:=", _ "0", "XSectionNumSegments:=", "0", "XSectionBendType:=", "Corner")), Array("NAME:Attributes", "Name:=", _ "EquationCurve2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", "Global", "UDMID:=", "", "MaterialValue:=", "" &amp; Chr(34) &amp; "copper" &amp; Chr(34) &amp; "", "SurfaceMaterialValue:=", "" &amp; Chr(34) &amp; "" &amp; Chr(34) &amp; "", "SolveInside:=", _ false, "ShellElement:=", false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", _ true, "UseMaterialAppearance:=", false, "IsLightweight:=", false) </pre>

## CreateEquationSurface

Creates an equation-based surface.

UI Access	Draw > Equation-Based Surface.		
<b>Parameters</b>	Name <i>&lt;Parameters&gt;</i>	Type Array	Description Structured array.  Array ("NAME:EquationBasedSurfaceParameters", "XuvFunction:=" , <string equation containing Function, Operators and/or quantities _u, _v, or PI>, "YuvFunction:=" , <string equation containing Function, Operators and/or quantities _u, _v, or PI>, "ZuvFunction:=" , <string equation containing Function, Operators and/or quantities _u, _v, or PI>, "uStart:=" , <string>, "uEnd:=" , <string>, "vStart:=" , <string>, "vEnd:=" , <string>, "Version:=" , <integer>)
	<i>&lt;AttributesArray&gt;</i>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateEquationSurface(<Parameters>, <Attributes>)
<b>Python Example</b>	oEditor.CreateEquationSurface (

```
[ "NAME:EquationBasedSurfaceParameters",
    "XuvFunction:="           , "_u",
    "YuvFunction:="           , "_v",
    "ZuvFunction:="           , "sin(_u)+cos(_v)",
    "uStart:="                , "1",
    "uEnd:="                 , "10",
    "vStart:="                , "1",
    "vEnd:="                 , "10",
    "Version:="               , 1
],
[ "NAME:Attributes",
    "Name:="                  , "EquationSurface1",
    "Flags:="                 , "",
    "Color:="                 , "(143 175 143)",
    "Transparency:="          , 0,
    "PartCoordinateSystem:="   , "Global",
    "UDMId:="                 , "",
    "MaterialValue:="          , "\"copper\"",
    "SurfaceMaterialValue:="   , "\"\"",
    "SolveInside:="            , False,
```

```

    "ShellElement:="           , False,
    "ShellElementThickness:=", "0mm",
    "IsMaterialEditable:="   , True,
    "UseMaterialAppearance:=", False,
    "IsLightweight:="         , False
]
)

```

VB Syntax	CreateEquationSurface <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateEquationSurface Array("NAME:EquationBasedSurfaceParameters", "XuvFunction:=", _ "__u", "YuvFunction:=", "__v", "ZuvFunction:=", "sin(__u)+cos(__v)", "uStart:=", "1", "uEnd:=", _ "10", "vStart:=", "1", "vEnd:=", "10", "Version:=", 1), Array("NAME:Attributes", "Name:=", _ "EquationSurface2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", _ 0, "PartCoordinateSystem:=", "Global", "UDMID:=", "", "MaterialValue:=", _ "" &amp; Chr(34) &amp; "copper" &amp; Chr(34) &amp; "", "SurfaceMaterialValue:=", "" &amp; Chr(34) &amp; "" &amp; Chr(34) &amp; "", "SolveInside:=", _ false, "ShellElement:=", false, "ShellElementThickness:=", "0mm", "IsMa- terialEditable:=", _ true, "UseMaterialAppearance:=", false, "IsLightweight:=", false) </pre>

## CreateHelix

Creates a helix based on a sweep of specified objects.

UI Access	Draw > Helix.		
	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
Parameters	<Parameters>	Array	<pre>Array ("NAME:HelixParameters",        "XCenter:=" , &lt;string&gt;,        "YCenter:=" , &lt;string&gt;,        "ZCenter:=" , &lt;string&gt;,        "XStartDir:=" , &lt;string&gt;,        "YStartDir:=" , &lt;string&gt;,        "ZStartDir:=" , &lt;string&gt;,        "NumThread:=" , &lt;string&gt;,        "RightHand:=" , &lt;boolean&gt;,        "RadiusIncrement:=" , &lt;string&gt;,        "Thread:=" , &lt;string&gt;)</pre>
Return Value	None.		

Python Syntax	CreateHelix(<SelectionsArray>, <Parameters>)
---------------	--

<b>Python Example</b>	<pre> oEditor.CreateHelix(     ["NAME:Selections",         "Selections:=" , "EquationSurface2",         "NewPartsModelFlag:=" , "Model"     ],     ["NAME:HelixParameters",         "XCenter:=" , "10000mm",         "YCenter:=" , "40000mm",         "ZCenter:=" , "0mm",         "XStartDir:=" , "0mm",         "YStartDir:=" , "10000mm",         "ZStartDir:=" , "0mm",         "NumThread:=" , "1",         "RightHand:=" , True,         "RadiusIncrement:=" , "0mm",         "Thread:=" , "1mm"     ] ) </pre>
-----------------------	---

<b>VB Syntax</b>	CreateHelix <SelectionsArray>, <Parameters>
<b>VB Example</b>	<pre> oEditor.CreateHelix Array("NAME:Selections", "Selections:=", "EquationSurface1",     "NewPartsModelFlag:=",     ... ) </pre>

e	<pre>"Model"), Array("NAME:HelixParameters", "XCenter:=", "1000mm", "YCenter:=", "10000mm", "ZCenter:=", _ "2.39945573144418mm", "XStartDir:=", "-41000mm", "YStartDir:=", "-10000mm", "ZStartDir:=", _ "-2.39945573144418mm", "NumThread:=", "1", "RightHand:=", true, "RadiusIncrement:=", _ "0mm", "Thread:=", "1mm")</pre>
---	--

## CreatePoint

Creates a point.

UI Access	Draw > Point.		
Parameters	Name <i>&lt;Parameters&gt;</i>	Type Array	Description Structured array.  Array ("NAME:PointParameters", "PointX:=", <value>, "PointY:=", <value>, "PointZ:=", <value>)
	<i>&lt;AttributesArray&gt;</i>	Array	Structured array. See: <a href="#">AttributesArray</a> . CreatePoint takes only the Name and Color attributes.
Return Value	None.		

Python Syntax	CreatePoint(<Parameters>, <Attributes>)
Python Example	oEditor.CreatePoint(

```
[ "NAME:PointParameters",
  "PointX:=" , "0.2mm",
  "PointY:=" , "-0.2mm",
  "PointZ:=" , "0mm"
],
[ "NAME:Attributes",
  "Name:=" , "Point1",
  "Color:=" , "(143 175 143)"
])
```

<b>VB Syntax</b>	CreatePoint <Parameters>, <Attributes>
<b>VB Example</b>	<pre>oEditor.CreatePoint   Array("NAME:PointParameters",     "PointX:=" , "0.2mm",     "PointY:=" , "-0.2mm",     "PointZ:=" , "0mm") ,   Array("NAME:Attributes",</pre>

	<pre>"Name:=" , "Point1", "Color:=" , "(143 175 143)")</pre>
--	--

## CreateUserDefinedPart

Creates a user-defined part.

UI Access	Draw > User-Defined Primitive > [Part].		
Parameters	Name	Type	Description
	<Parameters>	Array	<p>Structured array.</p> <pre>Array ("NAME:UserDefinedPrimitiveParameters",       "DllName:=" , &lt;string&gt;,       "Version:=" , &lt;string&gt;,       "NoOfParameters:=" , &lt;integer&gt;,       "Library:=" , &lt;string&gt;,       &lt;paramVectorArray&gt;)</pre>
	<paramVectorArray>	Array	<p>Structured array containing arrays for each pair:</p> <pre>Array ("NAME:ParamVector",       &lt;pair&gt;, &lt;pair&gt;, &lt;pair&gt;,...)</pre>
	<pair>	Array	<p>Structured array:</p> <pre>Array ("NAME:Pair",       "Name:=" , &lt;string&gt;,</pre>

		"Value:=" , <string>)
<Attributes>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.	

<b>Python Syntax</b>	CreateUserDefinedPart(<Parameters>, <Attributes>)
<b>Python Example</b>	<pre> oEditor.CreateUserDefinedPart(     [         "NAME:UserDefinedPrimitiveParameters",         "DllName:=" , "RMxprt/LapCoil.dll",         "Version:=" , "16.0",         "NoOfParameters:=" , 22,         "Library:=" , "syslib",         [             "NAME:ParamVector",             [                 "NAME:Pair",                 "Name:=" , "DiaGap",                 "Value:=" , "100mm"             ],             [                 "NAME:Pair",                 "Name:=" , "DiaYoke",                 "Value:=" , "20mm"             ],             [                 "NAME:Pair",                 "Name:=" , "LapCoil"             ]         ]     ] ) </pre>

```
"Name:=" , "Length",
"Value:=" , "100mm"
] ,
["NAME:Pair",
"Name:=" , "Skew",
"Value:=" , "0deg"
] ,
["NAME:Pair",
"Name:=" , "Slots",
"Value:=" , "18"
] ,
["NAME:Pair",
"Name:=" , "SlotType",
"Value:=" , "1"
] ,
["NAME:Pair",
"Name:=" , "Hs0",
"Value:=" , "1mm"
] ,
["NAME:Pair",
```

```
"Name:=" , "Hs1",
"Value:=" , "1mm"
] ,
["NAME:Pair",
"Name:=" , "Hs2",
"Value:=" , "10mm"
] ,
["NAME:Pair",
"Name:=" , "Bs0",
"Value:=" , "2.5mm"
] ,
["NAME:Pair",
"Name:=" , "Bs1",
"Value:=" , "8mm"
] ,
["NAME:Pair",
"Name:=" , "Bs2",
"Value:=" , "5mm"
] ,
["NAME:Pair",
"Name:=" , "Rs",
```

```
"Value:=" , "0mm"
] ,
[ "NAME:Pair",
  "Name:=" , "FilletType",
  "Value:=" , "0"
] ,
[ "NAME:Pair",
  "Name:=" , "Layers",
  "Value:=" , "2"
] ,
[ "NAME:Pair",
  "Name:=" , "CoilPitch",
  "Value:=" , "4"
] ,
[ "NAME:Pair",
  "Name:=" , "EndExt",
  "Value:=" , "5mm"
] ,
[ "NAME:Pair",
  "Name:=" , "SpanExt",
```

```
    "Value:=" , "25mm"
] ,
[ "NAME:Pair",
  "Name:=" , "BendAngle",
  "Value:=" , "0deg"
] ,
[ "NAME:Pair",
  "Name:=" , "SegAngle",
  "Value:=" , "10deg"
] ,
[ "NAME:Pair",
  "Name:=" , "LenRegion",
  "Value:=" , "200mm"
] ,
[ "NAME:Pair",
  "Name:=" , "InfoCoil",
  "Value:=" , "0"
]
],
[ "NAME:Attributes",
```

```

    "Name:="                  , "LapCoil1",
    "Flags:="                 , "", ,
    "Color:="                 , "(143 175 143)",
    "Transparency:="          , 0,
    "PartCoordinateSystem:=", "Global",
    "UDMID:="                 , "", ,
    "MaterialValue:="         , "\"copper\"",
    "SurfaceMaterialValue:=", "\\"",
    "SolveInside:="            , False,
    "ShellElement:="           , False,
    "ShellElementThickness:=", "0mm",
    "IsMaterialEditable:="   , True,
    "UseMaterialAppearance:=", False,
    "IsLightweight:="          , False
  ]
)

```

<b>VB Syntax</b>	CreateUserDefinedPart <Parameters>, <Attributes>
<b>VB Example</b>	<pre> oEditor.CreateUserDefinedPart Array("NAME:UserDefinedPrimitiveParameters", "DllName:=",                                   -                                   "SegmentedHelix/RectHelix.dll", "Version:=", "1.0", "NoOfParameters:=", 8, "Library:=", </pre>

```

    -
    "syslib", Array("NAME:ParamVector", Array("NAME:Pair", "Name:=", "RectHeight",
    "Value:=", _ 
    "1mm"), Array("NAME:Pair", "Name:=", "RectWidth", "Value:=", "2mm"), Array("NAME:Pair",
    "Name:=", _ 
    "StartHelixRadius", "Value:=", "10mm"), Array("NAME:Pair", "Name:=", "RadiusChange",
    "Value:=", _ 
    "0mm"), Array("NAME:Pair", "Name:=", "Pitch", "Value:=", "3mm"), Array("NAME:Pair",
    "Name:=", _ 
    "Turns", "Value:=", "2"), Array("NAME:Pair", "Name:=", "SegmentsPerTurn", "Value:=", _ 
    "36"), Array("NAME:Pair", "Name:=", "RightHanded", "Value:=", "1"))), Array("NAME:At-
tributes", "Name:=", _ 
    "RectHelix1", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0,
    "PartCoordinateSystem:=", _ 
    "Global", "UDMID:=", "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "Sur-
faceMaterialValue:=", _ 
    "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _ 
    false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa-
terialAppearance:=", _ 
    false, "IsLightweight:=", false)

```

## CreatePolyline

Creates a polyline.

**UI Access**

**Draw > Line.**

	Name	Type	Description
<b>Parameters</b>	<code>&lt;Parameters&gt;</code>	Array	Structured array.  <code>Array("NAME:PolylineParameters",      "IsPolylineCovered:=", &lt;bool&gt;,      "IsPolylineClosed:=", &lt;bool&gt;,      &lt;PolylinePointsArray&gt;,      &lt;PolylineSegmentsArray&gt;)</code>
	<code>&lt;PolylinePointsArray&gt;</code>	Array	<code>Array("NAME:PolylinePoints", &lt;OnePointArray&gt;,      &lt;OnePointArray&gt;, ...)</code>
	<code>&lt;OnePointArray&gt;</code>	Array	<code>Array("NAME:PLPoint",      "X:=", &lt;value&gt;,      "Y:=", &lt;value&gt;,      "Z:=", &lt;value&gt;))</code>
	<code>&lt;PolylineSegmentsArray&gt;</code>	Array	<code>&lt;PolylineSegmentsArray&gt;       Array("NAME:PolylineSegments",      &lt;OneSegmentArray&gt;, &lt;OneSegmentArray&gt;, ...)</code>
	<code>&lt;OneSegmentArray&gt;</code>	Array	<code>Array("NAME:PLSegment",      "SegmentType:=", &lt;"Line", "Arc", "Spline", or      "AngularArc"&gt;,      "StartIndex:=", &lt;value&gt;,      "NoOfPoints:=", &lt;value&gt;)</code>
	<code>&lt;AttributesArray&gt;</code>	Array	Structured array. See: <a href="#">AttributesArray</a> .

<b>Return Value</b>	None.
---------------------	-------

<b>Python Syntax</b>	CreatePolyline(<Parameters>, <Attributes>)
<b>Python Example</b>	<pre> oEditor.CreatePolyline(     [         ["NAME:PolylineParameters",             "IsPolylineCovered:=" , True,             "IsPolylineClosed:=" , False,         ],         ["NAME:PolylinePoints",             [                 ["NAME:PLPoint",                     "X:=" , "2000mm",                     "Y:=" , "-2000mm",                     "Z:=" , "0mm"                 ],                 ["NAME:PLPoint",                     "X:=" , "-9000mm",                     "Y:=" , "2000mm",                     "Z:=" , "0mm"                 ],                 ["NAME:PLPoint",                     "X:=" , "1000mm",                     "Y:=" , "-14000mm",                 ]             ]         ]     ] ) </pre>

```
    "Z:="                  , "0mm"
]
],
[ "NAME:PolylineSegments",
  [ "NAME:PLSegment",
    "SegmentType:="      , "Line",
    "StartIndex:="        , 0,
    "NoOfPoints:="        , 2
  ],
  [ "NAME:PLSegment",
    "SegmentType:="      , "Line",
    "StartIndex:="        , 1,
    "NoOfPoints:="        , 2
  ]
],
[ "NAME:PolylineXSection",
  "XSectionType:="      , "None",
  "XSectionOrient:="     , "Auto",
  "XSectionWidth:="      , "0mm",
  "XSectionTopWidth:="    , "0mm",
```

```
"XSectionHeight:="      , "0mm",
"XSectionNumSegments:=" , "0",
"XSectionBendType:="   , "Corner"
]] ,
[ "NAME:Attributes",
  "Name:="                  , "Polyline1",
  "Flags:="                 , "",
  "Color:="                 , "(143 175 143)",
  "Transparency:="          , 0,
  "PartCoordinateSystem:=", "Global",
  "UDMId:="                 , "",
  "MaterialValue:="          , "\"copper\"",
  "SurfaceMaterialValue:=", "\\"",
  "SolveInside:="            , False,
  "ShellElement:="           , False,
  "ShellElementThickness:=" , "0mm",
  "IsMaterialEditable:="    , True,
  "UseMaterialAppearance:=" , False,
  "IsLightweight:="          , False
])
```

VB Syntax	CreatePolyline <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreatePolyline Array("NAME:PolylineParameters", "IsPolylineCovered:=", true, "IsPolylineClosed:=", _ false, Array("NAME:PolylinePoints", Array("NAME:PLPoint", "X:=", "40000mm", "Y:=", _ "50000mm", "Z:=", "0mm"), Array("NAME:PLPoint", "X:=", "-150000mm", "Y:=", "-50000mm", "Z:=", _  "0mm")), Array("NAME:PolylineSegments", Array("NAME:PLSegment", "SegmentType:=", "Line", "startIndex:=", _  0, "NoOfPoints:=", 2)), Array("NAME:PolylineXSection", "XSectionType:=", "None", "XSectionOrient:=", _  "Auto", "XSectionWidth:=", "0mm", "XSectionTopWidth:=", "0mm", "XSectionHeight:=", _  "0mm", "XSectionNumSegments:=", "0", "XSectionBendType:=", "Corner")), Array("NAME:Attributes", "Name:=", _  "Polyline2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", _  "Global", "UDMID:=", "", "MaterialValue:=", "" &amp; Chr(34) &amp; "copper" &amp; Chr(34) &amp; "", "Sur- faceMaterialValue:=", _  "" &amp; Chr(34) &amp; "" &amp; Chr(34) &amp; "", "SolveInside:=", false, "ShellElement:=", _  false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa- terialAppearance:=", _  false, "IsLightweight:=", false) </pre>

## CreateRectangle

Creates a rectangle.

UI Access	Draw > Rectangle.		
<b>Parameters</b>	Name <i>&lt;Parameters&gt;</i>	Type Array	Description Structured array.  Array ("NAME:RectangleParameters", "IsCovered:=" , <boolean>, "XStart:=" , <string>, "YStart:=" , <string>, "ZStart:=" , <string>, "Width:=" , <string>, "Height:=" , <string>, "WhichAxis:=" , <string "X", "Y", or "Z">)
	<i>&lt;AttributesArray&gt;</i>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateRectangle(<Parameters>, <Attributes>)
<b>Python Example</b>	<pre>oEditor.CreateRectangle(     [ "NAME:RectangleParameters",         "IsCovered:=" , True,         "XStart:=" , "-80000mm",</pre>

```
"YStart:="           , "-90000mm",
"ZStart:="           , "0mm",
"Width:="            , "20000mm",
"Height:="           , "30000mm",
"WhichAxis:="        , "Z"
],
["NAME:Attributes",
 "Name:="              , "Rectangle1",
 "Flags:="             , "",
 "Color:="             , "(143 175 143)",
 "Transparency:="      , 0,
 "PartCoordinateSystem:=", "Global",
 "UDMId:="             , "",
 "MaterialValue:="     , "\"copper\"",
 "SurfaceMaterialValue:=", "\"\"",
 "SolveInside:="        , False,
 "ShellElement:="       , False,
 "ShellElementThickness:=", "0mm",
 "IsMaterialEditable:=" , True,
 "UseMaterialAppearance:=", False,
```

	"IsLightweight:=" , False ])
--	---------------------------------

VB Syntax	CreateRectangle <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateRectangle Array("NAME:RectangleParameters", "IsCovered:=", true, "XStart:=", _ "10000mm", "YStart:=", "-40000mm", "ZStart:=", "0mm", "Width:=", "40000mm", "Height:=", - "10000mm", "WhichAxis:=", "Z"), Array("NAME:Attributes", "Name:=", "Rectangle2", "Flags:=",  "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", _ "Global", "UDMID:=", "", "MaterialValue:=", "" &amp; Chr(34) &amp; "copper" &amp; Chr(34) &amp; "", "SurfaceMaterialValue:=", _ "" &amp; Chr(34) &amp; "" &amp; Chr(34) &amp; "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMaterialAppearance:=", _ false, "IsLightweight:=", false) </pre>

## CreateRegion

Creates a region containing the design.

UI Access	Draw > Region.		
Parameters	Name <Parameters>	Type Array	Description Structured array.

		<pre>Array("NAME:RegionParameters",       "+XPaddingType:=", &lt;string "Percentage Offset",       "Absolute Offset", or "Absolute Position",       "+XPadding:=", &lt;string X value&gt;,       "-XPaddingType:=", &lt;string "Percentage Offset",       "Absolute Offset", or "Absolute Position",       "-XPadding:=", &lt;string -X value&gt;,       "+YPaddingType:=", &lt;string "Percentage Offset",       "Absolute Offset", or "Absolute Position",       "+YPadding:=", &lt;string Y value&gt;,       "-YPaddingType:=", &lt;string "Percentage Offset",       "Absolute Offset", or "Absolute Position",       "-YPadding:=", &lt;string -Y value&gt;,       "+ZPaddingType:=", &lt;string "Percentage Offset",       "Absolute Offset", or "Absolute Position",       "+ZPadding:=", &lt;string Z value&gt;,       "-ZPaddingType:=", &lt;string "Percentage Offset",       "Absolute Offset", or "Absolute Position",       "-ZPadding:=", &lt;string -Z value&gt;)</pre>
	<b>&lt;AttributesArray&gt;</b>	Array Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.	

<b>Python Syntax</b>	<pre>CreateRegion(&lt;Parameters&gt;, &lt;Attributes&gt;)</pre>
<b>Python Example</b>	<pre> oEditor.CreateRegion(     [         ["NAME:RegionParameters",             "+XPaddingType:=" , "Percentage Offset",             "+XPadding:=" , "0",             "-XPaddingType:=" , "Percentage Offset",             "-XPadding:=" , "0",             "+YPaddingType:=" , "Percentage Offset",             "+YPadding:=" , "0",             "-YPaddingType:=" , "Percentage Offset",             "-YPadding:=" , "0",             "+ZPaddingType:=" , "Percentage Offset",             "+ZPadding:=" , "0",             "-ZPaddingType:=" , "Percentage Offset",             "-ZPadding:=" , "0"         ],         ["NAME:Attributes",             "Name:=" , "Region",             "Flags:=" , "Wireframe#",             "Color:=" , "(143 175 143)",             "Transparency:=" , 0,         ]     ] ) </pre>

```

    "PartCoordinateSystem:=", "Global",
    "UDMId:="                 , "",,
    "MaterialValue:="          , "\"vacuum\"",
    "SurfaceMaterialValue:="   , "\"\"",
    "SolveInside:="            , False,
    "ShellElement:="           , False,
    "ShellElementThickness:="  , "nan ",
    "IsMaterialEditable:="    , True,
    "UseMaterialAppearance:=" , False,
    "IsLightweight:="          , False
  ]
)

```

VB Syntax	CreateRegion <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateRegion Array("NAME:RegionParameters", "+XPaddingType:=", _ "Percentage Offset", "+XPadding:=", "0", "-XPaddingType:=", "Percentage Offset", "-XPad- ding:=", _ "0", "+YPaddingType:=", "Percentage Offset", "+YPadding:=", "0", "-YPaddingType:=", _ "Percentage Offset", "-YPadding:=", "0", "+ZPaddingType:=", "Percentage Offset", "+ZPad- ding:=", _ "0", "-ZPaddingType:=", "Percentage Offset", "-ZPadding:=", "0"), Array("NAME:At- tributes", "Name:=", _ </pre>

```
"Region", "Flags:=", "Wireframe#", "Color:=", "(143 175 143)", "Transparency:=", _  
0, "PartCoordinateSystem:=", "Global", "UDMID:=", "", "MaterialValue:=", _  
"" & Chr(34) & "vacuum" & Chr(34) & "", "SurfaceMaterialValue:=", "" & Chr(34) & "" &  
Chr(34) & "", "SolveInside:=", _  
false, "ShellElement:=", false, "ShellElementThickness:=", "nan ", "IsMa-  
terialEditable:=", _  
true, "UseMaterialAppearance:=", false, "IsLightweight:=", false)
```

## CreateRegularPolyhedron

Creates a regular polyhedron.

UI Access	Draw > Regular Polyhedron.		
Parameters	Name <Parameters>	Type Array	Description Structured array.  Array("NAME:PolyhedronParameters", "XCenter:=" , <string>, "YCenter:=" , <string>, "ZCenter:=" , <string>, "XStart:=" , <string>, "YStart:=" , <string>, "ZStart:=" , <string>, "Height:=" , <string>, "NumSides:=" , <string containing number greater than 2>,

			"WhichAxis:=" , <string "X", "Y", or "Z">)
	<AttributesArray>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateRegularPolyhedron(<Parameters>, <Attributes>)
<b>Python Example</b>	<pre> oEditor.CreateRegularPolyhedron(     [         ["NAME:PolyhedronParameters",             "XCenter:=" , "40000mm",             "YCenter:=" , "-80000mm",             "ZCenter:=" , "0mm",             "XStart:=" , "50000mm",             "YStart:=" , "-70000mm",             "ZStart:=" , "0mm",             "Height:=" , "50000mm",             "NumSides:=" , "8",             "WhichAxis:=" , "Z"         ],         ["NAME:Attributes",             "Name:=" , "RegularPolyhedron1",             "Flags:=" , ""         ]     ] ) </pre>

```

    "Color:="           , "(143 175 143)",
    "Transparency:="   , 0,
    "PartCoordinateSystem:=", "Global",
    "UDMID:="          , "",
    "MaterialValue:="   , "\"copper\"",
    "SurfaceMaterialValue:=", "\",
    "SolveInside:="     , False,
    "ShellElement:="    , False,
    "ShellElementThickness:=", "0mm",
    "IsMaterialEditable:=" , True,
    "UseMaterialAppearance:=", False,
    "IsLightweight:="    , False
  ])
)

```

<b>VB Syntax</b>	CreateRegularPolyhedron <Parameters>, <Attributes>
<b>VB Example</b>	<pre> oEditor.CreateRegularPolyhedron Array("NAME:PolyhedronParameters", "XCenter:=",                                      "-10000mm", "YCenter:=", "-50000mm", "ZCenter:=", "0mm", "XStart:=", "0mm", "YStart:=",                                      "-50000mm", "ZStart:=", "0mm", "Height:=", "90000mm", "NumSides:=", "8", "WhichAxis:=",                                      "Z"), Array("NAME:Attributes", "Name:=", "RegularPolyhedron2", "Flags:=", "", "Color:=",                                      "Transparency:=") </pre>

```

    "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", "Global", "UDMID:=", _
    "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "Sur-
    faceMaterialValue:=", _
    "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _
    false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa-
    terialAppearance:=", _
    false, "IsLightweight:=", false)

```

## CreateRegularPolygon

Creates a regular polygon.

UI Access	Draw > Regular Polygon.								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre>         Array("NAME:RegularPolygonParameters",               "IsCovered:=" , &lt;boolean&gt;,               "XCenter:=" , &lt;string&gt;,               "YCenter:=" , &lt;string&gt;,               "ZCenter:=" , &lt;string&gt;,               "XStart:=" , &lt;string&gt;,               "YStart:=" , &lt;string&gt;,               "ZStart:=" , &lt;string&gt;,               </pre> </td></tr> </tbody> </table>	Name	Type	Description	<Parameters>	Array	<p>Structured array.</p> <pre>         Array("NAME:RegularPolygonParameters",               "IsCovered:=" , &lt;boolean&gt;,               "XCenter:=" , &lt;string&gt;,               "YCenter:=" , &lt;string&gt;,               "ZCenter:=" , &lt;string&gt;,               "XStart:=" , &lt;string&gt;,               "YStart:=" , &lt;string&gt;,               "ZStart:=" , &lt;string&gt;,               </pre>		
Name	Type	Description							
<Parameters>	Array	<p>Structured array.</p> <pre>         Array("NAME:RegularPolygonParameters",               "IsCovered:=" , &lt;boolean&gt;,               "XCenter:=" , &lt;string&gt;,               "YCenter:=" , &lt;string&gt;,               "ZCenter:=" , &lt;string&gt;,               "XStart:=" , &lt;string&gt;,               "YStart:=" , &lt;string&gt;,               "ZStart:=" , &lt;string&gt;,               </pre>							

			"NumSides:=" , <string containing number greater than 2>, "WhichAxis:=" , <string "X", "Y", or "Z")
	<AttributesArray>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateRegularPolygon(<Parameters>, <Attributes>)
<b>Python Example</b>	<pre> oEditor.CreateRegularPolygon(     [         ["NAME:RegularPolygonParameters",             "IsCovered:=" , True,             "XCenter:=" , "-70000mm",             "YCenter:=" , "-100000mm",             "ZCenter:=" , "0mm",             "XStart:=" , "-50000mm",             "YStart:=" , "-80000mm",             "ZStart:=" , "0mm",             "NumSides:=" , "12",             "WhichAxis:=" , "Z"         ],         ["NAME:Attributes",             "Name:=" , "Polygon1",             "Flags:=" , ""         ]     ] ) </pre>

```

    "Color:="           , "(143 175 143)",
    "Transparency:="   , 0,
    "PartCoordinateSystem:=", "Global",
    "UDMID:="          , "",

    "MaterialValue:="   , "\"copper\"",
    "SurfaceMaterialValue:=", "\",
    "SolveInside:="     , False,
    "ShellElement:="    , False,
    "ShellElementThickness:=", "0mm",
    "IsMaterialEditable:=" , True,
    "UseMaterialAppearance:=", False,
    "IsLightweight:="    , False
)

```

<b>VB Syntax</b>	CreateRegularPolygon <Parameters>, <Attributes>
<b>VB Example</b>	<pre> oEditor.CreateRegularPolygon Array("NAME:RegularPolygonParameters", "IsCovered:=", true, "XCenter:=", "-60000mm", "YCenter:=", "40000mm", "ZCenter:=", "0mm", "XStart:=", - "-50000mm", "YStart:=", "50000mm", "ZStart:=", "0mm", "NumSides:=", "8", "WhichAxis:=", - </pre>

```

"Z"), Array("NAME:Attributes", "Name:=", "Polygon2", "Flags:=", "", "Color:=", _  

"(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", "Global", "UDMID:=", _  

"", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "Sur-  

faceMaterialValue:=", _  

"" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _  

false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa-  

terialAppearance:=", _  

false, "IsLightweight:=", false)

```

## CreateSphere

Creates a sphere.

UI Access	Draw > Sphere.		
<b>Parameters</b>	Name <i>&lt;Parameters&gt;</i>	Type Array	Description Structured array.  Array ("NAME:SphereParameters", "XCenter:=", <string>, "YCenter:=", <string>, "ZCenter:=", <string>, "Radius:=", <string>)  <i>&lt;AttributesArray&gt;</i>
	<i>&lt;AttributesArray&gt;</i>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateSphere( <i>&lt;Parameters&gt;</i> , <i>&lt;Attributes&gt;</i> )
----------------------	---

**Python Example**

```
oEditor.CreateSphere(  
    [ "NAME:SphereParameters",  
        "XCenter:=" , "-40000mm",  
        "YCenter:=" , "-130000mm",  
        "ZCenter:=" , "0mm",  
        "Radius:=" , "22360.6797749979mm"  
    ],  
    [ "NAME:Attributes",  
        "Name:=" , "Sphere1",  
        "Flags:=" , "",  
        "Color:=" , "(143 175 143)",  
        "Transparency:=" , 0,  
        "PartCoordinateSystem:=" , "Global",  
        "UDMID:=" , "",  
        "MaterialValue:=" , "\"copper\"",  
        "SurfaceMaterialValue:=" , "\"\"",  
        "SolveInside:=" , False,  
        "ShellElement:=" , False,  
        "ShellElementThickness:=" , "0mm",  
        "IsMaterialEditable:=" , True,
```

```

    "UseMaterialAppearance:=", False,
    "IsLightweight:=", False
]
)

```

<b>VB Syntax</b>	CreateSphere < <i>Parameters</i> >, < <i>Attributes</i> >
<b>VB Example</b>	<pre> oEditor.CreateSphere Array("NAME:SphereParameters", "XCenter:=", "-120000mm", "YCenter:=", _ "90000mm", "ZCenter:=", "0mm", "Radius:=", "1000mm"), Array("NAME:Attributes", "Name:=", _ "Sphere2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", _ "Global", "UDMID:=", "", "MaterialValue:=", "" &amp; Chr(34) &amp; "copper" &amp; Chr(34) &amp; "",_ "SurfaceMaterialValue:=", _ "" &amp; Chr(34) &amp; "" &amp; Chr(34) &amp; "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMaterialAppearance:=", _ false, "IsLightweight:=", false) </pre>

## CreateSpiral

Creates a spiral by sweeping the specified object(s).

<b>UI Access</b>	<b>Draw &gt; Spiral.</b>								
<b>Parameters</b>	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Name</th> <th style="text-align: left; padding: 2px;">Type</th> <th style="text-align: left; padding: 2px;">Description</th> </tr> </thead> <tbody> <tr> <td style="padding: 2px;"><i>&lt;SelectionsArray&gt;</i></td> <td style="padding: 2px;">Array</td> <td style="padding: 2px;">Structured array. See: <a href="#">SelectionsArray</a>.</td> </tr> </tbody> </table>	Name	Type	Description	<i>&lt;SelectionsArray&gt;</i>	Array	Structured array. See: <a href="#">SelectionsArray</a> .		
Name	Type	Description							
<i>&lt;SelectionsArray&gt;</i>	Array	Structured array. See: <a href="#">SelectionsArray</a> .							

	<code>&lt;Parameters&gt;</code>	Array	<p>Structured array.</p> <pre>Array("NAME:SpiralParameters",       "XCenter:=" , &lt;string&gt;,       "YCenter:=" , &lt;string&gt;,       "ZCenter:=" , &lt;string&gt;,       "YStartDir:=" , &lt;string&gt;,       "ZStartDir:=" , &lt;string&gt;,       "NumThread:=" , &lt;string&gt;,       "RightHand:=" , &lt;boolean&gt;,       "RadiusIncrement:=" , &lt;string&gt;)</pre>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>CreateSpiral(&lt;Parameters&gt;, &lt;Attributes&gt;)</code>
<b>Python Example</b>	<pre>oEditor.CreateSpiral(   ["NAME:Selections",     "Selections:="           , "Polygon2",     "NewPartsModelFlag:="     , "Model"   ],   ["NAME:SpiralParameters",     "XCenter:="              , "-70000mm",     "YCenter:="              , "0mm",     "ZCenter:="              , "0mm",     "YStartDir:="             , "Up"   ])</pre>

```

    "YCenter:="           , "5000mm",
    "ZCenter:="           , "0mm",
    "XStartDir:="         , "-6000mm",
    "YStartDir:="         , "-1000mm",
    "ZStartDir:="         , "0mm",
    "NumThread:="         , "1",
    "RightHand:="         , True,
    "RadiusIncrement:="   , "1mm"
  ]
)

```

<b>VB Syntax</b>	CreateSpiral <Parameters>, <Attributes>
<b>VB Example</b>	<pre> oEditor.CreateSpiral Array("NAME:Selections", "Selections:=", "Rectangle2", "NewPartsModelFlag:=", "Model"), Array("NAME:SpiralParameters", "XCenter:=", "3000mm", "YCenter:=", _  "-3000mm", "ZCenter:=", "0mm", "XStartDir:=", "-9000mm", "YStartDir:=", _  "-3000mm", "ZStartDir:=", "0mm", "NumThread:=", "1", "RightHand:=", false, "Radi- usIncrement:=", _  "1mm") </pre>

## CreateTorus

Creates a torus.

UI Access	Draw > Torus.		
<b>Parameters</b>	Name	Type	Description
	<Parameters>	Array	<p>Structured array.</p> <pre>Array ("NAME:TorusParameters",       "XCenter:=" , &lt;string&gt;,       "YCenter:=" , &lt;string&gt;,       "ZCenter:=" , &lt;string&gt;,       "MajorRadius:=" , &lt;string&gt;,       "MinorRadius:=" , &lt;string&gt;,       "WhichAxis:=" , &lt;string "X", "Y", or "Z"&gt;)</pre>
	<AttributesArray>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

Python Syntax	CreateTorus(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateTorus (     [ "NAME:TorusParameters",         "XCenter:=" , "0.6mm",         "YCenter:=" , "-0.6mm",         "ZCenter:=" , "0mm",         "MajorRadius:=" , "0.365028153987289mm",</pre>

```
"MinorRadius:=" , "0.0821854415126694mm",
"WhichAxis:=" , "Z"
],
[ "NAME:Attributes",
  "Name:=" , "Torus1",
  "Flags:=" , "",
  "Color:=" , "(143 175 143)",
  "Transparency:=" , 0,
  "PartCoordinateSystem:=", "Global",
  "UDMID:=" , "",
  "MaterialValue:=" , "\"copper\"",
  "SurfaceMaterialValue:=" , "\"\"",
  "SolveInside:=" , False,
  "ShellElement:=" , False,
  "ShellElementThickness:=" , "0mm",
  "IsMaterialEditable:=" , True,
  "UseMaterialAppearance:=" , False,
  "IsLightweight:=" , False
])
```

**VB  
Syntax**

CreateTorus &lt;Parameters&gt;, &lt;Attributes&gt;

**VB Example**

```

oEditor.CreateTorus Array("NAME:TorusParameters", "XCenter:=", "0.6mm", "YCenter:=", _
"-2mm", "ZCenter:=", "0mm", "MajorRadius:=", "0.365028153987288mm", "MinorRadius:=", _
"0.0821854415126694mm", "WhichAxis:=", "Z"), Array("NAME:Attributes", "Name:=", _
"Torus2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", _
"Global", "UDMID:=", "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", _
"SurfaceMaterialValue:=", _
"" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _
false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMaterialAppearance:=", _
false, "IsLightweight:=", false)

```

**CreateUserDefinedModel**

Creates a user-defined model.

UI Access	Draw > User-Defined Model > [Model].		
Parameters	Name <Parameters>	Type Array	Description Structured array.  Array ("NAME:UserDefinedModelParameters", <definitionArray>, <optionsArray>, <geometryParamsArray>, "DllName:=", <string filepath>,

		<pre>"Library:=", &lt;string&gt;, "Version:=", &lt;string&gt; "ConnectionID:=", &lt;string&gt;)</pre>
<code>&lt;definitionArray&gt;</code>	Array	Structured array containing string "NAME:Definition".
<code>&lt;optionsArray&gt;</code>	Array	Structured array containing string "NAME:Options".
<code>&lt;geometryParamsArray&gt;</code>	Array	<p>Structured array containing arrays for individual parameters:</p> <pre>Array("NAME:GeometryParameters",       Array("NAME:UDMParam",             "Name:=", &lt;string&gt;,             "Value:=", &lt;string&gt;,             "PropType2:=", &lt;integer&gt;,             "PropFlag2:=", &lt;integer&gt;))</pre> <p>Required UDM parameters depend on the UDM being created. To see which properties apply to a UDM, right-click the UDM in the Project Tree and select <b>Properties</b>. Then select the <b>Parameters</b> tab.</p> <p>PropType2 can be any of the following:</p> <ul style="list-style-type: none"> <li>• <b>0</b> – Property takes a string value.</li> <li>• <b>1</b> – Property is a menu option.</li> <li>• <b>2</b> – Property takes a number (integer or double).</li> <li>• <b>3</b> – Property takes a value (numbers, variables, or expressions).</li> <li>• <b>4</b> – Property is a file name.</li> <li>• <b>5</b> – Property corresponds to a check box.</li> <li>• <b>6</b> – Property specifies a 3D position.</li> </ul>

			<p>PropFlag2 can be any of the following:</p> <ul style="list-style-type: none"> <li>• <b>0</b> – No flags</li> <li>• <b>1</b> – Read-only</li> <li>• <b>2</b> – Must be integer</li> <li>• <b>4</b> – Must be real</li> <li>• <b>8</b> – Hidden</li> </ul> <p>PropFlag2 values can be combined. For example, a read-only property that must be an integer would take the value 3. A hidden property that must be real would take the value 12.</p> <p>These values are further described in the <code>UserDefinedPrimitiveStructures.h</code> file included with the installation under <code>AnsysEM[Version]\Win64\UserDefinedPrimitives\Examples\Headers\</code></p>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>CreateUserDefinedModel(&lt;Parameters&gt;)</code>
<b>Python Example</b>	<pre> oEditor.CreateUserDefinedModel (     [ "NAME:UserDefinedModelParameters",         [ "NAME:Definition",             [ "NAME:Options",                 [ "NAME:GeometryParams", </pre>

```
[ "NAME:UDMParam",
  "Name:=" , "ILD Thickness (ILD)",
  "Value:=" , "0.006mm",
  "PropType2:=" , 3,
  "PropFlag2:=" , 4
] ,
[ "NAME:UDMParam",
  "Name:=" , "Line Spacing (LS)",
  "Value:=" , "0.004mm",
  "PropType2:=" , 3,
  "PropFlag2:=" , 4
] ,
[ "NAME:UDMParam",
  "Name:=" , "Line Thickness (LT)",
  "Value:=" , "0.005mm",
  "PropType2:=" , 3,
  "PropFlag2:=" , 4
] ,
[ "NAME:UDMParam",
  "Name:=" , "Line Width (LW)",
  "Value:=" , "0.004mm",
```

```
"PropType2:=" , 3,  
"PropFlag2:=" , 4  
,  
["NAME:UDMParam",  
"Name:=" , "No. of Turns (N)",  
"Value:=" , "2",  
"PropType2:=" , 3,  
"PropFlag2:=" , 2  
,  
["NAME:UDMParam",  
"Name:=" , "Outer Diameter (OD)",  
"Value:=" , "0.15mm",  
"PropType2:=" , 3,  
"PropFlag2:=" , 4  
,  
["NAME:UDMParam",  
"Name:=" , "Substrate Thickness",  
"Value:=" , "0.2mm",  
"PropType2:=" , 3,  
"PropFlag2:=" , 4
```

```
        ] ,  
  
        [ "NAME:UDMParam",  
          "Name:=" , "Inductor Type",  
          "Value:=" , "\\"Square,Square,Octagonal,Circular,Square-Differential,Octagonal-  
Differential,Circular-Differential\\\"",  
          "DataType:=" , "String",  
          "PropType2:=" , 1,  
          "PropFlag2:=" , 0  
        ] ,  
  
        [ "NAME:UDMParam",  
          "Name:=" , "Underpass Thickness (UT)",  
          "Value:=" , "0.001mm",  
          "PropType2:=" , 3,  
          "PropFlag2:=" , 4  
        ] ,  
  
        [ "NAME:UDMParam",  
          "Name:=" , "Via Thickness (VT)",  
          "Value:=" , "0.001mm",  
          "PropType2:=" , 3,  
          "PropFlag2:=" , 4  
        ]  
      ] ,
```

```

    "DllName:="           , "Maxwell3D/OnDieSpiralInductor.py",
    "Library:="          , "syslib",
    "Version:="          , "2.0",
    "ConnectionID:="     , ""
  ]
)

```

VB Syntax	CreateUserDefinedModel <Parameters>
VB Example	<pre> oEditor.CreateUserDefinedModel Array("NAME:UserDefinedModelParameters", Array ("NAME:Definition"), Arra ("NAME:Options"), Array("NAME:GeometryParams", Array("NAME:UDMParam", "Name:=", "ILD Thickness (ILD)", "Value:=", "0.006mm", "PropType2:=", 3, "PropFlag2:=", 4), Array("NAME:UDMParam", "Name:=", "Line Spacing (LS)", "Value:=", "0.004mm", "PropType2:=",  3, "PropFlag2:=", 4), Array("NAME:UDMParam", "Name:=", "Line Thickness (LT)", "Value:=",  "0.005mm", "PropType2:=", 3, "PropFlag2:=", 4), Array("NAME:UDMParam", "Name:=", "Line Width (LW)", "Value:=", "0.004mm", "PropType2:=", 3, "PropFlag2:=", 4), Array ("NAME:UDMParam", "Name:=", "No. of Turns (N)", "Value:=", "2", "PropType2:=", 3, "PropFlag2:=", 2), Array ("NAME:UDMParam", "Name:=", </pre>

```

    -
    "Outer Diameter (OD)", "Value:=", "0.15mm", "PropType2:=", 3, "PropFlag2:=", 4), Array
    ("NAME:UDMParam",
    "Name:=", _
    "Substrate Thickness", "Value:=", "0.2mm", "PropType2:=", 3, "PropFlag2:=", 4), Array
    ("NAME:UDMParam",
    "Name:=", _
    "Inductor Type", "Value:=", _
    "" & Chr(34) & "Square,Square,Octagonal,Circular,Square-Differential,Octagonal-" & _
    "Differential,Circular-Differential" & Chr(34) & "", "DataType:=", "String",
    "PropType2:=", _
    1, "PropFlag2:=", 0), Array("NAME:UDMParam", "Name:=", "Underpass Thickness (UT)",
    "Value:=", _
    "0.001mm", "PropType2:=", 3, "PropFlag2:=", 4), Array("NAME:UDMParam", "Name:=", _
    "Via Thickness (VT)", "Value:=", "0.001mm", "PropType2:=", 3, "PropFlag2:=", 4),
    "DllName:=", _
    "Maxwell3D/OnDieSpiralInductor.py", "Library:=", "syslib", "Version:=", "2.0", "Con-
    nectionID:=", _
    ""))

```

## CreateUserDefinedPart

Creates a user-defined part.

**UI Access**

**Draw > User-Defined Primitive > [Part].**

	Name	Type	Description
<b>Parameters</b>	<i>&lt;Parameters&gt;</i>	Array	Structured array.  Array ("NAME:UserDefinedPrimitiveParameters", "DllName:=" , <string>, "Version:=" , <string>, "NoOfParameters:=" , <integer>, "Library:=" , <string>, <paramVectorArray>)
	<i>&lt;paramVectorArray&gt;</i>	Array	Structured array containing arrays for each pair:  Array ("NAME:ParamVector", <pair>, <pair>, <pair>,... )
	<i>&lt;pair&gt;</i>	Array	Structured array:  Array ("NAME:Pair", "Name:=" , <string>, "Value:=" , <string>)
	<i>&lt;Attributes&gt;</i>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateUserDefinedPart(<Parameters>, <Attributes>)
<b>Python Example</b>	oEditor.CreateUserDefinedPart (

```
[ "NAME:UserDefinedPrimitiveParameters",
  "DllName:=" , "RMxprt/LapCoil.dll",
  "Version:=" , "16.0",
  "NoOfParameters:=" , 22,
  "Library:=" , "syslib",
  [ "NAME:ParamVector",
    [ "NAME:Pair",
      "Name:=" , "DiaGap",
      "Value:=" , "100mm"
    ],
    [ "NAME:Pair",
      "Name:=" , "DiaYoke",
      "Value:=" , "20mm"
    ],
    [ "NAME:Pair",
      "Name:=" , "Length",
      "Value:=" , "100mm"
    ],
    [ "NAME:Pair",
      "Name:=" , "Skew",
      "Value:=" , "0deg"
```

```
        ] ,  
        [ "NAME:Pair",  
          "Name:=" , "Slots",  
          "Value:=" , "18"  
        ] ,  
        [ "NAME:Pair",  
          "Name:=" , "SlotType",  
          "Value:=" , "1"  
        ] ,  
        [ "NAME:Pair",  
          "Name:=" , "Hs0",  
          "Value:=" , "1mm"  
        ] ,  
        [ "NAME:Pair",  
          "Name:=" , "Hs1",  
          "Value:=" , "1mm"  
        ] ,  
        [ "NAME:Pair",  
          "Name:=" , "Hs2",  
          "Value:=" , "10mm"
```

```
],  
["NAME:Pair",  
"Name:=" , "Bs0",  
"Value:=" , "2.5mm"  
,  
["NAME:Pair",  
"Name:=" , "Bs1",  
"Value:=" , "8mm"  
,  
["NAME:Pair",  
"Name:=" , "Bs2",  
"Value:=" , "5mm"  
,  
["NAME:Pair",  
"Name:=" , "Rs",  
"Value:=" , "0mm"  
,  
["NAME:Pair",  
"Name:=" , "FilletType",  
"Value:=" , "0"  
,
```

```
[ "NAME:Pair",
  "Name:=" , "Layers",
  "Value:=" , "2"
] ,
[ "NAME:Pair",
  "Name:=" , "CoilPitch",
  "Value:=" , "4"
] ,
[ "NAME:Pair",
  "Name:=" , "EndExt",
  "Value:=" , "5mm"
] ,
[ "NAME:Pair",
  "Name:=" , "SpanExt",
  "Value:=" , "25mm"
] ,
[ "NAME:Pair",
  "Name:=" , "BendAngle",
  "Value:=" , "0deg"
] ,
```

```
[ "NAME:Pair",
  "Name:=" , "SegAngle",
  "Value:=" , "10deg"
] ,
[ "NAME:Pair",
  "Name:=" , "LenRegion",
  "Value:=" , "200mm"
] ,
[ "NAME:Pair",
  "Name:=" , "InfoCoil",
  "Value:=" , "0"
]
]

],
[ "NAME:Attributes",
  "Name:=" , "LapCoill",
  "Flags:=" , "",
  "Color:=" , "(143 175 143)",
  "Transparency:=" , 0,
  "PartCoordinateSystem:=" , "Global",
  "UDMId:=" , ""
```

```

    "MaterialValue:="      , "\"copper\",
    "SurfaceMaterialValue:=", "\\"\\",
    "SolveInside:="        , False,
    "ShellElement:="       , False,
    "ShellElementThickness:=", "0mm",
    "IsMaterialEditable:=" , True,
    "UseMaterialAppearance:=", False,
    "IsLightweight:="      , False
)

```

VB Syntax	CreateUserDefinedPart <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateUserDefinedPart Array("NAME:UserDefinedPrimitiveParameters", "DllName:", - "SegmentedHelix/RectHelix.dll", "Version:=", "1.0", "NoOfParameters:=", 8, "Library:", - "syslib", Array("NAME:ParamVector", Array("NAME:Pair", "Name:=", "RectHeight", "Value:=", _ "1mm"), Array("NAME:Pair", "Name:=", "RectWidth", "Value:=", "2mm"), Array("NAME:Pair", "Name:=", _ "StartHelixRadius", "Value:=", "10mm"), Array("NAME:Pair", "Name:=", "RadiusChange", "Value:=", _ </pre>

```

    "0mm"), Array("NAME:Pair", "Name:=", "Pitch", "Value:=", "3mm"), Array("NAME:Pair",
    "Name:=", _
    "Turns", "Value:=", "2"), Array("NAME:Pair", "Name:=", "SegmentsPerTurn", "Value:=", _
    "36"), Array("NAME:Pair", "Name:=", "RightHanded", "Value:=", "1"))), Array("NAME:At-
    tributes", "Name:=", _
    "RectHelix1", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0,
    "PartCoordinateSystem:=", _
    "Global", "UDMID:=", "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "Sur-
    faceMaterialValue:=", _
    "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _
    false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa-
    terialAppearance:=", _
    false, "IsLightweight:=", false)

```

## Edit3DComponent

Edits a specified 3D component.

<b>UI Access</b>	N/A										
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;compName&gt;</td> <td>String</td> <td>Component name.</td> </tr> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td>Structured array.             Array ("NAME&gt;EditComponentParametersData",           "NewComponentName:=", &lt;string&gt;,           "GeometryParameters:=", &lt;string&gt;,           "MaterialParameters:=", &lt;string&gt;,         </td> </tr> </tbody> </table>	Name	Type	Description	<compName>	String	Component name.	<Parameters>	Array	Structured array.  Array ("NAME>EditComponentParametersData",           "NewComponentName:=", <string>,           "GeometryParameters:=", <string>,           "MaterialParameters:=", <string>,	
Name	Type	Description									
<compName>	String	Component name.									
<Parameters>	Array	Structured array.  Array ("NAME>EditComponentParametersData",           "NewComponentName:=", <string>,           "GeometryParameters:=", <string>,           "MaterialParameters:=", <string>,									

			<pre>"DesignParameters:=", &lt;string&gt;, &lt;ComponentMeshing&gt;, &lt;Excitations&gt;)</pre>
	<i>&lt;ComponentMeshing&gt;</i>	Array	<p>Structured array.</p> <pre>Array ("NAME:Component Meshing",       "MeshAssembly:=", &lt;boolean&gt;)</pre>
	<i>&lt;Excitations&gt;</i>	Array	<p>Structured array containing array of suppressed excitations.</p> <pre>Array ("NAME:Excitations",       "Suppressed:=", &lt;array&gt;)</pre>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	Edit3DComponent (<compName>, <Parameters>)
<b>Python Example</b>	<pre>oEditor.Edit3DComponent(     "Connector1",     [         "NAME:EditComponentParametersData",         "NewComponentName:=", "Connector2",         "GeometryParameters:=", "",         "MaterialParameters:=", "",         "DesignParameters:=", "",         ["NAME:Component Meshing",         [</pre>

```

    "MeshAssembly:=", False],
    ["NAME:Excitations",
     "Suppressed:=", []
   ]
)

```

<b>VB Syntax</b>	Edit3DComponent <compName>, <Parameters>
<b>VB Example</b>	<pre> oEditor.Edit3DComponent  "Connector1", Array("NAME:EditComponentParametersData",       "NewComponentName:=", "Connector2",       "GeometryParameters:=", "",       "MaterialParameters:=", "",       "DesignParameters:=", "",       Array("NAME:Component Meshing",             "MeshAssembly:=", false),       Array("NAME:Excitations",             "Suppressed:=", Array())) ) </pre>

## [Beta] EditNativeComponentDefinition [Maxwell]

Edit a native component definition.

<b>UI Access</b>	In the <b>Project Manager</b> , expand <b>3D Components</b> , right-click on the component, and select <b>Edit Definition</b> .						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;LinkParameters&gt;</td><td>Array</td><td>Data of the 3D component.</td></tr></table>	Name	Type	Description	<LinkParameters>	Array	Data of the 3D component.
Name	Type	Description					
<LinkParameters>	Array	Data of the 3D component.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	EditNativeComponentDefinition (<LinkParameters>)
<b>Python Example</b>	<pre>oEditor.EditNativeComponentDefinition(     [         "NAME:EditNativeComponentDefinitionData",         "DefinitionName:=" , "LC1",         [             "NAME:GeometryDefinitionParameters",             [                 "NAME:VariableOrders"             ]         ],         [             "NAME:DesignDefinitionParameters",             [ </pre>

```
        "NAME:VariableOrders"
    ]
],
[
    "NAME:MaterialDefinitionParameters",
    [
        "NAME:VariableOrders"
    ]
],
"NextUniqueID:="      , 0,
"MoveBackwards:="     , False,
"DatasetType:="       , "ComponentDatasetType",
[
    "NAME:DatasetDefinitions"
],
[
    "NAME:NativeComponentDefinitionProvider",
    "Type:="              , "Layout Component",
    "Unit:="              , "mm",
    "Version:="           , 1.1,
    "EDBDefinition:="     , "EdgeCircuitPort_Inductor1",
```

```
[  
    "NAME:VariableMap"  
,  
    "ReferenceCS:=" , "U0",  
    "CSToImport:=" , ["Global","U0"]  
,  
    "ComponentName:=" , "LC1",  
    "Company:=" , "",  
    "Company URL:=" , "",  
    "Model Number:=" , "",  
    "Help URL:=" , "",  
    "Version:=" , "1.0",  
    "Notes:=" , "",  
    "IconType:=" , "Layout Component"  
]  
)
```

## EditPolyline

Modifies a specified polyline. See: [CreatePolyline](#).

UI Access	N/A
-----------	-----

	Name	Type	Description
<b>Parameters</b>	<code>&lt;SelectionsArray&gt;</code>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
	<code>&lt;Parameters&gt;</code>	Array	Structured array.  <code>Array ("NAME:PolylineParameters",            "IsPolylineCovered:=", &lt;bool&gt;,            "IsPolylineClosed:=", &lt;bool&gt;,            &lt;PolylinePointsArray&gt;,            &lt;PolylineSegmentsArray&gt;)</code>
	<code>&lt;PolylinePointsArray&gt;</code>	Array	<code>Array ("NAME:PolylinePoints", &lt;OnePointArray&gt;,            &lt;OnePointArray&gt;, ...)</code>
	<code>&lt;OnePointArray&gt;</code>	Array	<code>Array ("NAME:PLPoint",            "X:=", &lt;value&gt;,            "Y:=", &lt;value&gt;,            "Z:=", &lt;value&gt;))</code>
	<code>&lt;PolylineSegmentsArray&gt;</code>	Array	<code>&lt;PolylineSegmentsArray&gt;             Array ("NAME:PolylineSegments",            &lt;OneSegmentArray&gt;, &lt;OneSegmentArray&gt;, ...)</code>
	<code>&lt;OneSegmentArray&gt;</code>	Array	<code>Array ("NAME:PLSegment",            "SegmentType:=", &lt;"Line", "Arc", "Spline", or            "AngularArc"&gt;,            "StartIndex:=", &lt;value&gt;,            "NoOfPoints:=", &lt;value&gt;)</code>
<b>Return Value</b>	None.		

Python Syntax	EditPolyline(<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.EditPolyline(     ["NAME:Selections",         "Selections:=", "Polyline1"]     ["NAME:PolylineParameters",         "IsPolylineCovered:=" , True,         "IsPolylineClosed:=" , False,     ["NAME:PolylinePoints",         ["NAME:PLPoint",             "X:=" , "2000mm",             "Y:=" , "-2000mm",             "Z:=" , "0mm"         ],         ["NAME:PLPoint",             "X:=" , "-9000mm",             "Y:=" , "2000mm",             "Z:=" , "0mm"         ],         ["NAME:PLPoint",     ]]</pre>

```
"X:=" , "10000mm",
"Y:=" , "-140000mm",
"Z:=" , "0mm"
]
],
[ "NAME:PolylineSegments",
[ "NAME:PLSegment",
"SegmentType:=" , "Line",
"startIndex:=" , 0,
"noOfPoints:=" , 2
],
[ "NAME:PLSegment",
"SegmentType:=" , "Line",
"startIndex:=" , 1,
"noOfPoints:=" , 2
]
],
[ "NAME:PolylineXSection",
"XSectionType:=" , "None",
"XSectionOrient:=" , "Auto",
"XSectionWidth:=" , "0mm",
```

```

    "XSectionTopWidth:=", "0mm",
    "XSectionHeight:=", "0mm",
    "XSectionNumSegments:=", "0",
    "XSectionBendType:=", "Corner"
]
)

```

<b>VB Syntax</b> <pre>EditPolyline &lt;SelectionsArray&gt;, &lt;Parameters&gt;</pre>	
<b>VB Example</b> <pre> oEditor.EditPolyline Array("NAME:Selections", "Selections:=", "Polyline1") Array ("NAME:PolylineParameters", "IsPolylineCovered:=", true, "IsPolylineClosed:=", _ false, Array("NAME:PolylinePoints", Array("NAME:PLPoint", "X:=", "40000mm", "Y:=", _ "50000mm", "Z:=", "0mm"), Array("NAME:PLPoint", "X:=", "-150000mm", "Y:=", "-50000mm", "Z:=", _ "0mm")), Array("NAME:PolylineSegments", Array("NAME:PLSegment", "SegmentType:=", "Line", "StartIndex:=", _ 0, "NoOfPoints:=", 2)), Array("NAME:PolylineXSection", "XSectionType:=", "None", "XSec- tionOrient:=", _ "Auto", "XSectionWidth:=", "0mm", "XSectionTopWidth:=", "0mm", "XSectionHeight:=", _ "0mm", "XSectionNumSegments:=", "0", "XSectionBendType:=", "Corner")), Array("NAME:At- tributes", "Name:=", _ "Polyline2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordi- nates:=", "0mm", "XSectionWidth:=", "0mm", "XSectionTopWidth:=", "0mm", "XSectionHeight:=", _ "0mm", "XSectionNumSegments:=", "0", "XSectionBendType:=", "Corner")) </pre>	

```

    ateSystem:=", _  

    "Global", "UDMID:="", "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "Sur-  

    faceMaterialValue:=", _  

    "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _  

    false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa-  

    terialAppearance:=", _  

    false, "IsLightweight:=", false)

```

## Get3DComponentParameters

Returns parameters for a specified 3D component.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;compName&gt;</td> <td>String</td> <td>3D component name.</td> </tr> </tbody> </table>	Name	Type	Description	<compName>	String	3D component name.
Name	Type	Description					
<compName>	String	3D component name.					
<b>Return Value</b>	Array containing component parameters.						

<b>Python Syntax</b>	Get3DComponentParameters(<compName>)
<b>Python Example</b>	<code>oEditor.Get3DComponentParameters ('Connector')</code>

<b>VB Syntax</b>	Get3DComponentParameters <compName>
<b>VB Example</b>	<code>oEditor.Get3DComponentParameters "Connector"</code>

## Get3DComponentDefinitionNames

Gets names of 3D component definitions.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array of strings containing component definition names.

<b>Python Syntax</b>	Get3DComponentDefinitionNames()
<b>Python Example</b>	<code>oEditor.Get3DComponentDefinitionNames ()</code>

<b>VB Syntax</b>	Get3DComponentDefinitionNames()
<b>VB Example</b>	<pre>Dim defNames defNames = oEditor.Get3DComponentDefinitionNames ()</pre>

## Get3DComponentInstanceNames

Returns instance names of 3D component definitions.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><i>&lt;DefinitionName&gt;</i></td><td>String</td><td>Definition name.</td></tr></tbody></table>			Name	Type	Description	<i>&lt;DefinitionName&gt;</i>	String	Definition name.
Name	Type	Description							
<i>&lt;DefinitionName&gt;</i>	String	Definition name.							
<b>Return Value</b>	Array containing instance names.								

<b>Python Syntax</b>	Get3DComponentInstanceNames(<DefinitionName>)
----------------------	---

<b>Python Example</b>	<code>oEditor.Get3DComponentInstanceNames ("Connector")</code>
-----------------------	--

<b>VB Syntax</b>	<code>Get3DComponentInstanceNames &lt;DefinitionName&gt;</code>
------------------	---

<b>VB Example</b>	<code>oEditor.Get3DComponentInstanceNames "Connector"</code>
-------------------	--

## Get3DComponentMaterialNames

Returns material names for a specified \*.a3dcomp format 3D component.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td><code>&lt;InstanceName&gt;</code></td> <td>String</td> <td>Component instance name.</td> </tr> </table>	Name	Type	Description	<code>&lt;InstanceName&gt;</code>	String	Component instance name.
Name	Type	Description					
<code>&lt;InstanceName&gt;</code>	String	Component instance name.					
<b>Return Value</b>	Array containing material names.						

<b>Python Syntax</b>	<code>Get3DComponentMaterialNames(&lt;InstanceName&gt;)</code>
----------------------	--

<b>Python Example</b>	<code>oEditor.Get3DComponentMaterialNames ("Connector1.a3dcomp")</code>
-----------------------	---

<b>VB Syntax</b>	<code>Get3DComponentMaterialNames &lt;InstanceName&gt;</code>
------------------	---

<b>VB Example</b>	<code>oEditor.Get3DComponentMaterialNames "Connector1.a3dcomp"</code>
-------------------	---

## Get3DComponentMaterialProperties

Returns material properties for a specified 3D component.

<b>UI Access</b>	N/A
------------------	-----

Parameters	Name	Type	Description
	<MaterialName>	String	Material name.
Return Value	Array containing material properties.		

Python Syntax	Get3DComponentMaterialProperties(<MaterialName>)
Python Example	<code>oEditor.Get3DComponentMaterialProperties ('Connector1:Material01')</code>

VB Syntax	Get3DComponentMaterialProperties <MaterialName>
VB Example	<code>oEditor.Get3DComponentMaterialProperties "Connector1:Material01"</code>

## Insert3DComponent

Inserts a 3D component.

UI Access	Draw > 3D Component Library > Browse > [Component].		
Parameters	Name <ComponentData>	Type Array	Description Structured array.  Array ("NAME:InsertComponentData", "Parameters:=", <string>, "TargetCS:=", <string>, "ComponentFile:=", <string filepath>)

<b>Return Value</b>	None.
---------------------	-------

<b>Python Syntax</b>	Insert3DComponent(<ComponentData>)
<b>Python Example</b>	<pre> oEditor.Insert3DComponent(     [ "NAME:InsertComponentData",         "Parameters:=", "",         "TargetCS:=", "Global",         "ComponentFile:=", "C:\tmp\Connector.a3dcomp" ] ) </pre>

<b>VB Syntax</b>	Insert3DComponent <ComponentData>
<b>VB Example</b>	<pre> oEditor.Insert3DComponent Array("NAME:InsertComponentData",     "Parameters:=", "",     "TargetCS:=", "Global",     "ComponentFile:=", "C:\tmp\Connector.a3dcomp") </pre>

## [Beta] InsertNativeComponent [Layout Component to Maxwell with CS]

**[Beta]** Inserts a Layout Component to Maxwell 3D with custom coordinate system definitions.

<b>UI Access</b>	<b>Create Layout Component ...</b>		
<b>Parameters</b>	Name <LinkParameters>	Type Array	Description Structured array containing data of inserted layout component.

<b>Return Value</b>	None.
---------------------	-------

<b>Python Syntax</b>	InsertNativeComponent (<LinkedParameters>)
<b>Python Example</b>	<pre>oEditor.InsertNativeComponent (     [         "NAME:InsertNativeComponentData",         "TargetCS:="           , "Global",         "SubmodelDefinitionName:=", "LC1",         [             "NAME:ComponentPriorityLists"         ],         "NextUniqueID:="       , 0,         "MoveBackwards:="      , False,         "DatasetType:="        , "ComponentDatasetType",         [             "NAME:DatasetDefinitions"         ],         [             "NAME:BasicComponentInfo",</pre>

```
"ComponentName:=" , "LC1",
"Company:=" , "", 
"Company URL:=" , "", 
"Model Number:=" , "", 
"Help URL:=" , "", 
"Version:=" , "1.0",
"Notes:=" , "", 
"IconType:=" , "Layout Component"
],
[
  "NAME:GeometryDefinitionParameters",
  [
    "NAME:VariableOrders"
  ]
],
[
  "NAME:DesignDefinitionParameters",
  [
    "NAME:VariableOrders"
  ]
],
```

```
[  
    "NAME:MaterialDefinitionParameters",  
    [  
        "NAME:VariableOrders"  
    ]  
,  
    "DefReferenceCSID:="      , 1,  
    "MapInstanceParameters:=", "DesignVariable",  
    "UniqueDefinitionIdentifier:=", "27e7d8f5-da68-409b-8c4d-bf2bc6d95f69",  
    "OriginFilePath:="        , "",  
    "IsLocal:="                , False,  
    "ChecksumString:="        , "",  
    "ChecksumHistory:="       , [],  
    "VersionHistory:="        , [],  
    [  
        "NAME:NativeComponentDefinitionProvider",  
        "Type:="                  , "Layout Component",  
        "Unit:="                  , "mm",  
        "EDBDefinition:="         , "EdgeCircuitPort_Inductor1",  
        [  
    ]
```

```

        "NAME:VariableMap"
    ],
    "ReferenceCS:=" , "Global",
    "CSToImport:=" , ["Global", "U0"]
],
[
    "NAME:InstanceParameters",
    "GeometryParameters:=" , "",
    "MaterialParameters:=" , "",
    "DesignParameters:=" , ""
]
)

```

## InsertPolylineSegment

Insets a polyline segment before or after a specified existing segment.

UI Access	Draw > Line Segment > [Selection].		
Parameters	Name	Type	Description
	<Parameters>	Array	<p>Structured array.</p> <pre> Array("NAME:Insert Polyline Segment",       "Selections:=" , &lt;string&gt;,       "Segment Indices:=" , &lt;array containing </pre>

		<pre>integers&gt;,     "At Start:=" , &lt;boolean&gt;,     "SegmentType:=" , &lt;string "Line", "Arc", " Spline", or "AngularArc"&gt;, &lt;PolylinePointsArray&gt;</pre>
	<code>&lt;PolylinePointsArray&gt;</code>	Array Structured array. See: <a href="#">CreatePolyline</a> .
<b>Return Value</b>	None.	

<b>Python Syntax</b>	<code>InsertPolylineSegment(&lt;Parameters&gt;)</code>
<b>Python Example</b>	<pre>oEditor.InsertPolylineSegment(     ["NAME:Insert Polyline Segment",      "Selections:=" , "Polyline1&gt;CreatePolyline:1",      "Segment Indices:=" , [0],      "At Start:=" , True,      "SegmentType:=" , "Line",      ["NAME:PolylinePoints",       ["NAME:PLPoint",        "X:=" , "1.1mm",        "Y:=" , "0.8mm",        "Z:=" , "0mm"]]]</pre>

```
    ] ,  
    [ "NAME:PLPoint",  
      "X:=" , "0.6mm",  
      "Y:=" , "-0.8mm",  
      "Z:=" , "0mm"  
    ]  
  ] )
```

VB Syntax	InsertPolylineSegment <Parameters>
VB Example	<pre data-bbox="380 799 1727 902"> oEditor.InsertPolylineSegment Array("NAME:Insert Polyline Segment", "Selections:=", "Polyline1&gt;CreatePolyline:1", "Segment Indices:=", Array(1), "At Start:=", false, "SegmentType:=", "Spline", Array("NAME:PolylinePoints", Array("NAME:PLPoint", "X:=", "-1.4mm", "Y:=", "0.4mm", "Z:=", "0mm"), Array("NAME:PLPoint", "X:=", "0.5mm", "Y:=", "-1.1mm", "Z:=", "0mm"), Array("NAME:PLPoint", "X:=", "0.7mm", "Y:=", "-2.1mm", "Z:=", "0mm"), Array("NAME:PLPoint", "X:=", "0.4mm", "Y:=", "-1.1mm", "Z:=", "0mm")) ) </pre>

# SweepAlongPath

Sweeps the specified 1D or 2D parts along a path. The last 1D object specified is the path for the sweep.

UI Access	Draw > Sweep > Along Path.		
	Name	Type	Description
Parameters	<p>&lt;SelectionsArray&gt;</p> <p>&lt;PathSweepParametersArray&gt;</p>	<p>Array</p> <p>Array</p>	<p>Structured array. See: <a href="#">SelectionsArray</a>.</p> <p>Array ("NAME:PathSweepParameters",            "DraftAngle:=", &lt;value&gt;,            "DraftType:=", &lt;string&gt;,            "CheckFaceFaceIntersection:=", &lt;bool&gt;,            "TwistAngle:=", &lt;value&gt;)            Possible values for DraftType are "Extended", "Round", and "Natural".</p>
Return Value	None.		

Python Syntax	SweepAlongPath(<SelectionsArray>, <PathSweepParametersArray>)
Python Example	<pre> oEditor.SweepAlongPath(     [         "NAME:Selections",         "Selections:=" , "Rectangle1,Polyline1",         "NewPartsModelFlag:=" , "Model"     ],     [ </pre>

```

    "NAME:PathSweepParameters",
    "DraftAngle:=", "0deg",
    "DraftType:=", "Round",
    "CheckFaceFaceIntersection:=", False,
    "TwistAngle:=", "0deg"
)
)

```

<b>VB Syntax</b>	SweepAlongPath <SelectionsArray>, <PathSweepParametersArray>
<b>VB Example</b>	<pre> oEditor.SweepAlongPath  Array("NAME:Selections", "Selections:=",       "Polygon1,Polyline1"),_  Array("NAME:PathSweepParameters", _       "DraftAngle:=", "0deg",_       "DraftType:=", "Round",_       "CheckFaceFaceIntersection:=", False,_       "TwistAngle:=", "30deg") </pre>

## SweepAlongVector

Sweeps the specified 1D or 2D parts along a vector.

**UI Access**

**Draw > Sweep > Along Vector.**

	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
<b>Parameters</b>	<VecSweepParametersArray>	Array	<p>Array ("NAME:VectorSweepParameters",</p> <p>"DraftAngle:=", &lt;value&gt;,</p> <p>"DraftType:=", &lt;string&gt;,</p> <p>"CheckFaceFaceIntersection:=", &lt;bool&gt;,</p> <p>"SweepVectorX:=", &lt;value&gt;</p> <p>"SweepVectorY:=", &lt;value&gt;</p> <p>"SweepVectorZ:=", &lt;value&gt;)</p> <p>Possible values for DraftType are "Extended", "Round", and "Natural".</p>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	SweepAlongVector(<SelectionsArray>, <VecSweepParametersArray>)
<b>Python Example</b>	<pre> oEditor.SweepAlongVector( [     "NAME:Selections",     "Selections:=" , "Rectangle1",     "NewPartsModelFlag:=" , "Model" ], </pre>

```
[  

  "NAME:VectorSweepParameters",  

  "DraftAngle:=", "0deg",  

  "DraftType:=", "Round",  

  "CheckFaceFaceIntersection:=", False,  

  "SweepVectorX:=", "0mm"  

  "SweepVectorY:=", "0mm"  

  "SweepVectorZ:=", "12mm"  

])
```

VB Syntax	SweepAlongVector <SelectionsArray>, <VecSweepParametersArray>
VB Example	<pre> oEditor.SweepAlongPath  Array("NAME:Selections",_       "Selections:=", "Rectangle1",_       "NewPartsModelFlag:=", "Model")  Array("NAME:VectorSweepParameters", _        "DraftAngle:=", "0deg", _        "DraftType:=", "Round", _        "CheckFaceFaceIntersection:=", False, _        "SweepVectorX:=", "0mm", _        "SweepVectorY:=", "0mm", _</pre>

	"SweepVectorZ:=" , "12mm")
--	----------------------------

## SweepAroundAxis

Sweeps the specified 1D or 2D parts around an axis.

UI Access	Draw > Sweep > Around Axis.		
Parameters	Name <i>&lt;SelectionsArray&gt;</i> <i>&lt;AxisSweepParametersArray&gt;</i>	Type Array Array	Description Structured array. See: <a href="#">SelectionsArray</a> . Array ("NAME:AxisSweepParameters", "DraftAngle:=", <value>, "DraftType:=", <string>, "CheckFaceFaceIntersection:=", <bool>, "SweepAxis:=", <value> "SweepAngle:=", <value> "NumOfSegments:=", <value>)  Possible values for DraftType are "Extended", "Round", and "Natural".  Possible values for SweepAxis are "X", "Y", and "Z".
Return Value	None.		

Python Syntax	SweepAroundAxis(<SelectionsArray>, <AxisSweepParametersArray>)
Python Example	oEditor.SweepAroundAxis (

```
[

    "NAME:Selections",
    "Selections:=", "Rectangle1",
    "NewPartsModelFlag:=", "Model"

],


[

    "NAME:AxisSweepParameters",
    "DraftAngle:=", "0deg",
    "DraftType:=", "Round",
    "CheckFaceFaceIntersection:=", False,
    "SweepAxis:=", "X"
    "SweepAngle:=", "360deg"
    "NumOfSegments:=", "12"
])
)
```

<b>VB Syntax</b>	SweepAroundAxis <SelectionsArray>, <AxisSweepParametersArray>
<b>VB Example</b>	<pre> oEditor.SweepAroundAxis      Array("NAME:Selections",_         "Selections:=", "Rectangle1",_         "NewPartsModelFlag:=", "Model")      Array("NAME:AxisSweepParameters", _</pre>

```

    "DraftAngle:=", "0deg",
    "DraftType:=", "Round",
    "CheckFaceFaceIntersection:=", False,
    "SweepAxis:=", "X",
    "SweepAngle:=", "360deg",
    "NumOfSegments:=", "12")

```

## SweepFacesAlongNormal

Sweep the specified face(s) along normal.

UI Access	Modeler > Surface > Sweep Faces Along Normal		
	Name <i>&lt;SelectionsArray&gt;</i>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
Parameters	<i>&lt;parameters&gt;</i>	Array	Structured array.  Array ("NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", <i>&lt;faceIDarray&gt;</i> , "LengthOfSweep:=", " <i>&lt;value&gt;&lt;units&gt;</i> ")
Return Value	None		

Python Syntax	SweepFacesAlongNormal( <i>&lt;SelectionsArray&gt; &lt;parameters&gt;</i> )
---------------	--

<b>Python Example</b>	<pre><code>oEditor.SweepFacesAlongNormal (     ["NAME:Selections",         "Selections:=", "Rectangle1",         "NewPartsModelFlag:=", "Model"],     ["NAME:Parameters",         "NAME:SweepFaceAlongNormalToParameters",         "FacesToDetach:=", [183],         "LengthOfSweep:=", "0.1mm"])</code></pre>
-----------------------	--

VB Syntax	SweepFacesAlongNormal < <i>SelectionsArray</i> > < <i>parameters</i> >
<b>VB Example</b>	<pre><code>oEditor.SweepFacesAlongNormal     Array("NAME:Selections",         "Selections:=", "Rectangle1",         "NewPartsModelFlag:=", "Model"),     Array("NAME:Parameters",         "NAME:SweepFaceAlongNormalToParameters",         "FacesToDetach:=", Array(57),         "LengthOfSweep:=", "0.5mm")     )</code></pre>

## SweepFacesAlongNormalWithAttributes

Sweep a face along normal, and specify attributes of the new object.

UI Access	Modeler > Surface > Sweep Faces Along Normal		
<b>Parameters</b>	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
	<parameters>	Array	Array ("NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", <faceIDarray>, "LengthOfSweep:=", "<value><units>")
	<AttributesArray>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None		

Python Syntax	SweepFacesAlongNormalWithAttributes(<SelectionsArray>, <parameters>, <AttributesArray>)
<b>Python Example</b>	<pre>oEditor.SweepFacesAlongNormalWithAttributes (     ["NAME:Selections",         "Selections:=", "Rectangle1",         "NewPartsModelFlag:=", "Model"],     ["NAME:Parameters",         "NAME:SweepFaceAlongNormalToParameters",         "FacesToDetach:=", [183],</pre>

```

    "LengthOfSweep:=", "0.1mm"],
["NAME:Attributes",
 "Name:=", "Box3",
 "Flags:=", "",
 "Color:=", "(143 175 143)",
 "Transparency:=", 0,
 "PartCoordinateSystem:=", "Global",
 "UDMID:=", "",
 "MaterialValue:=", "\"copper\"",
 "SurfaceMaterialValue:=", "\",
 "SolveInside:=", False,
 "ShellElement:=", False,
 "ShellElementThickness:=", "0mm",
 "IsMaterialEditable:=", True,
 "UseMaterialAppearance:=", False,
 "IsLightweight:=", False])

```

<b>VB Syntax</b>	SweepFacesAlongNormalWithAttributes <SelectionsArray>, <parameters>, <AttributesArray>
<b>VB Example</b>	<pre> oEditor.SweepFacesAlongNormalWithAttributes Array("NAME:Selections", </pre>

```
"Selections:=", "Rectangle1",
"NewPartsModelFlag:=", "Model"),
Array("NAME:Parameters",
      "NAME:SweepFaceAlongNormalToParameters",
      "FacesToDetach:=", Array(57),
      "LengthOfSweep:=", "0.5mm"),
Array("NAME:Attributes",
      "Name:=", "Box3",
      "Flags:=", "",
      "Color:=", "(143 175 143)",
      "Transparency:=", 0,
      "PartCoordinateSystem:=", "Global",
      "UDMID:=", "",
      "MaterialValue:=", "\copper\",
      "SurfaceMaterialValue:=", "\\""",
      "SolveInside:=", false,
      "ShellElement:=", false,
      "ShellElementThickness:=", "0mm",
      "IsMaterialEditable:=", true,
      "UseMaterialAppearance:=", false,
```

	<pre>"IsLightweight:=" , false) )</pre>
--	---

## UpdateComponentDefinition

Updates a 3D component's definition.

UI Access	Draw > 3D Component Library > Definitions.		
Parameters	Name	Type	Description
	<data>	Array	<p>Structured array.</p> <pre>Array ("NAME:UpdateDefinitionData",       "DefinitionNames:=", &lt;string&gt;,       "Passwords:=", &lt;array of strings&gt;)</pre>
Return Value	None.		

Python Syntax	UpdateComponentDefinition(<data>)
Python Example	<pre>oEditor.UpdateComponentDefinition (     [ 'NAME:UpdateDefinitionData',       'DefinitionNames:=' , 'Connector, Magic_Tee',       'Passwords:=' , [ '' , '' ]     ] )</pre>

VB Syntax	UpdateComponentDefinition <data>
-----------	----------------------------------

**VB Example**

```
oEditor.UpdateComponentDefinition  
Array("NAME:UpdateDefinitionData",  
      "DefinitionNames:=", " Connector, Magic_Tee",  
      "Passwords:=", Array("", ""))
```

## Edit Menu Commands

[Copy](#)[DeletePolylinePoint](#)[DuplicateAlongLine](#)[DuplicateAroundAxis](#)[DuplicateMirror](#)[Mirror](#)[Move](#)[OffsetFaces](#)[Paste](#)[Rotate](#)[Scale](#)**[Copy](#)**

Copies specified part(s) to the clipboard.

UI Access	N/A
-----------	-----

<b>Parameters</b>	Name <i>&lt;SelectionsArray&gt;</i>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	Copy( <i>&lt;SelectionsArray&gt;</i> )
<b>Python Example</b>	<pre>oEditor.Copy([     "NAME:Selections",     "Selections:=", "Box1"])</pre>

<b>VB Syntax</b>	Copy < <i>SelectionsArray</i> >
<b>VB Example</b>	<pre>oEditor.Copy Array(     "NAME:Selections",     "Selections:=", "Box1")</pre>

## DeletePolylinePoint

Deletes either a start point or an end point from an existing polyline segment.

<b>UI Access</b>	<b>Edit &gt; Delete [Start/End] Point</b>		
<b>Parameters</b>	Name <i>&lt;DeletePointArray&gt;</i>	Type Array	Description Structured array. <pre>Array("NAME:Delete Point",       "Selections:=", &lt;string "&lt;PolylineName&gt;:&lt;PolylineAction&gt;:&lt;int&gt;&gt;,</pre>

			"Segment Index:=", <integer>, "At Start:=", <bool True for start point; False for end point>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	DeletePolylinePoint (<DeletePointArray>)
<b>Python Example</b>	<pre>oEditor.DeletePolylinePoint(["NAME:Delete Point",     "Selections:=", "Polyline1&gt;CreatePolyline:1",     "Segment Index:=", 1,     "At Start:=", True])</pre>

<b>VB Syntax</b>	DeletePolylinePoint <DeletePointArray>
<b>VB Example</b>	<pre>oEditor.DeletePolylinePoint Array("NAME:Delete Point", "Selections:=", "Polyline1&gt;CreatePolyline:1", "Segment Index:=", 1, "At Start:=", True)</pre>

## DuplicateAlongLine

Duplicates specified parts along a line.

<b>UI Access</b>	Edit > Duplicate > Along Line.		
<b>Parameters</b>	Name	Type	Description

	<code>&lt;SelectionsArray&gt;</code>	Array	Structured array.  Array ("NAME:Selections", "Selections:=" , <string>, "NewPartsModelFlag:=" , <string>)
	<code>&lt;ParametersArray&gt;</code>	Array	Structured array.  Array ("NAME:DuplicateToAlongLineParameters", "CreateNewObjects:=" , <boolean>, "XComponent:=" , <string>, "YComponent:=" , <string>, "ZComponent:=" , <string>, "NumClones:=" , <string containing number greater than 1>)
	<code>&lt;OptionsArray&gt;</code>	Array	Structured array.  Array ("NAME:Options", "DuplicateAssignments:=" , <boolean>)
	<code>&lt;CreateGroup&gt;</code>	Array	Optional. Structured array.  Array ("CreateGroupsForNewObjects:=" , <boolean>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	DuplicateAlongLine (<SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>)
<b>Python Example</b>	<code>oEditor.DuplicateAlongLine (</code> <code>  ["NAME:Selections",</code>

```
"Selections:=" , "Box1",
"NewPartsModelFlag:=" , "Model"
],
[ "NAME:DuplicateToAlongLineParameters",
  "CreateNewObjects:=" , False,
  "XComponent:=" , "1mm",
  "YComponent:=" , "-0.7mm",
  "ZComponent:=" , "0mm",
  "NumClones:=" , "2"
],
[ "NAME:Options",
  "DuplicateAssignments:=", False
],
[ "CreateGroupsForNewObjects:=", False
])
```

<b>VB Syntax</b>	DuplicateAlongLine < <i>SelectionsArray</i> >, < <i>ParametersArray</i> >, < <i>OptionsArray</i> >, < <i>CreateGroup</i> >
<b>VB Example</b>	<pre>oEditor.DuplicateAlongLine   Array(  "NAME:Selections",</pre>

```

    "Selections:=", "Box1",
    "NewPartsModelFlag:=", "Model")

Array( "NAME:DuplicateToAlongLineParameters",
       "CreateNewObjects:=", false,
       "XComponent:=", "1mm",
       "YComponent:=", "-0.7mm",
       "ZComponent:=", "0mm",
       "NumClones:=", "2")

Array( "NAME:Options",
       "DuplicateAssignments:=", false)

Array( "CreateGroupsForNewObjects:=", false)

```

## DuplicateAroundAxis

Duplicates specified parts around an axis.

UI Access	Edit > Duplicate > Around Axis.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. Array ("NAME:Selections",       "Selections:=" , <string>,       "NewPartsModelFlag:=" , <string>)
	<ParametersArray>	Array	Structured array. Array ("NAME:DuplicateAroundAxisParameters",

			<pre>"CreateNewObjects:=" , &lt;boolean&gt;, "WhichAxis:=" , &lt;string&gt;, "AngleStr:=" , &lt;string&gt;, "NumClones:=" , &lt;string containing number greater than 1&gt;)</pre>
	<p><b>&lt;OptionsArray&gt;</b></p>	Array	<p>Structured array.</p> <pre>Array("NAME:Options", "DuplicateAssignments:=" , &lt;boolean&gt;)</pre>
	<p><b>&lt;CreateGroup&gt;</b></p>	Array	<p>Optional. Structured array.</p> <pre>Array("CreateGroupsForNewObjects:=" , &lt;boolean&gt;)</pre>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	DuplicateAroundAxis (<SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>)
<b>Python Example</b>	<pre>oEditor.DuplicateAroundAxis( ["NAME:Selections",  "Selections:=" , "Box1",  "NewPartsModelFlag:=" , "Model" ], ["NAME:DuplicateAroundAxisParameters",  "CreateNewObjects:=" , True,</pre>

```

    "WhichAxis:="           , "Z",
    "AngleStr:="           , "90deg",
    "NumClones:="          , "2"
],
[ "NAME:Options",
  "DuplicateAssignments:=", False
],
[ "CreateGroupsForNewObjects:=", False
])

```

<b>VB Syntax</b>	DuplicateAroundAxis <SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>
<b>VB Example</b>	<pre> oEditor.DuplicateAroundAxis  Array("NAME:Selections",       "Selections:=", "Box1",       "NewPartsModelFlag:=", "Model")  Array("NAME:DuplicateAroundAxisParameters",       "CreateNewObjects:=", true,       "WhichAxis:=", "Z",       "AngleStr:=", "90deg",       "NumClones:=", "2")  Array("NAME:Options", </pre>

	<pre>"DuplicateAssignments:=", false) Array("CreateGroupsForNewObjects:=", false)</pre>
--	---

## DuplicateMirror

Duplicates specified parts according to a mirror plane.

UI Access	Edit > Duplicate > Mirror.		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;SelectionsArray&gt;</code>	Array	<p>Structured array.</p> <pre>Array("NAME:Selections",       "Selections:=" , &lt;string&gt;,       "NewPartsModelFlag:=" , &lt;string&gt;)</pre>
	<code>&lt;ParametersArray&gt;</code>	Array	<p>Structured array.</p> <pre>Array("NAME:DuplicateToMirrorParameters",       "DuplicateMirrorBaseX:=" , &lt;string&gt;,       "DuplicateMirrorBaseY:=" , &lt;string&gt;,       "DuplicateMirrorNormalX:=" , &lt;string&gt;,       "DuplicateMirrorNormalY:=" , &lt;string&gt;,       "DuplicateMirrorNormalZ:=" , &lt;string&gt;)  For Maxwell 2D XY designs, Z parameters should be set to "0". For Maxwell 2D RZ designs, Y parameters should be set to "0".</pre>
	<code>&lt;OptionsArray&gt;</code>	Array	Structured array.

		Array ("NAME:Options", "DuplicateAssignments:=", <boolean>)
<CreateGroup>	Array	Optional. Structured array. Array ("CreateGroupsForNewObjects:=", <boolean>)
<b>Return Value</b>	None.	

<b>Python Syntax</b>	DuplicateMirror (<SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>)
<b>Python Example</b>	<pre> oEditor.DuplicateMirror(     [         "NAME:Selections",         "Selections:=",         "Box1",         "NewPartsModelFlag:=",         "Model"     ],     [         "NAME:DuplicateToMirrorParameters",         "DuplicateMirrorBaseX:=",         "-0.4mm",         "DuplicateMirrorBaseY:=",         "-1.2mm",         "DuplicateMirrorBaseZ:=",         "0mm",         "DuplicateMirrorNormalX:=",         "0.124034734589208mm",         "DuplicateMirrorNormalY:=",         "0.992277876713668mm",         "DuplicateMirrorNormalZ:=",         "0mm"     ],     [         "NAME:Options",         "DuplicateAssignments:=",         "True"     ] ) </pre>

```
    "DuplicateAssignments:=", False  
],  
["CreateGroupsForNewObjects:=", False  
])
```

<b>VB Syntax</b>	DuplicateMirror <SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>
<b>VB Example</b>	<pre>oEditor.DuplicateMirror Array("NAME:Selections",       "Selections:=", "Box1",       "NewPartsModelFlag:=", "Model) Array("NAME:DuplicateToMirrorParameters",       "DuplicateMirrorBaseX:=", "-0.4mm",       "DuplicateMirrorBaseY:=", "-1.2mm",       "DuplicateMirrorBaseZ:=", "0mm",       "DuplicateMirrorNormalX:=", "0.124034734589208mm",       "DuplicateMirrorNormalY:=", "0.992277876713668mm",       "DuplicateMirrorNormalZ:=", "0mm") Array("NAME:Options",       "DuplicateAssignments:=", false)</pre>

	Array ("CreateGroupsForNewObjects:=", false)
--	--

## Mirror

Mirrors specified part(s).

UI Access	Edit > Arrange > Mirror.			
Parameters	Name <i>&lt;SelectionsArray&gt;</i>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .	
	<i>&lt;MirrorParameters&gt;</i>	Array	Structured array. <pre>Array ("NAME:MirrorParameters",       "MirrorBaseX:=" , &lt;string&gt;,       "MirrorBaseY:=" , &lt;string&gt;,       "MirrorBaseZ:=" , &lt;string&gt;,       "MirrorNormalX:=" , &lt;string&gt;,       "MirrorNormalY:=" , &lt;string&gt;,       "MirrorNormalZ:=" , &lt;string&gt;)</pre>	
Return Value	None.			

Python Syntax	Mirror(<SelectionsArray>, <MirrorParameters>)
Python Example	<pre>oEditor.Mirror(   [ "NAME:Selections",     "Selections:="           , "Box1_1",     "NewPartsModelFlag:="    , "Model"</pre>

```
        ],
        [
            "NAME:MirrorParameters",
            "MirrorBaseX:=", "-0.2mm",
            "MirrorBaseY:=", "-1.2mm",
            "MirrorBaseZ:=", "0mm",
            "MirrorNormalX:=", "-0.316227766016838mm",
            "MirrorNormalY:=", "0.948683298050514mm",
            "MirrorNormalZ:=", "0mm"
        ]
    )
```

VB Syntax	Mirror <SelectionsArray> <MirrorParameters>
<b>VB Example</b>	<pre>oEditor.Mirror  Array("NAME:Selections",       "Selections:=", "Box1_1",       "NewPartsModelFlag:=", "Model")  Array("NAME:MirrorParameters",       "MirrorBaseX:=", "-0.2mm",       "MirrorBaseY:=", "-1.2mm",       "MirrorBaseZ:=", "0mm",       "MirrorNormalX:=", "-0.316227766016838mm",</pre>

	<pre>"MirrorNormalY:=", "0.948683298050514mm", "MirrorNormalZ:=", "0mm")</pre>

## Move

Moves specified part(s).

UI Access	Edit > Arrange > Move.											
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SelectionsArray&gt;</td> <td>Array</td> <td>Structured array. See: <a href="#">SelectionsArray</a>.</td> </tr> <tr> <td>&lt;TranslateParameters&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array([ "NAME:TranslateParameters",        "TranslateVectorX:=" , &lt;string&gt;,        "TranslateVectorY:=" , &lt;string&gt;,        "TranslateVectorZ:=" , &lt;string&gt;) For Maxwell 2D XY designs, "TranslateVectorZ:=" should be set to "0". For Maxwell 2D RZ designs, "TranslateVectorY:=" should be set to "0".</pre> </td> </tr> </tbody> </table>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .	<TranslateParameters>	Array	<p>Structured array.</p> <pre>Array([ "NAME:TranslateParameters",        "TranslateVectorX:=" , &lt;string&gt;,        "TranslateVectorY:=" , &lt;string&gt;,        "TranslateVectorZ:=" , &lt;string&gt;) For Maxwell 2D XY designs, "TranslateVectorZ:=" should be set to "0". For Maxwell 2D RZ designs, "TranslateVectorY:=" should be set to "0".</pre>		
Name	Type	Description										
<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .										
<TranslateParameters>	Array	<p>Structured array.</p> <pre>Array([ "NAME:TranslateParameters",        "TranslateVectorX:=" , &lt;string&gt;,        "TranslateVectorY:=" , &lt;string&gt;,        "TranslateVectorZ:=" , &lt;string&gt;) For Maxwell 2D XY designs, "TranslateVectorZ:=" should be set to "0". For Maxwell 2D RZ designs, "TranslateVectorY:=" should be set to "0".</pre>										
Return Value	None.											

Python Syntax	Move(<SelectionsArray>, <TranslateParameters>)
Python Example	<pre>oEditor.Move(     [ "NAME:Selections",       "Selections:=" , "Box1_1",</pre>

```
"NewPartsModelFlag:="      , "Model"
] ,
[ "NAME:TranslateParameters",
  "TranslateVectorX:="      , "-0.5mm",
  "TranslateVectorY:="      , "0.1mm",
  "TranslateVectorZ:="      , "0mm"
])
```

VB Syntax	Move <SelectionsArray> <TranslateParameters>
VB Example	<pre>oEditor.Move  Array("NAME:Selections",       "Selections:=", "Box1_1",       "NewPartsModelFlag:=", "Model")  Array("NAME:TranslateParameters",       "TranslateVectorX:=", "-0.5mm",       "TranslateVectorY:=", "0.1mm",       "TranslateVectorZ:=", "0mm")</pre>

## OffsetFaces

Offsets the faces of selected part(s).

<b>UI Access</b>	Edit > Arrange > Offset.									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SelectionsArray&gt;</td> <td>Array</td> <td>Structured array. See: <a href="#">SelectionsArray</a>.</td> </tr> <tr> <td>&lt;OffsetParameters&gt;</td> <td>Array</td> <td>Structured array.  Array ("NAME:OffsetParameters", "OffsetDistance:=", &lt;string&gt;)</td> </tr> </tbody> </table>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .	<OffsetParameters>	Array	Structured array.  Array ("NAME:OffsetParameters", "OffsetDistance:=", <string>)
Name	Type	Description								
<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .								
<OffsetParameters>	Array	Structured array.  Array ("NAME:OffsetParameters", "OffsetDistance:=", <string>)								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	OffsetFaces (<SelectionsArray>, <OffsetParameters>)
<b>Python Example</b>	<pre> oEditor.OffsetFaces (     ["NAME:Selections",         "Selections:=", "Box1_1",         "NewPartsModelFlag:=", "Model"     ],     ["NAME:OffsetParameters",         "OffsetDistance:=", "16mm"     ] ) </pre>

<b>VB Syntax</b>	OffsetFaces <SelectionsArray> <OffsetParameters>
<b>VB Example</b>	<pre> oEditor.OffsetFaces     Array("NAME:Selections", </pre>

	<pre>"Selections:=", "Box1_1", "NewPartsModelFlag:=", "Model") Array("NAME:OffsetParameters", "OffsetDistance:=", "16mm")</pre>
--	---

## Paste (Model Editor)

Pastes previously copied object(s). See: [Copy](#).

<b>UI Access</b>	<b>Edit &gt; Paste.</b>
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	Paste()
<b>Python Example</b>	<pre>oEditor.Copy( ["NAME:Selections",  "Selections:=", "Box1_2" ]) oEditor.Paste()</pre>

<b>VB Syntax</b>	Paste
<b>VB Example</b>	<pre>oEditor.Copy Array("NAME:Selections", "Selections:=", "Box1_2")</pre>

	<code>oEditor.Paste</code>
--	----------------------------

## Rotate

Rotates specified object(s).

UI Access	<b>Edit &gt; Arrange &gt; Rotate.</b>		
<b>Parameters</b>	Name <code>&lt;SelectionsArray&gt;</code>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
	<code>&lt;RotateParameters&gt;</code>	Array	<p>Structured array.</p> <pre>Array ("NAME:RotateParameters",       "RotateAxis:=" , &lt;string "X", "Y", or "Z"&gt;,       "RotateAngle:=" , &lt;string&gt;)</pre> <p>For Maxwell 2D XY Designs, "RotateAxis:=" should be set to "Z".</p> <p>For Maxwell 2D RZ Designs, "RotateAxis:=" should be set to "Y".</p>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>Rotate(&lt;SelectionsArray&gt;, &lt;RotateParameters&gt;)</code>
<b>Python Example</b>	<pre>oEditor.Rotate(     [ "NAME:Selections",         "Selections:=", "Box1_1",         "NewPartsModelFlag:=", "Model"     ],     [ "NAME:RotateParameters",         "RotateAxis:=" , "Z",         "RotateAngle:=" , "90"     ] )</pre>

```
"RotateAxis:=", "Z",
"RotateAngle:=", "90deg"
])
```

<b>VB Syntax</b>	Rotate <SelectionsArray> <RotateParameters>
<b>VB Example</b>	<pre>oEditor.Rotate     Array("NAME:Selections",         "Selections:=", "Box1_1",         "NewPartsModelFlag:=", "Model")     Array("NAME:RotateParameters",         "RotateAxis:=", "Z",         "RotateAngle:=", "90deg")</pre>

## Scale

Scales specified object(s).

<b>UI Access</b>	Edit > Scale.									
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;SelectionsArray&gt;</td><td>Array</td><td>Structured array. See: <a href="#">SelectionsArray</a>.</td></tr><tr><td>&lt;ScaleParameters&gt;</td><td>Array</td><td>Structured array. Array ("NAME:ScaleParameters",     "ScaleX:=" , &lt;string containing scale factor&gt;,</td></tr></tbody></table>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .	<ScaleParameters>	Array	Structured array. Array ("NAME:ScaleParameters",     "ScaleX:=" , <string containing scale factor>,
Name	Type	Description								
<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .								
<ScaleParameters>	Array	Structured array. Array ("NAME:ScaleParameters",     "ScaleX:=" , <string containing scale factor>,								

			<pre>"ScaleY:=" , &lt;string containing scale factor&gt;, "ScaleZ:=" , &lt;string containing scale factor&gt;)</pre> <p>For Maxwell 2D XY Designs, "ScaleZ:=" should be set to "0".</p> <p>For Maxwell 2D RZ Designs, "ScaleY:=" should be set to "0".</p>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	Scale (<SelectionsArray>, <ScaleParameters>)
<b>Python Example</b>	<pre>oEditor.Scale( ["NAME:Selections",  "Selections:=", "Box1",  "NewPartsModelFlag:=", "Model" ], ["NAME:ScaleParameters",  "ScaleX:=", "2",  "ScaleY:=", "2",  "ScaleZ:=", "2" ])</pre>

<b>VB Syntax</b>	Scale <SelectionsArray> <ScaleParameters>
<b>VB Example</b>	<pre>oEditor.Scale Array("NAME:Selections",</pre>

```
"Selections:=", "Box1",
"NewPartsModelFlag:=", "Model")

Array("NAME:ScaleParameters",
      "ScaleX:=", "2",
      "ScaleY:=", "2",
      "ScaleZ:=", "2")
```

## Modeler Menu Commands

[AssignMaterial](#)

[Chamfer](#)

[Connect](#)

[CoverLines](#)

[CoverSurfaces](#)

[CreateEntityList](#)

[CreateFaceCS](#)

[CreateGroup](#)

[CreateObjectCS](#)

[CreateObjectFromEdges](#)

[CreateObjectFromFaces](#)

[CreateRelativeCS](#)

[DeleteEmptyGroups](#)

[DeleteLastOperation](#)

[DetachFaces](#)

[EditEntityList](#)

[EditFaceCS](#)

[EditObjectCS](#)

[EditRelativeCS](#)

[Export](#)

[ExportModelImageToFile](#)

[ExportModelMeshToFile](#)

[Fillet](#)

[FlattenGroup](#)

[Generate History](#)

[GetActiveCoordinateSystem](#)

[GetCoordinateSystems](#)

[HealObject](#)

[Import](#)

[ImportDXF](#)

[ImportGDSII \[Modeler\]](#)

[Intersect](#)

[MoveCSToEnd](#)

[MoveEntityToGroup](#)

[MoveFaces](#)

[ProjectSheet](#)

[PurgeHistory](#)

[ReplaceWith3DComponent](#)

[Section](#)

[SeparateBody](#)

[SetModelUnits](#)

[SetWCS](#)

[ShowWindow](#)

[Split](#)

[Subtract](#)

[SweepFacesAlongNormal](#)

[ThickenSheet](#)

[UncoverFaces](#)

[Unite](#)

[Ungroup](#)

[WrapSheet](#)

## **AssignMaterial**

Assigns a material to specified object(s).

**UI Access**

**Modeler > Assign Material.**

	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
	<AttributesArray>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Parameters</b>			This script supports the following attributes: <ul style="list-style-type: none"><li>• MaterialValue</li><li>• SolveInside</li><li>• ShellElement</li><li>• ShellElementThickness</li><li>• IsMaterialEditable</li><li>• UseMaterialAppearance</li><li>• IsLightweight</li></ul>
<b>Return Value</b>	None.		

**Python Syntax**

```
AssignMaterial(<SelectionsArray>, <AttributesArray>)
```

**Python Example**

```
oEditor.AssignMaterial(  
    [ "NAME:Selections",  
        "AllowRegionDependentPartSelectionForPMLCreation:=", True,  
        "AllowRegionSelectionForPMLCreation:=", True,  
        "Selections:=" , "Box1"  
    ],  
    [ "NAME:Attributes",  
        "MaterialValue:=" , "diamond",  
        "SolveInside:=" , False,  
        "ShellElement:=" , False,  
        "ShellElementThickness:=" , "nan",  
        "IsMaterialEditable:=" , True,  
        "UseMaterialAppearance:=" , False,  
        "IsLightweight:=" , False  
    ] )
```

**VB Syntax**

```
AssignMaterial <SelectionsArray> <AttributesArray>
```

<b>VB Example</b>	<pre> oEditor.AssignMaterial      Array("NAME:Selections",           "AllowRegionDependentPartSelectionForPMLCreation:=", true,           "AllowRegionSelectionForPMLCreation:=", true,           "Selections:=" , "Box1")      Array("NAME:Attributes",           "MaterialValue:=" , "diamond",           "SolveInside:=" , false,           "ShellElement:=" , false,           "ShellElementThickness:=" , "nan",           "IsMaterialEditable:=" , true,           "UseMaterialAppearance:=" , false,           "IsLightweight:=" , false) </pre>
-------------------	---

## Chamfer

Creates a chamfer.

UI Access	Modeler > Chamfer.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
	<Parameters>	Array	Structured array.  Array ("NAME:Parameters",                  Array ("NAME:ChamferParameters",

			<pre>         "Edges:=" , &lt;array containing integer&gt;,         "Vertices:=" , &lt;array&gt;,         "LeftDistance:=" , &lt;string&gt;,         "RightDistance:=" , &lt;string&gt;,         "ChamferType:=" , &lt;string "Symmetric", "Left Distance-Angle", "Right Distance-Angle", or "Left Distance-Right Distance") ) </pre>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	Chamfer(<SelectionsArray>, <Parameters>)
<b>Python Example</b>	<pre> oEditor.Chamfer(     [         ["NAME:Selections",             "Selections:=" , "Box2",             "NewPartsModelFlag:=" , "Model"         ],         ["NAME:Parameters",             [                 ["NAME:ChamferParameters",                     "Edges:=" , [42],                     "Vertices:=" , [],                     "LeftDistance:=" , "0.1mm",                     "RightDistance:=" , "0.1mm"                 ]             ]         ]     ] ) </pre>

```

    "RightDistance:="           , "0.1mm",
    "ChamferType:="            , "Symmetric"
]
])

```

<b>VB Syntax</b>	Chamfer <SelectionsArray> <Parameters>
<b>VB Example</b>	<pre> oEditor.Chamfer  Array("NAME:Selections",       "Selections:="           , "Box2",       "NewPartsModelFlag:="     , "Model")  Array("NAME:Parameters",       Array("NAME:ChamferParameters",             "Edges:="             , [42],             "Vertices:="          , [],             "LeftDistance:="       , "0.1mm",             "RightDistance:="      , "0.1mm",             "ChamferType:="        , "Symmetric") ) </pre>

## Connect

Connects two or more 1D polyline objects or 2D sheet objects.

**UI Access**

**Modeler > Surface > Connect.**

<b>Parameters</b>	Name <code>&lt;SelectionsArray&gt;</code>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>Connect(&lt;SelectionsArray&gt;)</code>
<b>Python Example</b>	<pre>oEditor.Connect(     ["NAME:Selections",      "Selections:=", "Polyline2,Polyline1"])</pre>

<b>VB Syntax</b>	<code>Connect &lt;SelectionsArray&gt;</code>
<b>VB Example</b>	<pre>oEditor.Connect Array(     "NAME:Selections",     "Selections:=", "Polyline2,Polyline1")</pre>

## CoverLines

Covers two or more 1D objects to form a sheet.

<b>UI Access</b>	<b>Modeler &gt; Surface &gt; Cover Lines.</b>		
<b>Parameters</b>	Name <code>&lt;SelectionsArray&gt;</code>		
<b>Return Value</b>	Type Array		

<b>Python Syntax</b>	CoverLines(<SelectionsArray>)
<b>Python Example</b>	<pre>oEditor.CoverLines([     "NAME:Selections",     "Selections:=", "Polyline3,Polyline4",     "NewPartsModelFlag:=", "Model"])</pre>

<b>VB Syntax</b>	CoverLines <SelectionsArray>
<b>VB Example</b>	<pre>oEditor.CoverLines Array(     "NAME:Selections",     "Selections:=", "Polyline3,Polyline4",     "NewPartsModelFlag:=", "Model")</pre>

## CoverSurfaces

Covers two or more faces to form a solid object.

<b>UI Access</b>	Modeler > Surface > Cover Faces.						
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;SelectionsArray&gt;</td> <td>Array</td> <td>Structured array. See: <a href="#">SelectionsArray</a>.</td> </tr> </table>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
Name	Type	Description					
<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	CoverSurfaces (<SelectionsArray>)
<b>Python Example</b>	<pre>oEditor.CoverSurfaces (</pre>

```
[ "NAME:Selections",
  "Selections:=", "Obj1_Face1,Obj2_Face2",
  "NewPartsModelFlag:=", "Model"
]
```

<b>VB Syntax</b>	CoverSurfaces <SelectionsArray>
<b>VB Example</b>	<pre>oEditor.CoverSurfaces Array(   "NAME:Selections",   "Selections:=", "Obj1_Face1,Obj2_Face2",   "NewPartsModelFlag:=", "Model")</pre>

## CreateEntityList

Creates a list of entities containing objects or faces (not both).

UI Access	Modeler > List > Create > [Object / Face] List.								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array ("NAME:GeometryEntityListParameters",       "EntityType:=" , &lt;string "Object" or "Face"&gt;,       "EntityList:=" , &lt;string of object names or IDs&gt;)</pre> <p>See <a href="#">GetObjectIDByName</a> for returning object IDs.</p> </td> </tr> </tbody> </table>	Name	Type	Description	<Parameters>	Array	<p>Structured array.</p> <pre>Array ("NAME:GeometryEntityListParameters",       "EntityType:=" , &lt;string "Object" or "Face"&gt;,       "EntityList:=" , &lt;string of object names or IDs&gt;)</pre> <p>See <a href="#">GetObjectIDByName</a> for returning object IDs.</p>		
Name	Type	Description							
<Parameters>	Array	<p>Structured array.</p> <pre>Array ("NAME:GeometryEntityListParameters",       "EntityType:=" , &lt;string "Object" or "Face"&gt;,       "EntityList:=" , &lt;string of object names or IDs&gt;)</pre> <p>See <a href="#">GetObjectIDByName</a> for returning object IDs.</p>							

	<code>&lt;AttributesArray&gt;</code>	Array	Structured array. See: <a href="#">AttributesArray</a> . CreateEntityList takes only the "Name" parameter.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>CreateEntityList(&lt;Parameters&gt;,&lt;AttributesArray&gt;)</code>
<b>Python Example</b>	<pre> oEditor.CreateEntityList(     [ "NAME:GeometryEntityListParameters",         "EntityType:=", "Object",         "EntityList:=", "Bondwire1,Bondwire2"     ],     [ "NAME:Attributes",         "Name:=", "Objectlist1"     ] ) </pre>

<b>VB Syntax</b>	<code>CreateEntityList&lt;Parameters&gt; &lt;AttributesArray&gt;</code>
<b>VB Example</b>	<pre> oEditor.CreateEntityList Array(     "NAME:GeometryEntityListParameters",     "EntityType:=", "Object",     "EntityList:=", "Bondwire1,Bondwire2")     Array("NAME:Attributes",     "Name:=", "Objectlist1") ) </pre>

## CreateFaceCS

Creates a Face Coordinate System from a selected face.

UI Access	Modeler > Coordinate System > Create > FaceCS.		
<b>Parameters</b>	Name <i>&lt;Parameters&gt;</i>	Type Array	Description <pre>Structured array.  Array ("NAME:FaceCSParameters",        &lt;OriginArray&gt;,        "MoveToEnd:=", &lt;boolean&gt;,        "FaceID:=", &lt;integer&gt;,        &lt;AxisPosnArray&gt;,        "WhichAxis:=" , &lt;string "X", "Y", or "Z"&gt;,        "ZRotationAngle:=" , &lt;string&gt;,        "XOffset:=" , &lt;string&gt;,        "YOffset:=" , &lt;string&gt;,        "AutoAxis:=" , &lt;boolean&gt;)</pre>
	<i>&lt;OriginArray&gt;</i>	Array	Structured array. <pre>Array ("NAME:Origin",        "IsAttachedToEntity:=" , &lt;boolean&gt;,        "EntityID:=" , &lt;integer&gt;,        "FacetedBodyTriangleIndex:=" , &lt;integer&gt;,</pre>

		<pre> "TriangleVertexIndex:=", &lt;integer&gt;, "PositionType:=", &lt;string "FaceCenter", "EdgeCenter", "OnVertex", "OnEdge", or "OnFace"&gt;, "UParam:=", &lt;float between 0 and 1 representing the relative position of the point on the edge or face&gt;, "VParam:=", &lt;float between 0 and 1 representing the relative position of the point on the edge or face&gt;, "XPosition:=", &lt;string&gt;, "YPosition:=", &lt;string&gt;, "ZPosition:=", &lt;string&gt;)  IsAttachedToEntity specifies whether the point is anchored to a vertex, edge, or face. If True, provide UParam and VParam. If False, provide XPosition, YPosition, and ZPosition to provide fixed position. Pass "0" for unused parameters. </pre>
<AxisPosnArray>	Array	<p>Structured array.</p> <pre> Array("NAME:AxisPosn",       "IsAttachedToEntity:=", &lt;boolean&gt;,       "EntityID:=", &lt;integer&gt;,       "FacetedBodyTriangleIndex:=", &lt;integer&gt;,       "TriangleVertexIndex:=", &lt;integer&gt;,       "PositionType:=", &lt;string "FaceCenter", "EdgeCenter", "OnVertex", "OnEdge", or "OnFace"&gt;,       "UParam:=", &lt;float&gt;, </pre>

			<pre>"VParam:=" , &lt;float&gt;, "XPosition:=" , &lt;string&gt;, "YPosition:=" , &lt;string&gt;, "ZPosition:=" , &lt;string&gt;)</pre>
	<code>&lt;AttributesArray&gt;</code>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>CreateFaceCS(&lt;Parameters&gt;,&lt;AttributesArray&gt;)</code>
<b>Python Example</b>	<pre>oEditor.CreateFaceCS (     [ "NAME:FaceCSParameters",         [ "NAME:Origin",             "IsAttachedToEntity:=" , True,             "EntityID:=" , 46,             "FacetedBodyTriangleIndex:=" , -1,             "TriangleVertexIndex:=" , -1,             "PositionType:=" , "FaceCenter",             "UParam:=" , 0,             "VParam:=" , 0,             "XPosition:=" , "0",             "YPosition:=" , "0",</pre>

```
        "ZPosition:="           , "0"
    ],
    "MoveToEnd:="           , False,
    "FaceID:="              , 46,
    [ "NAME:AxisPosn",
        "IsAttachedToEntity:=" , True,
        "EntityID:="          , 46,
        "FacetedBodyTriangleIndex:=" , -1,
        "TriangleVertexIndex:=" , -1,
        "PositionType:="       , "OnFace",
        "UParam:="              , 0.487129134674319,
        "VParam:="              , 0.308528523557527,
        "XPosition:="           , "1292.27748080459mm",
        "YPosition:="           , "-814.882885865484mm",
        "ZPosition:="           , "0mm"
    ],
    "WhichAxis:="           , "X",
    "ZRotationAngle:="      , "0deg",
    "XOffset:="              , "0mm",
    "YOffset:="              , "0mm",
    "AutoAxis:="             , False
```

```
        ],
        [ "NAME:Attributes",
          "Name:=" , "FaceCS1",
          "PartName:=" , "Rectangle1"
        ]
      )
```

<b>VB Syntax</b>	<b>CreateFaceCS &lt;Parameters&gt; &lt;AttributesArray&gt;</b>
<b>VB Example</b>	<pre> oEditor.CreateFaceCS Array("NAME:FaceCSParameters", Array("NAME:Origin", "IsAttachedToEntity:=", _   true, "EntityID:=", 58, "FacetedBodyTriangleIndex:=", -1, "TriangleVertexIndex:=", _   -1, "PositionType:=", "FaceCenter", "UParam:=", 0, "VParam:=", 0, "XPosition:=", _   "0", "YPosition:=", "0", "ZPosition:=", "0"), "MoveToEnd:=", false, "FaceID:=", _   58, Array("NAME:AxisPosn", "IsAttachedToEntity:=", true, "EntityID:=", 58,   "FacetedBodyTriangleIndex:=", _   -1, "TriangleVertexIndex:=", -1, "PositionType:=", "OnFace", "UParam:=", _   0.0664066713146499, "VParam:=", 0.407014331135309, "XPosition:=", _   "1826.56266852586mm", "YPosition:=", "-1355.79140131881mm", "ZPosition:=", "0mm"),   "WhichAxis:=", _   "X", "ZRotationAngle:=", "0deg", "XOffset:=", "0mm", "YOffset:=", "0mm", "AutoAxis:=",_   -   false), Array("NAME:Attributes", "Name:=", "FaceCS2", "PartName:=", "Rectangle2")</pre>

## CreateGroup

Creates a group from objects specified in the history tree.

UI Access	Modeler > Group > Create.		
Parameters	Name <i>&lt;Parameters&gt;</i>	Type Array	Description Structured array.  Array ("NAME:GroupParameter", "ParentGroupID:=" , <string>, "Parts:=" , <string>, "SubmodelInstances:=" , <string>, "Groups:=" , <string>)
Return Value	None.		

Python Syntax	CreateGroup(<Parameters>)
Python Example	<pre> oEditor.CreateGroup (     [ "NAME:GroupParameter",         "ParentGroupID:="      , "Model",         "Parts:="              , "Box1,Box2,Box3",         "SubmodelInstances:="   , "",         "Groups:="              , ""     ] ) </pre>

<b>VB Syntax</b>	CreateGroup <Parameters>
<b>VB Example</b>	<pre>oEditor.CreateGroup Array("NAME:GroupParameter", "ParentGroupID:=", "Model", _ "Parts:=", "Box1,Box2,Box3", "SubmodelInstances:=", "", "Groups:=", "")</pre>

## CreateObjectCS

Creates an object coordinate system from a selected object.

UI Access	<b>Modeler &gt; Coordinate System &gt; Create &gt; Object &gt; [Offset / Rotated / Both].</b>		
<b>Parameters</b>	Name <Parameters>	Type Array	Description Structured array.  Array ("NAME:ObjectCSParameters",  <OriginArray>  "MoveToEnd:=" , <boolean>,  "ReverseXAxis:=" , <boolean>,  "ReverseYAxis:=" , <boolean>,  <xAxisArray / xAxisPosArray>  <yAxisArray / yAxisPosArray>)  <b>Note:</b> xAxisArray and xAxisPosArray differ. Use xAxisArray for absolute position and xAxisPosArray for relative position. Do the same for yAxisArray and yAxisPosArray.
	<OriginArray>	Array	Structured array.

		<pre> Array("NAME:Origin",       "IsAttachedToEntity:=" , &lt;boolean&gt;,       "EntityID:=" , &lt;integer&gt;,       "FacetedBodyTriangleIndex:=" , &lt;integer&gt;,       "TriangleVertexIndex:=" , &lt;integer&gt;,       "PositionType:=" , &lt;string "OnVertex", "EdgeCenter", "FaceCenter", "OnEdge", or "Abso- lutePosition"&gt;,       "UParam:=" , &lt;float between 0 and 1 representing the relative position of the point on the edge or face&gt;,       "VParam:=" , &lt;float between 0 and 1 representing the relative position of the point on the edge or face&gt;,       "XPosition:=" , &lt;string&gt;,       "YPosition:=" , &lt;string&gt;,       "ZPosition:=" , &lt;string&gt;) IsAttachedToEntity specifies whether the point is anchored. If True, provide UParam and VParam. If False, provide XPosition, YPosition, and ZPosition to provide fixed position. Pass "0" for unused parameters. </pre>
<xAxisArray>	Array	<p>Structured array for absolute position:</p> <pre> Array("NAME:xAxis",       "DirectionType:=" , "AbsoluteDirection",       "EdgeID:=" , &lt;integer&gt;,       "FaceID:=" , &lt;integer&gt;, </pre>

			<pre>"xDirection:=" , &lt;string&gt;, "yDirection:=" , &lt;string&gt;, "zDirection:=" , &lt;string&gt;, "UParam:=" , &lt;float&gt;, "VParam:=" , &lt;float&gt;)</pre>
	<b>&lt;xAxisPosArray&gt;</b>	Array	<p>Structured array for relative position:</p> <pre>Array ("NAME:xAxisPos",       "IsAttachedToEntity:=" , &lt;boolean&gt;,       "EntityID:=" , &lt;integer&gt;,       "FacetedBodyTriangleIndex:=" , &lt;integer&gt;,       "TriangleVertexIndex:=" , &lt;integer&gt;,       "PositionType:=" , &lt;string "OnVertex", "EdgeCenter", "FaceCenter", or "OnEdge"&gt;,       "UParam:=" , &lt;float&gt;,       "VParam:=" , &lt;float&gt;,       "XPosition:=" , &lt;string&gt;,       "YPosition:=" , &lt;string&gt;,       "ZPosition:=" , &lt;string&gt;)</pre>
	<b>&lt;yAxisArray&gt;</b>	Array	<p>Structured array for absolute position:</p> <pre>Array ("NAME:yAxis",       "DirectionType:=" , "AbsoluteDirection",</pre>

			<pre>"EdgeID:=" , &lt;integer&gt;, "FaceID:=" , &lt;integer&gt;, "xDirection:=" , &lt;string&gt;, &gt;yDirection:=" , &lt;string&gt;, "zDirection:=" , &lt;string&gt;, "UParam:=" , &lt;float&gt;, "VParam:=" , &lt;float&gt;)</pre>
	<i>&lt;yAxisPosArray&gt;</i>	Array	<p>Structured array for relative position:</p> <pre>Array ("NAME:yAxisPos", "IsAttachedToEntity:=" , &lt;boolean&gt;, "EntityID:=" , &lt;integer&gt;, "FacetedBodyTriangleIndex:=" , &lt;integer&gt;, "TriangleVertexIndex:=" , &lt;integer&gt;, "PositionType:=" , &lt;string "OnVertex", "EdgeCenter", "FaceCenter", or "OnEdge"&gt;, "UParam:=" , &lt;float&gt;, "VParam:=" , &lt;float&gt;, "XPosition:=" , &lt;string&gt;, "YPosition:=" , &lt;string&gt;, "ZPosition:=" , &lt;string&gt;)</pre>
	<i>&lt;AttributesArray&gt;</i>	Array	Structured array. See: <a href="#">AttributesArray</a> .
<b>Return Value</b>	None.		

Python Syntax	CreateObjectCS(<Parameters>,<AttributesArray>)
Python Example	<pre>oEditor.CreateObjectCS(     [ "NAME:ObjectCSParameters",         [ "NAME:Origin",             "IsAttachedToEntity:=" , True,             "EntityID:=" , 59,             "FacetedBodyTriangleIndex:=" , -1,             "TriangleVertexIndex:=" , -1,             "PositionType:=" , "OnVertex",             "UParam:=" , 0,             "VParam:=" , 0,             "XPosition:=" , "0",             "YPosition:=" , "0",             "ZPosition:=" , "0"         ],         "MoveToEnd:=" , False,         "ReverseXAxis:=" , False,         "ReverseYAxis:=" , False,         [ "NAME:xAxis",     ]]</pre>

```
    "DirectionType:="      , "AbsoluteDirection",
    "EdgeID:="            , -1,
    "FaceID:="            , -1,
    "xDirection:="        , "1",
    "yDirection:="        , "0",
    "zDirection:="        , "0",
    "UParam:="            , 0,
    "VParam:="            , 0
  ],
  [
    "NAME:yAxis",
    "DirectionType:="      , "AbsoluteDirection",
    "EdgeID:="            , -1,
    "FaceID:="            , -1,
    "xDirection:="        , "0",
    "yDirection:="        , "1",
    "zDirection:="        , "0",
    "UParam:="            , 0,
    "VParam:="            , 0
  ]
],
[ "NAME:Attributes",
```

	<pre>         "Name:="                  , "ObjectCS1",         "PartName:="              , "Box2"     ] ) </pre>
--	--

VB Syntax	CreateObjectCS <Parameters> <AttributesArray>
VB Example	<pre> oEditor.CreateObjectCS Array("NAME:ObjectCSParameters", Array("NAME:Origin", "IsAttachedToEntity:=", _ false, "EntityID:=", -1, "FacetedBodyTriangleIndex:=", -1, "TriangleVertexIndex:=", _ -1, "PositionType:=", "AbsolutePosition", "UParam:=", 0, "VParam:=", 0, "XPosition:=", _ "0mm", "YPosition:=", "0mm", "ZPosition:=", "0mm"), "MoveToEnd:=", false, "ReverseXAxis:=", _ false, "ReverseYAxis:=", false, Array("NAME:xAxisPos", "IsAttachedToEntity:=", true, _ "EntityID:=", _ 80, "FacetedBodyTriangleIndex:=", -1, "TriangleVertexIndex:=", -1, "PositionType:=", _ "EdgeCenter", "UParam:=", 0, "VParam:=", 0, "XPosition:=", "0", "YPosition:=", _ "0", "ZPosition:=", "0"), Array("NAME:yAxisPos", "IsAttachedToEntity:=", true, _ "EntityID:=", _ 69, "FacetedBodyTriangleIndex:=", -1, "TriangleVertexIndex:=", -1, "PositionType:=", _ "EdgeCenter", "UParam:=", 0, "VParam:=", 0, "XPosition:=", "0", "YPosition:=", _ "0", "ZPosition:=", "0")), Array("NAME:Attributes", "Name:=", "ObjectCS2", _ "PartName:=", _ </pre>

	"Box3")
--	---------

## CreateObjectFromEdges

Creates an object from the specified object edge.

UI Access	Modeler > Edge > Create Object From Edge		
Parameters	Name <i>&lt;SelectionsArray&gt;</i>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
	<i>&lt;ParametersArray&gt;</i>	Array	Structured array.  Array ("NAME:Parameters",  Array ("NAME:BodyFromEdgeToPara- meters",  "Edges:=" , <array containing integer edges>  ) )
	<i>&lt;CreateGroupsForNewObjects&gt;</i>	Array	Structured array.  Array ("CreateGroupsForNewObjects:=", <boolean True to create groups for new objects; else False>)
Return Value	None.		

Python Syntax	CreateObjectFromEdges(<SelectionsArray>, <ParametersArray>, <CreateGroupsForNewObjects>)
Python Example	<code>oEditor.CreateObjectFromEdges (</code>

```
[ "NAME:Selections",
  "Selections:=", "Box2",
  "NewPartsModelFlag:=", "Model"
],
[ "NAME:Parameters",
  [ "NAME:BodyFromEdgeToParameters",
    "Edges:=", [41]
],
[ "CreateGroupsForNewObjects:=", False
])
```

<b>VB Syntax</b>	CreateObjectFromEdges <SelectionsArray> <ParametersArray> <CreateGroupsForNewObjects>
<b>VB Example</b>	<pre>oEditor.CreateObjectFromEdges Array("NAME:Selections", "Selections:=", "Box1",   "NewPartsModelFlag:=",   "Model"), Array("NAME:Parameters", Array("NAME:BodyFromEdgeToParameters", "Edges:=",   Array(     13))), Array("CreateGroupsForNewObjects:=", false)</pre>

## CreateObjectFromFaces

Creates 2D objects from specified face(s).

UI Access	Modeler > Surface > Create Object from Face		
<b>Parameters</b>	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
	<Parameters>	Array	Structured array. Array ("NAME:Parameters", <BodyFromFaceToParameters>)
	<CreateGroupsForNewObjects>	Array	Structured array. Array ("CreateGroupsForNewObjects:=", <boolean>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateObjectFromFaces (<SelectionsArray>,<Parameters>,<CreateGroupsForNewObjects>)
<b>Python Example</b>	<pre> oEditor.CreateObjectFromFaces (     [         ["NAME:Selections",             "Selections:="           , "Box3",             "NewPartsModelFlag:="     , "Model"         ],         ["NAME:Parameters",             ["NAME:BodyFromFaceToParameters",                 "FacesToDetach:="       , [68]             ]         ],     ], ) </pre>

	<pre>[ "CreateGroupsForNewObjects:=", False ])</pre>
--	--

<b>VB Syntax</b>	CreateObjectFromFaces <SelectionsArray> <Parameters> <CreateGroupsForNewObjects>
<b>VB Example</b>	<pre>oEditor.CreateObjectFromFaces Array("NAME:Selections", "Selections:=", "Box3", "NewPartsModelFlag:=" , "Model") Array("NAME:Parameters", Array("NAME:BodyFromFaceToParameters", "FacesToDetach:=", Array(68)))  Array("CreateGroupsForNewObjects:=", False)</pre>

## CreateRelativeCS

Creates a Relative Coordinate System.

UI Access	<b>Modeler &gt; Coordinate System &gt; Create &gt; Relative CS &gt; [Offset / Rotated / Both].</b>								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array ("NAME:RelativeCSPARAMETERS",       "Mode:=" , "Axis/Position",       "OriginX:=" , &lt;string&gt;,       "OriginY:=" , &lt;string&gt;,       "OriginZ:=" , &lt;string&gt;,       "XAxisXvec:=" , &lt;string&gt;,</pre> </td> </tr> </tbody> </table>	Name	Type	Description	<Parameters>	Array	<p>Structured array.</p> <pre>Array ("NAME:RelativeCSPARAMETERS",       "Mode:=" , "Axis/Position",       "OriginX:=" , &lt;string&gt;,       "OriginY:=" , &lt;string&gt;,       "OriginZ:=" , &lt;string&gt;,       "XAxisXvec:=" , &lt;string&gt;,</pre>		
Name	Type	Description							
<Parameters>	Array	<p>Structured array.</p> <pre>Array ("NAME:RelativeCSPARAMETERS",       "Mode:=" , "Axis/Position",       "OriginX:=" , &lt;string&gt;,       "OriginY:=" , &lt;string&gt;,       "OriginZ:=" , &lt;string&gt;,       "XAxisXvec:=" , &lt;string&gt;,</pre>							

			<pre>"XAxisYvec:=", &lt;string&gt;, "XAxisZvec:=", &lt;string&gt;, "YAxisXvec:=", &lt;string&gt;, "YAxisYvec:=", &lt;string&gt;, "YAxisZvec:=", &lt;string&gt;)</pre>
	<i>&lt;AttributesArray&gt;</i>	Array	Structured array. See: <a href="#">AttributesArray</a> . CreateRelativeCS supports only the "Name" parameter.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CreateRelativeCS(<Parameters>,<AttributesArray>)
<b>Python Example</b>	<pre>oEditor.CreateRelativeCS(     [         "NAME:RelativeCSParameters",         "Mode:=", "Axis/Position",         "OriginX:=", "0.62mm",         "OriginY:=", "-0.7mm",         "OriginZ:=", "0mm",         "XAxisXvec:=", "1mm",         "XAxisYvec:=", "0mm",         "XAxisZvec:=", "0mm",         "YAxisXvec:=", "0mm",         "YAxisYvec:=", "1mm",         "YAxisZvec:=", "0mm"     ] )</pre>

```
        ],
        [
        "NAME:Attributes",
            "Name:=" , "RelativeCS1"
        ]
    )
```

<b>VB Syntax</b>	CreateRelativeCS <Parameters> <AttributesArray>
<b>VB Example</b>	<pre>oEditor.CreateRelativeCS Array("NAME:RelativeCSParameters", "Mode:=", "Axis/Position", "OriginX:=", _ "0mm", "OriginY:=", "0mm", "OriginZ:=", "0mm", "XAxisXvec:=", "0.66mm", "XAxisYvec:=", - "0.28mm", "XAxisZvec:=", "0mm", "YAxisXvec:=", "0.06mm", "YAxisYvec:=", "0.14mm", "YAx- isZvec:=", _ "0mm"), Array("NAME:Attributes", "Name:=", "RelativeCS1")</pre>

## DeleteEmptyGroups

Deletes group(s) from the history tree.

<b>UI Access</b>	<b>Modeler &gt; Group &gt; Delete Empty.</b>		
<b>Parameters</b>	Name	Type	Description
	<Parameters>	Array	<p>Structured array.</p> <pre>Array("Groups:=" , &lt;array of string group IDs&gt;)</pre>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	DeleteEmptyGroups(<Parameters>)
<b>Python Example</b>	<pre>oEditor.DeleteEmptyGroups ( [     "Groups:=",  ["Group1", "Group2", "Group3"] ])</pre>

<b>VB Syntax</b>	DeleteEmptyGroups <Parameters>
<b>VB Example</b>	<pre>oEditor.DeleteEmptyGroups Array("Groups:=", Array("Group1", "Group2", "Group3"))</pre>

## DeleteLastOperation

Deletes the last operation performed on the specified object(s).

<b>UI Access</b>	Modeler > Delete Last Operation.			
<b>Parameters</b>	<table border="1"> <tr> <td>Name &lt;SelectionsArray&gt;</td> <td>Type Array</td> <td>Description Structured array. See: <a href="#">SelectionsArray</a>.</td> </tr> </table>	Name <SelectionsArray>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
Name <SelectionsArray>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .		
<b>Return Value</b>	None.			

<b>Python Syntax</b>	DeleteLastOperation(<SelectionsArray>)
<b>Python Example</b>	<pre>oEditor.DeleteLastOperation ( [     "NAME:Selections",     "Selections:=", "Box3",     "NewPartsModelFlag:=", "Model" ])</pre>

<b>VB Syntax</b>	DeleteLastOperation <SelectionsArray>
<b>VB Example</b>	<pre>oEditor.DeleteLastOperation Array("NAME:Selections", "Selections:=", "Box3", "NewPartsModelFlag:=", "Model")</pre>

## DetachFaces

Detaches the specified face(s) from an object.

UI Access	Modeler > Surface > Detach Faces.		
	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
Parameters	<Parameters>	Array	Structured array. Array ("NAME:Parameters", <DetachFacesArray>)
	<DetachFacesArray>	Array	Structured array. Array ("NAME:DetachFacesToParameters", "FacesToDetach:=" , <array containing integer face IDs>)
Return Value	None.		

<b>Python Syntax</b>	DetachFaces(<SelectionsArray>, <Parameters>)
<b>Python Example</b>	<pre>oEditor.DetachFaces (</pre>

```
[ "NAME:Selections",
  "Selections:=", "Box3",
  "NewPartsModelFlag:=", "Model"
],
[ "NAME:Parameters",
  [ "NAME:DetachFacesToParameters",
    "FacesToDetach:=", [68, 67]
  ]
])
```

<b>VB Syntax</b>	DetachFaces <SelectionsArray> <Parameters>
<b>VB Example</b>	<pre>oEditor.DetachFaces Array("NAME:Selections", "Selections:=", "Box3", "NewPartsModelFlag:=", "Model") Array("NAME:Parameters", Array("NAME:DetachFacesToParameters", "FacesToDetach:=", [68, 67]))</pre>

## EditEntityList

Modifies an entity list.

<b>UI Access</b>	<b>Modeler &gt; List &gt; Reassign.</b>		
<b>Parameters</b>	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
	<Parameters>	Array	Structured array.

			Array ("NAME:GeometryEntityListParameters", "EntityType:=" , <string "Object" or "Face">, "EntityList:=" , <string list> ) )
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EditEntityList(<SelectionsArray>, <Parameters>)
<b>Python Example</b>	<pre>oEditor.EditEntityList(           ["NAME:Selections",           "Selections:="                  , "Objectlist1"           ],           ["NAME:GeometryEntityListParameters",           "EntityType:="                  , "Object",           "EntityList:="                  , "Box1, Box2, Box3"           ])</pre>

<b>VB Syntax</b>	EditEntityList <SelectionsArray> <Parameters>
<b>VB Example</b>	<pre>oEditor.EditEntityList Array ("NAME:Selections",</pre>

```

    "Selections:=", "Objectlist1")
Array("NAME:GeometryEntityListParameters",
      "EntityType:=", "Object",
      "EntityList:=", "Box1, Box2, Box3")

```

## EditFaceCS

Recreates an existing face coordinate system. See: [CreateFaceCS](#).

UI Access	Modeler > Coordinate System > Edit.		
<b>Parameters</b>	Name <i>&lt;Parameters&gt;</i>	Type Array	Description Structured array.  Array ("NAME:FaceCSParameters",       <OriginArray>,       "MoveToEnd:=", <boolean>,       "FaceID:=", <integer>,       <AxisPosnArray>,       "WhichAxis:=" , <string "X", "Y", or "Z">,       "ZRotationAngle:=" , <string>,       "XOffset:=" , <string>,       "YOffset:=" , <string>,       "AutoAxis:=" , <boolean>)
	<OriginArray>	Array	Structured array.  Array ("NAME:Origin",

			<pre>     "IsAttachedToEntity:=" , &lt;boolean&gt;,     "EntityID:=" , &lt;integer&gt;,     "FacetedBodyTriangleIndex:=" , &lt;integer&gt;,     "TriangleVertexIndex:=" , &lt;integer&gt;,     "PositionType:=" , &lt;string "FaceCenter", "EdgeCenter", "OnVertex", "OnEdge", or "OnFace"&gt;,     "UParam:=" , &lt;float between 0 and 1 representing the relative position of the point on the edge or face&gt;,     "VParam:=" , &lt;float between 0 and 1 representing the relative position of the point on the edge or face&gt;,     "XPosition:=" , &lt;string&gt;,     "YPosition:=" , &lt;string&gt;,     "ZPosition:=" , &lt;string&gt; </pre> <p>IsAttachedToEntity specifies whether the point is anchored to a vertex, edge, or face. If True, provide UParam and VParam. If False, provide XPosition, YPosition, and ZPosition to provide fixed position. Pass "0" for unused parameters.</p>
	<b>&lt;AxisPosnArray&gt;</b>	Array	<p>Structured array.</p> <pre> Array("NAME:AxisPosn",     "IsAttachedToEntity:=" , &lt;boolean&gt;,     "EntityID:=" , &lt;integer&gt;, </pre>

			<pre> "FacetedBodyTriangleIndex:=" , &lt;integer&gt;, "TriangleVertexIndex:=" , &lt;integer&gt;, "PositionType:=" , &lt;string "FaceCenter", "EdgeCenter", "OnVertex", "OnEdge", or "OnFace"&gt;, "UParam:=" , &lt;float&gt;, "VParam:=" , &lt;float&gt;, "XPosition:=" , &lt;string&gt;, "YPosition:=" , &lt;string&gt;, "ZPosition:=" , &lt;string&gt; </pre>
	<b>&lt;AttributesArray&gt;</b>	Array	Structured array. See: <a href="#">AttributesArray</a> . Use to select the coordinate system to edit.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EditFaceCS (<Parameters>, <AttributesArray>)
<b>Python Example</b>	<pre> oEditor.EditFaceCS (     [ "NAME:FaceCSPARAMETERS",         [ "NAME:Origin",             "IsAttachedToEntity:=" , True,             "EntityID:=" , 12,             "FacetedBodyTriangleIndex:=" , -1,             "TriangleVertexIndex:=" , -1,             "PositionType:=" , "FaceCenter", </pre>

```
"UParam:=" , 0,
"VParam:=" , 0,
"XPosition:=" , "0",
"YPosition:=" , "0",
"ZPosition:=" , "0"
],
"MoveToEnd:=" , False,
"FaceID:=" , 12,
["NAME:AxisPosn",
 "IsAttachedToEntity:=" , True,
 "EntityID:=" , 12,
 "FacetedBodyTriangleIndex:=" , -1,
 "TriangleVertexIndex:=" , -1,
 "PositionType:=" , "OnFace",
 "UParam:=" , 0.62951717774066,
 "VParam:=" , 0.226514925559344,
 "XPosition:=" , "1200mm",
 "YPosition:=" , "-354.697014888131mm",
 "ZPosition:=" , "125.903435548132mm"
],
```

```

    "WhichAxis:="           , "X",
    "ZRotationAngle:="     , "0deg",
    "XOffset:="            , "0mm",
    "YOffset:="             , "0mm",
    "AutoAxis:="            , False
],
[ "NAME:Attributes",
  "Name:="                , "FaceCS1",
  "PartName:="             , "Box1"
])

```

VB Syntax	EditFaceCS <Parameters> <AttributesArray>
VB Example	<pre> oEditor.EditFaceCS  Array("NAME:FaceCSParameters",       Array("NAME:Origin",             "IsAttachedToEntity:=" , True,             "EntityID:="          , 12,             "FacetedBodyTriangleIndex:=" , -1,             "TriangleVertexIndex:=" , -1,             "PositionType:="        , "FaceCenter",             "UParam:="              , 0, </pre>

```
    "VParam:=" , 0,
    "XPosition:=" , "0",
    "YPosition:=" , "0",
    "ZPosition:=" , "0"),
    "MoveToEnd:=" , False,
    "FaceID:=" , 12,
    Array("NAME:AxisPosn",
        "IsAttachedToEntity:=" , True,
        "EntityID:=" , 12,
        "FacetedBodyTriangleIndex:=" , -1,
        "TriangleVertexIndex:=" , -1,
        "PositionType:=" , "OnFace",
        "UParam:=" , 0.62951717774066,
        "VParam:=" , 0.226514925559344,
        "XPosition:=" , "1200mm",
        "YPosition:=" , "-354.697014888131mm",
        "ZPosition:=" , "125.903435548132mm"),
    "WhichAxis:=" , "X",
    "ZRotationAngle:=" , "0deg",
    "XOffset:=" , "0mm",
```

```

        "YOffset:="           , "0mm",
        "AutoAxis:="          , False)

Array("NAME:Attributes",
      "Name:="              , "FaceCS1",
      "PartName:="           , "Box1")

```

## EditObjectCS

Edits an existing object coordinate system. See: [CreateObjectCS](#).

UI Access	Modeler > Coordinate System > Edit.		
<b>Parameters</b>	Name <i>&lt;Parameters&gt;</i>	Type Array	Description <p>Structured array.</p> <pre> Array("NAME:ObjectCSParameters",       &lt;OriginArray&gt;       "MoveToEnd:=" , &lt;boolean&gt;,       "ReverseXAxis:=" , &lt;boolean&gt;,       "ReverseYAxis:=" , &lt;boolean&gt;,       &lt;xAxisArray / xAxisPosArray&gt;       &lt;yAxisArray / yAxisPosArray&gt; ) </pre> <p><b>Note:</b> xAxisArray and xAxisPosArray differ. Use xAxisArray for absolute position and xAxisPosArray for relative position. Do the same for yAxisArray and yAxisPosArray.</p>
	<OriginArray>	Array	Structured array. <pre> Array("NAME:Origin", </pre>

		<pre>     "IsAttachedToEntity:=" , &lt;boolean&gt;,     "EntityID:=" , &lt;integer&gt;,     "FacetedBodyTriangleIndex:=" , &lt;integer&gt;,     "TriangleVertexIndex:=" , &lt;integer&gt;,     "PositionType:=" , &lt;string "OnVertex", "EdgeCenter", "FaceCenter", "OnEdge", or "Abso- lutePosition"&gt;,     "UParam:=" , &lt;float between 0 and 1 representing the relative position of the point on the edge or face&gt;,     "VParam:=" , &lt;float between 0 and 1 representing the relative position of the point on the edge or face&gt;,     "XPosition:=" , &lt;string&gt;,     "YPosition:=" , &lt;string&gt;,     "ZPosition:=" , &lt;string&gt; </pre> <p>IsAttachedToEntity specifies whether the point is anchored. If True, provide UParam and VParam. If False, provide XPosition, YPosition, and ZPosition to provide fixed position. Pass "0" for unused parameters.</p>
<xAxisArray>	Array	<p>Structured array for absolute position:</p> <pre> Array("NAME:xAxis",       "DirectionType:=" , "AbsoluteDirection",       "EdgeID:=" , &lt;integer&gt;, </pre>

			<pre>"FaceID:=" , &lt;integer&gt;, "xDirection:=" , &lt;string&gt;, &gt;yDirection:=" , &lt;string&gt;, "zDirection:=" , &lt;string&gt;, "UParam:=" , &lt;float&gt;, "VParam:=" , &lt;float&gt;)</pre>
	<code>&lt;xAxisPosArray&gt;</code>	Array	<p>Structured array for relative position:</p> <pre>Array ("NAME:xAxisPos", "IsAttachedToEntity:=" , &lt;boolean&gt;, "EntityID:=" , &lt;integer&gt;, "FacetedBodyTriangleIndex:=" , &lt;integer&gt;, "TriangleVertexIndex:=" , &lt;integer&gt;, "PositionType:=" , &lt;string "OnVertex", "EdgeCenter", "FaceCenter", or "OnEdge"&gt;, "UParam:=" , &lt;float&gt;, "VParam:=" , &lt;float&gt;, "XPosition:=" , &lt;string&gt;, "YPosition:=" , &lt;string&gt;, "ZPosition:=" , &lt;string&gt;)</pre>
	<code>&lt;yAxisArray&gt;</code>	Array	<p>Structured array for absolute position:</p> <pre>Array ("NAME:yAxis", "DirectionType:=" , "AbsoluteDirection",</pre>

			<pre>"EdgeID:=" , &lt;integer&gt;, "FaceID:=" , &lt;integer&gt;, "xDirection:=" , &lt;string&gt;, "yDirection:=" , &lt;string&gt;, "zDirection:=" , &lt;string&gt;, "UParam:=" , &lt;float&gt;, "VParam:=" , &lt;float&gt;)</pre>
<i>&lt;yAxisPosArray&gt;</i>	Array	Structured array for relative position:	<pre>Array ("NAME:yAxisPos",       "IsAttachedToEntity:=" , &lt;boolean&gt;,       "EntityID:=" , &lt;integer&gt;,       "FacetedBodyTriangleIndex:=" , &lt;integer&gt;,       "TriangleVertexIndex:=" , &lt;integer&gt;,       "PositionType:=" , &lt;string "OnVertex",       "EdgeCenter", "FaceCenter", or "OnEdge"&gt;,       "UParam:=" , &lt;float&gt;,       "VParam:=" , &lt;float&gt;,       "XPosition:=" , &lt;string&gt;,       "YPosition:=" , &lt;string&gt;,       "ZPosition:=" , &lt;string&gt;)</pre>
<i>&lt;AttributesArray&gt;</i>	Array	Structured array. See: <a href="#">AttributesArray</a> . Use to select the coordinate sys-	

		tem to edit.
<b>Return Value</b>	None.	

<b>Python Syntax</b>	EditObjectCS(<Parameters>,<AttributesArray>)
	<pre> oEditor.EditObjectCS(     [         "NAME:ObjectCSParameters",         ["NAME:Origin",             "IsAttachedToEntity:=", True,             "EntityID:=", 59,             "FacetedBodyTriangleIndex:=", -1,             "TriangleVertexIndex:=", -1,             "PositionType:=", "OnVertex",             "UParam:=", 0,             "VParam:=", 0,             "XPosition:=", "0",             "YPosition:=", "0",             "ZPosition:=", "0"         ],         "MoveToEnd:=", False,         "ReverseXAxis:=", False,         "ReverseYAxis:=", False     ] ) </pre>
<b>Python Example</b>	

```
[ "NAME:xAxis",
    "DirectionType:=" , "AbsoluteDirection",
    "EdgeID:=" , -1,
    "FaceID:=" , -1,
    "xDirection:=" , "1",
    "yDirection:=" , "0",
    "zDirection:=" , "0",
    "UParam:=" , 0,
    "VParam:=" , 0
] ,
[ "NAME:yAxis",
    "DirectionType:=" , "AbsoluteDirection",
    "EdgeID:=" , -1,
    "FaceID:=" , -1,
    "xDirection:=" , "0",
    "yDirection:=" , "1",
    "zDirection:=" , "0",
    "UParam:=" , 0,
    "VParam:=" , 0
]
```

```
],
["NAME:Attributes",
 "Name:=" , "ObjectCS1",
 "PartName:=" , "Box2"
])
```

<b>VB Syntax</b> <pre>EditObjectCS &lt;Parameters&gt; &lt;AttributesArray&gt;</pre>	
<b>VB Example</b> <pre>oEditor.EditObjectCS Array("NAME:ObjectCSParameters", Array("NAME:Origin", "IsAttachedToEntity:=",  false, "EntityID:=", -1, "FacetedBodyTriangleIndex:=", -1, "TriangleVertexIndex:=",  -1, "PositionType:=", "AbsolutePosition", "UParam:=", 0, "VParam:=", 0, "XPosition:=",  "0mm", "YPosition:=", "0mm", "ZPosition:=", "0mm"), "MoveToEnd:=", false, "ReverseXAxis:=",  false, "ReverseYAxis:=", false, Array("NAME:xAxisPos", "IsAttachedToEntity:=", true,  "EntityID:=",  80, "FacetedBodyTriangleIndex:=", -1, "TriangleVertexIndex:=", -1, "PositionType:=",  "EdgeCenter", "UParam:=", 0, "VParam:=", 0, "XPosition:=", "0", "YPosition:=",  "0", "ZPosition:=", "0"), Array("NAME:yAxisPos", "IsAttachedToEntity:=", true,  "EntityID:=",  69, "FacetedBodyTriangleIndex:=", -1, "TriangleVertexIndex:=", -1, "PositionType:=",  "EdgeCenter", "UParam:=", 0, "VParam:=", 0, "XPosition:=", "0", "YPosition:=",  "0", "ZPosition:=", "0")), Array("NAME:Attributes", "Name:=", "ObjectCS2",</pre>	

```
"PartName:=",  
  "  
"Box3")
```

## EditRelativeCS

Edits an existing Relative Coordinate System. See: [CreateRelativeCS](#).

UI Access	Modeler > Coordinate System > Edit.								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>&lt;Parameters&gt;</i></td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array ("NAME:RelativeCSParameters",       "Mode:=" , "Axis/Position",       "OriginX:=" , &lt;string&gt;,       "OriginY:=" , &lt;string&gt;,       "OriginZ:=" , &lt;string&gt;,       "XAxisXvec:=" , &lt;string&gt;,       "XAxisYvec:=" , &lt;string&gt;,       "XAxisZvec:=" , &lt;string&gt;,       "YAxisXvec:=" , &lt;string&gt;,       "YAxisYvec:=" , &lt;string&gt;,       "YAxisZvec:=" , &lt;string&gt;)</pre> </td> </tr> </tbody> </table>	Name	Type	Description	<i>&lt;Parameters&gt;</i>	Array	<p>Structured array.</p> <pre>Array ("NAME:RelativeCSParameters",       "Mode:=" , "Axis/Position",       "OriginX:=" , &lt;string&gt;,       "OriginY:=" , &lt;string&gt;,       "OriginZ:=" , &lt;string&gt;,       "XAxisXvec:=" , &lt;string&gt;,       "XAxisYvec:=" , &lt;string&gt;,       "XAxisZvec:=" , &lt;string&gt;,       "YAxisXvec:=" , &lt;string&gt;,       "YAxisYvec:=" , &lt;string&gt;,       "YAxisZvec:=" , &lt;string&gt;)</pre>		
Name	Type	Description							
<i>&lt;Parameters&gt;</i>	Array	<p>Structured array.</p> <pre>Array ("NAME:RelativeCSParameters",       "Mode:=" , "Axis/Position",       "OriginX:=" , &lt;string&gt;,       "OriginY:=" , &lt;string&gt;,       "OriginZ:=" , &lt;string&gt;,       "XAxisXvec:=" , &lt;string&gt;,       "XAxisYvec:=" , &lt;string&gt;,       "XAxisZvec:=" , &lt;string&gt;,       "YAxisXvec:=" , &lt;string&gt;,       "YAxisYvec:=" , &lt;string&gt;,       "YAxisZvec:=" , &lt;string&gt;)</pre>							
<table border="1"> <tbody> <tr> <td><i>&lt;AttributesArray&gt;</i></td> <td>Array</td> <td>Structured array. See: <a href="#">AttributesArray</a>. Use to select the coordinate system to edit.</td></tr> </tbody> </table>	<i>&lt;AttributesArray&gt;</i>	Array	Structured array. See: <a href="#">AttributesArray</a> . Use to select the coordinate system to edit.						
<i>&lt;AttributesArray&gt;</i>	Array	Structured array. See: <a href="#">AttributesArray</a> . Use to select the coordinate system to edit.							

<b>Return Value</b>	None.
---------------------	-------

<b>Python Syntax</b>	EditRelativeCS(<Parameters>,<AttributesArray>)
<b>Python Example</b>	<pre> oEditor.EditRelativeCS(     [         "NAME:RelativeCSPARAMETERS",         "Mode:=" , "Axis/Position",         "OriginX:=" , "0.62mm",         "OriginY:=" , "-0.7mm",         "OriginZ:=" , "0mm",         "XAxisXvec:=" , "1mm",         "XAxisYvec:=" , "0mm",         "XAxisZvec:=" , "0mm",         "YAxisXvec:=" , "0mm",         "YAxisYvec:=" , "1mm",         "YAxisZvec:=" , "0mm"     ],     [         "NAME:ATTRIBUTES",         "Name:=" , "RelativeCS1"     ] ) </pre>

<b>VB Syn- tax</b>	EditRelativeCS <Parameters> <AttributesArray>
------------------------	---

**VB Example**

```

oEditor.EditRelativeCS Array("NAME:RelativeCSParameters", "Mode:=", "Axis/Position",
"OriginX:=", _
"0mm", "OriginY:=", "0mm", "OriginZ:=", "0mm", "XAxisXvec:=", "0.66mm", "XAxisYvec:=",
-
"0.28mm", "XAxisZvec:=", "0mm", "YAxisXvec:=", "0.06mm", "YAxisYvec:=", "0.14mm", "YAx-
isZvec:=", _
"0mm"), Array("NAME:Attributes", "Name:=", "RelativeCS1")

```

**Export**

Exports the model to a file.

**Note:**

This script does not export image file types or GDSII files. See: [ExportModelImageToFile](#) and [ExportGDSII](#).

UI Access	Modeler > Export.								
Parameters	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre> Array ("NAME:ExportParameters",       "AllowRe- gionDependentPartSelectionForPMLCreation:=",       &lt;boolean&gt;,       "AllowRegionSelectionForPMLCreation:=",       &lt;boolean&gt;,       </pre> </td> </tr> </table>	Name	Type	Description	<Parameters>	Array	<p>Structured array.</p> <pre> Array ("NAME:ExportParameters",       "AllowRe- gionDependentPartSelectionForPMLCreation:=",       &lt;boolean&gt;,       "AllowRegionSelectionForPMLCreation:=",       &lt;boolean&gt;,       </pre>		
Name	Type	Description							
<Parameters>	Array	<p>Structured array.</p> <pre> Array ("NAME:ExportParameters",       "AllowRe- gionDependentPartSelectionForPMLCreation:=",       &lt;boolean&gt;,       "AllowRegionSelectionForPMLCreation:=",       &lt;boolean&gt;,       </pre>							

			<pre>"Selections:=" , &lt;string list&gt;, "File Name:=" , &lt;string filepath&gt;, "Major Version:=" , &lt;integer (-1 if not applicable)&gt;, "Minor Version:=" , &lt;integer (-1 if not applicable)&gt;)</pre>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	Export(<Parameters>)
<b>Python Example</b>	<pre>oEditor.Export(     ["NAME:ExportParameters",      "AllowRegionDependentPartSelectionForPMLCreation:=", True,      "AllowRegionSelectionForPMLCreation:=", True,      "Selections:=" , "Box1,Box2,Box3",      "File Name:=" , "C:/Users/jdoe/Desktop/export.sab",      "Major Version:=" , 25,      "Minor Version:=" , 0     ])</pre>

<b>VB Syntax</b>	Export <Parameters>
<b>VB Example</b>	<pre>oEditor.Export</pre>

```

Array("NAME:ExportParameters",
      "AllowRegionDependentPartSelectionForPMLCreation:=", True,
      "AllowRegionSelectionForPMLCreation:=", True,
      "Selections:=" , "Box1,Box2,Box3",
      "File Name:=" , "C:/Users/jdoe/Desktop/export.sab",
      "Major Version:=" , 25,
      "Minor Version:=" , 0)

```

## ExportModelImageToFile

Exports the model as an image file (\*.avz, \*.bmp, \*.gif, \*.jpeg, \*.tiff, \*.wrl). In Release 23.1, this command is fully supports -ng (non-graphical) mode. To export to Ensight use \*.avz. For export to Ensight in -ng mode, the corresponding version of Ensight must be installed. On Linux, it might need manual set environment variable AWP\_ROOT212 to its installation path, e.g. AWP\_ROOT212-2=/installations/ansys\_inc/v212/ for AnsysEDT v21.2 and Ensight 21.2.

ExportModelImageToFile supports export overlay of polar plot 3D with existing transformation (scaling, rotation and translation) in -ng (non-graphical) mode.

UI Access	Modeler > Export.		
Parameters	Name	Type	Description
	<path>	String	Full file path including extension.
	<width>	Integer	Width in pixels (use 0 for default).
	<height>	Integer	Height in pixels (use 0 for default).
	<Parameters>	Array	Structured array.  Array ("NAME:SaveImageParams", "ShowAxis:=" , <string containing

		<pre> boolean&gt;,       "ShowGrid:=" , &lt;string containing boolean&gt;,       "ShowRuler:=" , &lt;string containing boolean&gt;,       "ShowRegion:=" , &lt;string&gt;,       "Selections:=" , &lt;string&gt;,       "FieldPlotSelections:=" , &lt;string&gt;' # Comma delimited string. #Use to set which field plot to show.       "Orientation:=" , &lt;string&gt;       "ShowOrientationGadget:=" , &lt;False&gt; </pre>
<b>Return Value</b>	None.	

<b>Python Syntax</b>	ExportModelImageToFile(<path> <width> <height> <Parameters>)
<b>Python Example</b>	<pre> oEditor.ExportModelImageToFile (     "C:/Users/jdoe/Desktop/export.bmp",     1920,     1080,     [         "NAME:SaveImageParams",         "ShowAxis:=" , "True",         "ShowGrid:=" , "True", </pre>

```
    "ShowRuler:=" , "True",
    "ShowRegion:=" , "Default",
    "Selections:=" , "",
    "FieldPlotSelections:=" , "",
    "FitToSelections:=" , "",
    "FitToFieldPlotSelections:=" , "",
    "Orientation:=" , ""
)
)
```

<b>VB Syntax</b>	ExportModelImageToFile <path> <width> <height> <Parameters>
<b>VB Example</b>	<pre>oEditor.ExportModelImageToFile     "C:/Users/jdoe/Desktop/export.bmp",     1383,     512,     Array ("NAME:SaveImageParams",         "ShowAxis:=" , "True",         "ShowGrid:=" , "True",         "ShowRuler:=" , "True",         "ShowRegion:=" , "Default",         "Selections:=" , ""</pre>

	<pre>"FitToFieldPlotSelections:=", "", "Orientation:=", ""</pre>
--	--

## Fillet

Performs a fillet on specified edge(s).

UI Access	<b>Modeler &gt; Fillet.</b>														
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;SelectionsArray&gt;</code></td> <td>Array</td> <td>Structured array. See: <a href="#">SelectionsArray</a>.</td> </tr> <tr> <td><code>&lt;Parameters&gt;</code></td> <td>Array</td> <td>Structured array.             Array("NAME:Parameters",            &lt;FilletParametersArray&gt;)</td> </tr> <tr> <td><code>&lt;FilletParametersArray&gt;</code></td> <td>Array</td> <td>Structured array.             Array("NAME:FilletParameters",            "Edges:=" , &lt;array containing integer edge IDs&gt;,            "Vertices:=" , &lt;empty array&gt;,            "Radius:=" , &lt;string&gt;,            "Setback:=" , &lt;string&gt;)</td> </tr> </tbody> </table>	Name	Type	Description	<code>&lt;SelectionsArray&gt;</code>	Array	Structured array. See: <a href="#">SelectionsArray</a> .	<code>&lt;Parameters&gt;</code>	Array	Structured array.  Array("NAME:Parameters", <FilletParametersArray>)	<code>&lt;FilletParametersArray&gt;</code>	Array	Structured array.  Array("NAME:FilletParameters", "Edges:=" , <array containing integer edge IDs>, "Vertices:=" , <empty array>, "Radius:=" , <string>, "Setback:=" , <string>)		
Name	Type	Description													
<code>&lt;SelectionsArray&gt;</code>	Array	Structured array. See: <a href="#">SelectionsArray</a> .													
<code>&lt;Parameters&gt;</code>	Array	Structured array.  Array("NAME:Parameters", <FilletParametersArray>)													
<code>&lt;FilletParametersArray&gt;</code>	Array	Structured array.  Array("NAME:FilletParameters", "Edges:=" , <array containing integer edge IDs>, "Vertices:=" , <empty array>, "Radius:=" , <string>, "Setback:=" , <string>)													
<b>Return Value</b>	None.														

<b>Python Syntax</b>	<code>Fillet(&lt;SelectionsArray&gt;, &lt;Parameters&gt;)</code>
<b>Python Example</b>	<pre>oEditor.Fillet(     [ "NAME:Selections",</pre>

```
"Selections:=" , "Box1",
"NewPartsModelFlag:=" , "Model"
],
[ "NAME:Parameters",
[ "NAME:FilletParameters",
"Edges:=" , [13],
"Vertices:=" , [],
"Radius:=" , "1mm",
"Setback:=" , "0mm"
]
])
```

VB Syntax	Fillet <SelectionsArray> <Parameters>
<b>VB Example</b>	<pre>oEditor.Fillet Array("NAME:Selections",       "Selections:=" , "Box1",       "NewPartsModelFlag:=" , "Model") Array("NAME:Parameters",       Array("NAME:FilletParameters",             "Edges:=" , Array(13),</pre>

	<pre>"Vertices:=" , Array(), "Radius:=" , "1mm", "Setback:=" , "0mm") )</pre>
--	---

## FlattenGroup

Flattens a specified history tree group.

UI Access	Modeler > Group > Flatten.		
Parameters	Name	Type	Description
	<GroupID>	Array	Structured array. Array("Groups:=", Array(<string list of group IDs>))
Return Value	None.		

Python Syntax	FlattenGroup (<GroupID>)
Python Example	oEditor.FlattenGroup (["Groups:=", ["Group1"]])

VB Syntax	FlattenGroup <GroupID>
VB Example	oEditor.FlattenGroup Array("Groups:=", Array("Group1"))

## GenerateHistory

Generates the history for specified 1D object(s).

<b>UI Access</b>	<b>Modeler &gt; Generate History.</b>		
<b>Parameters</b>	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	GenerateHistory (<SelectionsArray>)
<b>Python Example</b>	<pre> oEditor.GenerateHistory(     [         "NAME:Selections",         "Selections:=" , "Polyline1",         "NewPartsModelFlag:=" , "Model",         "UseCurrentCS:=" , True     ] ) </pre>

<b>VB Syntax</b>	GenerateHistory <SelectionsArray>
<b>VB Example</b>	<pre> oEditor.GenerateHistory Array("NAME:Selections",     [         "Selections:=" , "Polyline1",         "NewPartsModelFlag:=" , "Model",         "UseCurrentCS:=" , True     ] ) </pre>

## HealObject

Heals an imported object.

UI Access	Modeler > Model Preparation > Heal.											
	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SelectionsArray&gt;</td> <td>Array</td> <td>Structured array. See: <a href="#">SelectionsArray</a>.</td> </tr> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array ("NAME:ObjectHealingParameters",       "Version:=" , &lt;integer&gt;,       "AutoHeal:=" , &lt;boolean&gt;,       "TolerantStitch:=" , &lt;boolean&gt;,       "SimplifyGeom:=" , &lt;boolean&gt;,       "TightenGaps:=" , &lt;boolean&gt;,       "HealToSolid:=" , &lt;boolean&gt;,       "StopAfterFirstStitchError:=", &lt;boolean&gt;,       "MaxStitchTol:=" , &lt;float&gt;,       "ExplodeAndStitch:=" , &lt;boolean&gt;,       "GeomSimplificationTol:=", &lt;integer&gt;,       "MaxGeneratedRadiusForSimplification:=",       &lt;integer&gt;,       "SimplifyType:=" , &lt;integer&gt;,       "TightenGapsWidth:=" , &lt;float&gt;,</pre> </td> </tr> </tbody> </table>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .	<Parameters>	Array	<p>Structured array.</p> <pre>Array ("NAME:ObjectHealingParameters",       "Version:=" , &lt;integer&gt;,       "AutoHeal:=" , &lt;boolean&gt;,       "TolerantStitch:=" , &lt;boolean&gt;,       "SimplifyGeom:=" , &lt;boolean&gt;,       "TightenGaps:=" , &lt;boolean&gt;,       "HealToSolid:=" , &lt;boolean&gt;,       "StopAfterFirstStitchError:=", &lt;boolean&gt;,       "MaxStitchTol:=" , &lt;float&gt;,       "ExplodeAndStitch:=" , &lt;boolean&gt;,       "GeomSimplificationTol:=", &lt;integer&gt;,       "MaxGeneratedRadiusForSimplification:=",       &lt;integer&gt;,       "SimplifyType:=" , &lt;integer&gt;,       "TightenGapsWidth:=" , &lt;float&gt;,</pre>		
Name	Type	Description										
<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .										
<Parameters>	Array	<p>Structured array.</p> <pre>Array ("NAME:ObjectHealingParameters",       "Version:=" , &lt;integer&gt;,       "AutoHeal:=" , &lt;boolean&gt;,       "TolerantStitch:=" , &lt;boolean&gt;,       "SimplifyGeom:=" , &lt;boolean&gt;,       "TightenGaps:=" , &lt;boolean&gt;,       "HealToSolid:=" , &lt;boolean&gt;,       "StopAfterFirstStitchError:=", &lt;boolean&gt;,       "MaxStitchTol:=" , &lt;float&gt;,       "ExplodeAndStitch:=" , &lt;boolean&gt;,       "GeomSimplificationTol:=", &lt;integer&gt;,       "MaxGeneratedRadiusForSimplification:=",       &lt;integer&gt;,       "SimplifyType:=" , &lt;integer&gt;,       "TightenGapsWidth:=" , &lt;float&gt;,</pre>										
Parameters												

			<pre> "RemoveSliverFaces:=" , &lt;boolean&gt;, "RemoveSmallEdges:=" , &lt;boolean&gt;, "RemoveSmallFaces:=" , &lt;boolean&gt;, "SliverFaceTol:=" , &lt;integer&gt;, "SmallEdgeTol:=" , &lt;integer&gt;, "SmallFaceAreaTol:=" , &lt;integer&gt;, "BoundingBoxScaleFactor:=", &lt;integer&gt;, "RemoveHoles:=" , &lt;boolean&gt;, "RemoveChamfers:=" , &lt;boolean&gt;, "RemoveBlends:=" , &lt;boolean&gt;, "HoleRadiusTol:=" , &lt;integer&gt;, "ChamferWidthTol:=" , &lt;integer&gt;, "BlendRadiusTol:=" , &lt;integer&gt;, "AllowableSurfaceAreaChange:=" , &lt;integer&gt;, "AllowableVolumeChange:=" , &lt;integer&gt;) </pre>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	HealObject(<SelectionsArray>, <Parameters>)
<b>Python Example</b>	oEditor.HealObject(

```
[ "NAME:Selections",
    "Selections:=" , "Box1",
    "NewPartsModelFlag:=" , "Model"
],
[ "NAME:ObjectHealingParameters",
    "Version:=" , 1,
    "AutoHeal:=" , True,
    "TolerantStitch:=" , True,
    "SimplifyGeom:=" , True,
    "TightenGaps:=" , True,
    "HealToSolid:=" , False,
    "StopAfterFirstStitchError:=", False,
    "MaxStitchTol:=" , 0.001,
    "ExplodeAndStitch:=" , True,
    "GeomSimplificationTol:=", -1,
    "MaximumGeneratedRadiusForSimplification:=", -1,
    "SimplifyType:=" , 2,
    "TightenGapsWidth:=" , 1E-06,
    "RemoveSliverFaces:=" , False,
    "RemoveSmallEdges:=" , False,
    "RemoveSmallFaces:=" , False,
```

```

    "SliverFaceTol:="      , 0,
    "SmallEdgeTol:="       , 0,
    "SmallFaceAreaTol:="   , 0,
    "BoundingBoxScaleFactor:=", 1250,
    "RemoveHoles:="        , False,
    "RemoveChamfers:="     , False,
    "RemoveBlends:="       , False,
    "HoleRadiusTol:="      , 0,
    "ChamferWidthTol:="    , 0,
    "BlendRadiusTol:="     , 0,
    "AllowableSurfaceAreaChange:=", 5,
    "AllowableVolumeChange:=", 5
  )
)

```

<b>VB Syntax</b>	HealObject <SelectionsArray> <Parameters>
<b>VB Example</b>	<pre> oEditor.HealObject Array("NAME:Selections", "Selections:=", "Box2", "NewPartsModelFlag:=", _ "Model"), Array("NAME:ObjectHealingParameters", "Version:=", 1, "AutoHeal:=", false, "TolerantStitch:=", _ true, "SimplifyGeom:=", true, "TightenGaps:=", true, "HealToSolid:=", false, "StopAfter- FirstStitchError:=", _ </pre>

```

false, "MaxStitchTol:=", 0.001, "ExplodeAndStitch:=", true, "GeomSimplificationTol:=",
-
-1, "MaximumGeneratedRadiusForSimplification:=", -1, "SimplifyType:=", 2,
"TightenGapsWidth:=", _
1E-06, "RemoveSliverFaces:=", false, "RemoveSmallEdges:=", false, "RemoveSmallFaces:=",
-
false, "SliverFaceTol:=", 0, "SmallEdgeTol:=", 0, "SmallFaceAreaTol:=", 0, "Bound-
ingBoxScaleFactor:=", _
1250, "RemoveHoles:=", false, "RemoveChamfers:=", false, "RemoveBlends:=", _
false, "HoleRadiusTol:=", 0, "ChamferWidthTol:=", 0, "BlendRadiusTol:=", 0, "Allow-
ableSurfaceAreaChange:=", _
5, "AllowableVolumeChange:=", 5)

```

## GetActiveCoordinateSystem

Returns the active coordinate system.

<b>UI Access</b>	None.
<b>Parameters</b>	None.
<b>Return Value</b>	String name of active coordinate system.

<b>Python Syntax</b>	GetActiveCoordinateSystem()
<b>Python Example</b>	<code>oEditor.GetActiveCoordinateSystem()</code>

<b>VB Syntax</b>	GetActiveCoordinateSystem
------------------	---------------------------

**VB Example**

```
dim csName  
csName = oEditor.GetActiveCoordinateSystem
```

## GetCoordinateSystems

Returns the names of coordinate systems in the design.

<b>UI Access</b>	None.
<b>Parameters</b>	None.
<b>Return Value</b>	Array containing string names of coordinate systems.

**Python Syntax**

```
GetCoordinateSystems()
```

**Python Example**

```
oEditor.GetCoordinateSystems ()
```

**VB Syntax**

```
GetCoordinateSystems()
```

**VB Example**

```
dim csNames  
csNames = oEditor.GetCoordinateSystems
```

## Import

Imports a 3D model file.

**Note:**

This script does not import DXF or GDSII models. See: [ImportDXF](#) and [ImportGDSII](#).

UI Access	Modeler > Import.		
	Name	Type	Description
<b>Parameters</b>	<Parameters>	Array	<p>Structured array.</p> <pre>Array("NAME:NativeBodyParameters",       "HealOption:=" , &lt;integer&gt;,       "Options:=" , &lt;string&gt;,       "FileType:=" , &lt;string "UnRecognized"&gt;,       "MaxStitchTol:=" , &lt;integer&gt;,       "ImportFreeSurfaces:=" , &lt;boolean&gt;,       "GroupByAssembly:=" , &lt;boolean&gt;,       "CreateGroup:=" , &lt;boolean&gt;,       "STLFileUnit:=" , &lt;string&gt;,       "MergeFacesAngle:=" , &lt;float&gt;,       "HealSTL:=" , &lt;boolean&gt;,       "ReduceSTL:=" , &lt;boolean&gt;,       "ReduceMaxError:=" , &lt;integer&gt;,       "ReducePercentage:=" , &lt;integer&gt;,       "PointCoincidenceTol:=" , &lt;float&gt;,       "CreateLightweightPart:=" , &lt;boolean&gt;,       "ImportMaterialNames:=" , &lt;boolean&gt;,       "SeparateDisjointLumps:=" , &lt;boolean&gt;,</pre>

		"SourceFile:=" , <string>)
<b>Return Value</b>	None.	

<b>Python Syntax</b>	Import(<Parameters>)
<b>Python Example</b>	<pre> oEditor.Import(     ["NAME:NativeBodyParameters",         "HealOption:=" , 0,         "Options:=" , "-1",         "FileType:=" , "UnRecognized",         "MaxStitchTol:=" , -1,         "ImportFreeSurfaces:=" , False,         "GroupByAssembly:=" , False,         "CreateGroup:=" , True,         "STLFileUnit:=" , "Auto",         "MergeFacesAngle:=" , 0.02,         "HealSTL:=" , False,         "ReduceSTL:=" , False,         "ReduceMaxError:=" , 0,         "ReducePercentage:=" , 100,         "PointCoincidenceTol:=" , 1E-06,     ] ) </pre>

```

    "CreateLightweightPart:=", False,
    "ImportMaterialNames:=", False,
    "SeparateDisjointLumps:=", False,
    "SourceFile:="           , "C:\\\\Users\\\\jdoe\\\\Desktop\\\\export.model"
)

```

VB Syntax	Import <Parameters>
VB Example	<pre> oEditor.Import Array("NAME:NativeBodyParameters",                     "HealOption:="          , 0,                     "Options:="             , "-1",                     "FileType:="            , "UnRecognized",                     "MaxStitchTol:="        , -1,                     "ImportFreeSurfaces:="   , False,                     "GroupByAssembly:="     , False,                     "CreateGroup:="          , True,                     "STLFileUnit:="          , "Auto",                     "MergeFacesAngle:="      , 0.02,                     "HealSTL:="              , False,                     "ReduceSTL:="             , False,                     "ReduceMaxError:="        , 0,                     "ReducePercentage:="      , 100, </pre>

```

    "PointCoincidenceTol:=" , 1E-06,
    "CreateLightweightPart:=", False,
    "ImportMaterialNames:=" , False,
    "SeparateDisjointLumps:=", False,
    "SourceFile:=" , "C:\\Users\\jdoe\\Desktop\\export.model")

```

## ImportDXF [Modeler]

Imports a DXF model file.

UI Access	Modeler > Import.								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>&lt;Parameters&gt;</i></td> <td>Array</td> <td> <p>Structured array.</p> <pre>         Array("NAME:options",               "FileName:=" , &lt;string&gt;,               "Scale:=" , &lt;float&gt;,               "AutoDetectClosed:=" , &lt;boolean&gt;,               "SelfStitch:=" , &lt;boolean&gt;,               "DefeatureGeometry:=" , &lt;boolean&gt;,               "DefeatureDistance:=" , &lt;integer&gt;,               "RoundCoordinates:=" , &lt;boolean&gt;,               "RoundNumDigits:=" , &lt;integer&gt;,               "WritePolyWithWidthAsFilledPoly:=" , &lt;boolean&gt;, </pre> </td> </tr> </tbody> </table>	Name	Type	Description	<i>&lt;Parameters&gt;</i>	Array	<p>Structured array.</p> <pre>         Array("NAME:options",               "FileName:=" , &lt;string&gt;,               "Scale:=" , &lt;float&gt;,               "AutoDetectClosed:=" , &lt;boolean&gt;,               "SelfStitch:=" , &lt;boolean&gt;,               "DefeatureGeometry:=" , &lt;boolean&gt;,               "DefeatureDistance:=" , &lt;integer&gt;,               "RoundCoordinates:=" , &lt;boolean&gt;,               "RoundNumDigits:=" , &lt;integer&gt;,               "WritePolyWithWidthAsFilledPoly:=" , &lt;boolean&gt;, </pre>		
Name	Type	Description							
<i>&lt;Parameters&gt;</i>	Array	<p>Structured array.</p> <pre>         Array("NAME:options",               "FileName:=" , &lt;string&gt;,               "Scale:=" , &lt;float&gt;,               "AutoDetectClosed:=" , &lt;boolean&gt;,               "SelfStitch:=" , &lt;boolean&gt;,               "DefeatureGeometry:=" , &lt;boolean&gt;,               "DefeatureDistance:=" , &lt;integer&gt;,               "RoundCoordinates:=" , &lt;boolean&gt;,               "RoundNumDigits:=" , &lt;integer&gt;,               "WritePolyWithWidthAsFilledPoly:=" , &lt;boolean&gt;, </pre>							

			<pre>"ImportMethod:=" , &lt;integer&gt;, "2DSheetBodies:=" , &lt;boolean&gt;, &lt;layersArray&gt;)</pre>
	<b>&lt;layersArray&gt;</b>	Array	<p>Structured array.</p> <pre>Array("NAME:LayerInfo",       &lt;layer&gt;, &lt;layer&gt;, &lt;layer&gt;, ...)</pre>
	<b>&lt;layer&gt;</b>	Array	<p>Structured array.</p> <pre>Array("NAME:&lt;layerName&gt;",       "source:=" , &lt;string&gt;,       "display_source:=" , &lt;string&gt;,       "import:=" , &lt;boolean&gt;,       "dest:=" , &lt;string&gt;,       "dest_selected:=" , &lt;boolean&gt;,       "layer_type:=" , &lt;string&gt;)</pre>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ImportDXF(<Parameters>)
<b>Python Example</b>	<pre>oEditor.ImportDXF(     ["NAME:options",      "FileName:=", "C:/Users/jdoe/Desktop/export.dxf",      "Scale:=" , 0.001,      "AutoDetectClosed:=" , True,</pre>

```
"SelfStitch:=" , True,
"DefeatureGeometry:=" , False,
"DefeatureDistance:=" , 0,
"RoundCoordinates:=" , False,
"RoundNumDigits:=" , 4,
"WritePolyWithWidthAsFilledPoly:=", False,
"ImportMethod:=" , 1,
"2DSheetBodies:=" , False,
[ "NAME:LayerInfo",
  [ "NAME:0",
    "source:=" , "0",
    "display_source:=" , "0",
    "import:=" , True,
    "dest:=" , "0",
    "dest_selected:=" , False,
    "layer_type:=" , "signal"
  ],
  [ "NAME:LAYER_1",
    "source:=" , "LAYER_1",
    "display_source:=" , "LAYER_1",
    "import:=" , True,
    "dest:=" , "LAYER_1",
    "dest_selected:=" , False,
    "layer_type:=" , "signal"
  ]
],
```

```
"import:=" , True,
"dest:=" , "LAYER_1",
"dest_selected:=" , False,
"layer_type:=" , "signal"
] ,
["NAME:LAYER_2",
"source:=" , "LAYER_2",
"display_source:=" , "LAYER_2",
"import:=" , True,
"dest:=" , "LAYER_2",
"dest_selected:=" , False,
"layer_type:=" , "signal"
]
])
```

<b>VB Syntax</b>	ImportDXF <Parameters>
<b>VB Example</b>	<pre> oEditor.ImportDXF Array("NAME:options", "FileName:=", _ "C:/Users/jdoe/Desktop/export.dxf", "Scale:=", 0.001, "AutoDetectClosed:=", _ true, "SelfStitch:=", true, "DefeatureGeometry:=", false, "DefeatureDistance:=", _ 0, "RoundCoordinates:=", false, "RoundNumDigits:=", 4, </pre>

```

    "WritePolyWithWidthAsFilledPoly:=", __
    false, "ImportMethod:=", 1, "2DSheetBodies:=", false, Array("NAME:LayerInfo", Array
    ("NAME:0", "source:=", __
    "0", "display_source:=", "0", "import:=", true, "dest:=", "0", "dest_selected:=", __
    false, "layer_type:=", "signal"), Array("NAME:LAYER_1", "source:=", "LAYER_1", "dis-
    play_source:=", __
    "LAYER_1", "import:=", true, "dest:=", "LAYER_1", "dest_selected:=", false, "layer_
    type:=", __
    "signal"), Array("NAME:LAYER_2", "source:=", "LAYER_2", "display_source:=", "LAYER_2",
    "import:=", __
    true, "dest:=", "LAYER_2", "dest_selected:=", false, "layer_type:=", "signal")))
)

```

## ImportGDSII [Modeler]

Imports a GDSII model file.

UI Access	Modeler > Import.								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <p>Array("NAME:options",</p> <p>"FileName:=" , &lt;string&gt;,</p> <p>"FlattenHierarchy:=" , &lt;boolean&gt;,</p> <p>"ImportMethod:=" , &lt;integer&gt;,</p> <p>&lt;layerMapArray&gt;, &lt;layerMapArray&gt;, &lt;layerMapArray&gt;,...</p> </td></tr> </tbody> </table>	Name	Type	Description	<Parameters>	Array	<p>Structured array.</p> <p>Array("NAME:options",</p> <p>"FileName:=" , &lt;string&gt;,</p> <p>"FlattenHierarchy:=" , &lt;boolean&gt;,</p> <p>"ImportMethod:=" , &lt;integer&gt;,</p> <p>&lt;layerMapArray&gt;, &lt;layerMapArray&gt;, &lt;layerMapArray&gt;,...</p>		
Name	Type	Description							
<Parameters>	Array	<p>Structured array.</p> <p>Array("NAME:options",</p> <p>"FileName:=" , &lt;string&gt;,</p> <p>"FlattenHierarchy:=" , &lt;boolean&gt;,</p> <p>"ImportMethod:=" , &lt;integer&gt;,</p> <p>&lt;layerMapArray&gt;, &lt;layerMapArray&gt;, &lt;layerMapArray&gt;,...</p>							

		<pre>"OrderMap:=" , [ "entry:=" , &lt;entry&gt;, &lt;entry&gt;, &lt;entry&gt;, ] ])</pre>
	<b>&lt;layerMapArray&gt;</b>	<p>Structured array.</p> <pre>Array("NAME:LayerMapInfo", "LayerNum:=" , &lt;integer&gt;, "DestLayer:=" , &lt;string&gt;, "layer_type:=" , &lt;string&gt;)</pre>
	<b>&lt;entry&gt;</b>	<p>Structured array.</p> <pre>Array("order:=" , &lt;integer LayerNum&gt;, "layer:=" , &lt;string DestLayer&gt;)</pre>
<b>Return Value</b>	None.	

<b>Python Syntax</b>	<pre>ImportGDSII(&lt;Parameters&gt;)</pre>
<b>Python Example</b>	<pre>oEditor.ImportGDSII( ["NAME:options", "FileName:=", "C:/Users/coil2.gds", "FlattenHierarchy:=" , True, "ImportMethod:=" , 1, ["NAME:LayerMap",</pre>

```
[ "NAME:LayerMapInfo",
  "LayerNum:=" , 12,
  "DestLayer:=" , "Signal12",
  "layer_type:=" , "signal"
] ,
[ "NAME:LayerMapInfo",
  "LayerNum:=" , 13,
  "DestLayer:=" , "Signal13",
  "layer_type:=" , "signal"
] ,
[ "NAME:LayerMapInfo",
  "LayerNum:=" , 14,
  "DestLayer:=" , "Signal14",
  "layer_type:=" , "signal"
]
] ,
"OrderMap:=" ,
[
  "entry:=" ,
    [ "order:=", 0, "layer:=", "Signal12"] ,
```

```

    "entry:=",
    ["order:=", 1, "layer:=", "Signal13"],
    "entry:=",
    ["order:=", 2, "layer:=", "Signal14"]
]
])

```

<b>VB Syntax</b>	ImportGDSII <Parameters>
<b>VB Example</b>	<pre> oEditor.ImportGDSII Array("NAME:options",     "FileName:=", "C:/export.gds",     "FlattenHierarchy:=", True,     "ImportMethod:=", 1,     Array("NAME:LayerMap",         Array("NAME:LayerMapInfo",             "LayerNum:=", 1,             "DestLayer:=", "Signal1",             "layer_type:=", "signal"))) "OrderMap:=", Array(     "entry:=", Array(         "order:=", 0,         "layer:=", "Signal1"))) </pre>

## Intersect

Intersects specified objects.

UI Access	Modeler > Boolean > Intersect.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
Return Value	None.		

Python Syntax	Intersect(<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.Intersect(     [         "NAME:Selections",         "Selections:=", "Rectangle1,Rectangle2"     ],     [         "NAME:IntersectParameters",         "KeepOriginals:=", False     ] )</pre>

<b>VB Syntax</b>	Intersect <SelectionsArray> <Parameters>
<b>VB Example</b>	<pre> oEditor.Intersect  Array("NAME:Selections",       "Selections:=", "Rectangle1,Rectangle2")  Array("NAME:IntersectParameters",       "KeepOriginals:=", False) </pre>

## MoveCStoEnd

Moves a specified Object Coordinate System to the end of the History tree.

<b>UI Access</b>	<b>Modeler &gt; Coordinate System &gt; Move CS to End.</b>		
<b>Parameters</b>	Name <SelectionsArray>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	MoveCStoEnd (<SelectionsArray>)
<b>Python Example</b>	<pre> oEditor.MoveCSToEnd (   [ "NAME:Selections",     "Selections:=" , "ObjectCS1"   ] ) </pre>

<b>VB Syntax</b>	MoveCSToEnd <SelectionsArray>
<b>VB Example</b>	<code>oEditor.MoveCSToEnd Array("NAME:Selections", "Selections:=", "ObjectCS1")</code>

## MoveEntityToGroup

Moves a specified entity or entities to a specified group.

<b>UI Access</b>	Drag item into the group in the history tree.		
<b>Parameters</b>	<b>Name</b> <code>&lt;Objects&gt;</code>	<b>Type</b> Array	<b>Description</b> Structured array. Selects the entity/entities to move.  <code>Array("Objects:=" , &lt;array containing string object IDs&gt;)</code>
	<code>&lt;MoveEntityToGroup&gt;</code>	Array	Structured array.  <code>Array("ParentGroup:=" , &lt;string group name&gt;)</code>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	MoveEntityToGroup (<Objects>, <MoveEntityToGroup>)
<b>Python Example</b>	<code>oEditor.MoveEntityToGroup (</code> <code>["Objects:=", ["Box3"]],</code> <code>["ParentGroup:=", "Box_Group"])</code>

<b>VB Syntax</b>	MoveEntityToGroup <Objects> <MoveEntityToGroup>
------------------	---

<b>VB Example</b>	<pre> oEditor.MoveEntityToGroup Array("Objects:=", Array("Box3")) Array("ParentGroup:=", "Box_Group") </pre>
-------------------	--

## MoveFaces

Moves the specified faces along normal or along a vector.

UI Access	<b>Modeler &gt; Surface &gt; Move Faces &gt; Along [Normal/Vector].</b>														
Parameters	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;SelectionsArray&gt;</td> <td>Array</td> <td>Structured array. See: <a href="#">SelectionsArray</a>.</td> </tr> <tr> <td>&lt;MoveFacesParametersArray&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre> Array ("NAME:Parameters",       &lt;FacesOfOneObjToMove&gt;, &lt;FacesOfOneObjToMove&gt;, ...) </pre> </td> </tr> <tr> <td>&lt;FacesOfOneObjToMove&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre> Array ("Name:MoveFacesParameters",       "MoveAlongNormalFlag:=", &lt;boolean&gt;,       "OffsetDistance:=", &lt;string&gt;,       "MoveVectorX:=", &lt;string&gt;,       "MoveVectorY:=", &lt;string&gt;,       "MoveVectorZ:=", &lt;string&gt;,       "FacesToMove:=", &lt;array&gt;) </pre> <p>MoveAlongNormalFlag specifies whether to move along the face normal or along a vector. If false, provide the MoveVectorX, MoveVectorY, and MoveVectorZ parameters.</p> </td> </tr> </table>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .	<MoveFacesParametersArray>	Array	<p>Structured array.</p> <pre> Array ("NAME:Parameters",       &lt;FacesOfOneObjToMove&gt;, &lt;FacesOfOneObjToMove&gt;, ...) </pre>	<FacesOfOneObjToMove>	Array	<p>Structured array.</p> <pre> Array ("Name:MoveFacesParameters",       "MoveAlongNormalFlag:=", &lt;boolean&gt;,       "OffsetDistance:=", &lt;string&gt;,       "MoveVectorX:=", &lt;string&gt;,       "MoveVectorY:=", &lt;string&gt;,       "MoveVectorZ:=", &lt;string&gt;,       "FacesToMove:=", &lt;array&gt;) </pre> <p>MoveAlongNormalFlag specifies whether to move along the face normal or along a vector. If false, provide the MoveVectorX, MoveVectorY, and MoveVectorZ parameters.</p>		
Name	Type	Description													
<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .													
<MoveFacesParametersArray>	Array	<p>Structured array.</p> <pre> Array ("NAME:Parameters",       &lt;FacesOfOneObjToMove&gt;, &lt;FacesOfOneObjToMove&gt;, ...) </pre>													
<FacesOfOneObjToMove>	Array	<p>Structured array.</p> <pre> Array ("Name:MoveFacesParameters",       "MoveAlongNormalFlag:=", &lt;boolean&gt;,       "OffsetDistance:=", &lt;string&gt;,       "MoveVectorX:=", &lt;string&gt;,       "MoveVectorY:=", &lt;string&gt;,       "MoveVectorZ:=", &lt;string&gt;,       "FacesToMove:=", &lt;array&gt;) </pre> <p>MoveAlongNormalFlag specifies whether to move along the face normal or along a vector. If false, provide the MoveVectorX, MoveVectorY, and MoveVectorZ parameters.</p>													

			FacesToMove is an array of integers (the IDs of the faces to move).
<b>Return Value</b>	None		

<b>Python Syntax</b>	MoveFaces (<SelectionsArray>, <MoveFacesParametersArray>)
<b>Python Example</b>	<pre> oEditor.MoveFaces(     [         "NAME:Selections",         "Selections:="          , "Rectangle1",         "NewPartsModelFlag:="   , "Model"     ],     [         "NAME:Parameters",         [             "NAME:MoveFacesParameters",             "MoveAlongNormalFlag:="   , True,             "OffsetDistance:="       , "1mm",             "MoveVectorX:="          , "0mm",             "MoveVectorY:="          , "0mm",             "MoveVectorZ:="          , "0mm",             "FacesToMove:="          , [183]         ]     ] ) </pre>

<b>VB Syntax</b>	MoveFaces <SelectionsArray>, <MoveFacesParametersArray>
<b>VB Example</b>	<pre> oEditor.MoveFaces _</pre>

```

Array("NAME:Selections", "Selections:=", _
      "Box2,Box1"), _

Array("NAME:Parameters", _

Array("NAME:MoveFacesParameters", _
      "MoveAlongNormalFlag:=", true, _
      "OffsetDistance:=", "1mm", _
      "FacesToMove:=", Array(218)),

Array("NAME:MoveFacesParameters", _
      "MoveAlongNormalFlag:=", false,_
      "OffsetDistance:=", "1mm", _
      "MoveVectorX:=", "1mm", _
      "MoveVectorY:=", "0mm", _
      "MoveVectorZ:=", "0mm", _
      "FacesToMove:=", Array(185)))

```

## ProjectSheet

Project a sheet object, typically for modeling thin conformal deposits. Typically followed by **Thicken Sheet**.

<b>UI Access</b>	Click <b>Modeler &gt; Surface &gt; Project Sheet</b> .		
<b>Parameters</b>	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .

<b>Return Value</b>	None.
---------------------	-------

<b>Python Syntax</b>	ProjectSheet (<SelectionsArray>, <Parameters>)
<b>Python Example</b>	<pre> oEditor.ProjectSheet (     [ 'NAME:Selections',         'Selections:=' , 'Box1,Box2,Polyline1' ],     [ 'NAME:ProjectSheetParameters' ] ) </pre>

<b>VB Syntax</b>	ProjectSheet <SelectionsArray> <Parameters>
<b>VB Example</b>	<pre> oEditor.ProjectSheet Array("NAME:Selections",       "Selections:=", "Box1,Box2,Polyline1"), Array("NAME:ProjectSheetParameters") </pre>

## PurgeHistory

Purges the history of a specified object.

<b>UI Access</b>	<b>Modeler &gt; Purge History.</b>		
<b>Parameters</b>	Name	Type	Description

	<code>&lt;SelectionsArray&gt;</code>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>PurgeHistory (&lt;SelectionsArray&gt;)</code>
<b>Python Example</b>	<pre> oEditor.PurgeHistory(     ["NAME:Selections",         "Selections:=", "Box2",         "NewPartsModelFlag:=", "Model"     ] ) </pre>

<b>VB Syntax</b>	<code>PurgeHistory &lt;SelectionsArray&gt;</code>
<b>VB Example</b>	<pre> oEditor.PurgeHistory     Array("NAME:Selections",         "Selections:=", "Box2",         "NewPartsModelFlag:=", "Model") ) </pre>

## Section

Creates a 2D cross-section of the selection in the specified plane.

<b>UI Access</b>	<b>Modeler &gt; Surface &gt; Section.</b>		
<b>Parameters</b>	Name	Type	Description

	<code>&lt;SelectionsArray&gt;</code>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
	<code>&lt;Parameters&gt;</code>	Array	Structured array.  <code>Array("NAME:SectionToParameters",            "CreateNewObjects:=" , &lt;boolean&gt;,            "SectionPlane:=" , &lt;string "XY", "YZ", or            "ZX"&gt;,            "SectionCrossObject:=" , &lt;boolean&gt;)</code>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>Section (&lt;SelectionsArray&gt;, &lt;Parameters&gt;)</code>
<b>Python Example</b>	<pre> oEditor.Section(     ["NAME:Selections",         "Selections:="           , "Cone1",         "NewPartsModelFlag:="     , "Model"     ],     ["NAME:SectionToParameters",         "CreateNewObjects:="      , True,         "SectionPlane:="          , "XY",         "SectionCrossObject:="    , False     ] ) </pre>

<b>VB Syntax</b>	Section <SelectionsArray> <Parameters>
<b>VB Example</b>	<pre> oEditor.Section  Array("NAME:Selections",       "Selections:=" , "Cone1",       "NewPartsModelFlag:=" , "Model")  Array("NAME:SectionToParameters",       "CreateNewObjects:=" , True,       "SectionPlane:=" , "XY",       "SectionCrossObject:=" , False) </pre>

## SeparateBody

Separates the bodies of specified multi-lump objects.

<b>UI Access</b>	<b>Modeler &gt; Boolean &gt; Separate Bodies.</b>		
<b>Parameters</b>	Name <SelectionsArray>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	SeparateBody (<SelectionsArray>)
<b>Python Example</b>	<pre> oEditor.SeparateBody (     ["NAME:Selections", </pre>

```
"Selections:=", "Rectangle1,Circle1",
"NewPartsModelFlag:=", "Model"
])
```

<b>VB Syntax</b>	SeparateBody <SelectionsArray>
<b>VB Example</b>	<pre>oEditor.SeparateBody Array("NAME:Selections",       "Selections:=", "Rectangle1,Circle1",       "NewPartsModelFlag:=", "Model")</pre>

## SetModelUnits

Sets the model units.

UI Access	Modeler > Units.		
Parameters	Name <Parameters>	Type Array	Description Structured array.  Array("NAME:Units Parameter", "Units:=", <string>, "Rescale:=", <boolean True to rescale model; else False>)  To see valid unit strings, select Modeler > Units.

<b>Return Value</b>	None.
---------------------	-------

<b>Python Syntax</b>	SetModelUnits (<Parameters>)
<b>Python Example</b>	<pre>oEditor.SetModelUnits(     ["NAME:Units Parameter",         "Units:=", "km",         "Rescale:=", False     ] )</pre>

<b>VB Syntax</b>	SetModelUnits <Parameters>
<b>VB Example</b>	<pre>oEditor.SetModelUnits Array("NAME:Units Parameter",     "Units:=", "km",     "Rescale:=", False)</pre>

## SetWCS

Sets the working coordinate system.

<b>UI Access</b>	<b>Modeler &gt; Coordinate System &gt; Set Working CS.</b>								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;Parameters&gt;</td> <td>Array</td> <td>Structured array. Array ("NAME:SetWCS Parameter",</td> </tr> </tbody> </table>			Name	Type	Description	<Parameters>	Array	Structured array. Array ("NAME:SetWCS Parameter",
Name	Type	Description							
<Parameters>	Array	Structured array. Array ("NAME:SetWCS Parameter",							

			"Working Coordinate System:=", <string CS name>, "RegionDepCSOk:=", <boolean True if region-dependent; else False)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	SetWCS (<Parameters>)
<b>Python Example</b>	<pre>oEditor.SetWCS(     ["NAME:SetWCS Parameter",      "Working Coordinate System:=", "Global",      "RegionDepCSOk:=", False ])</pre>

<b>VB Syntax</b>	SetWCS <Parameters>
<b>VB Example</b>	<pre>oEditor.SetWCS Array("NAME:SetWCS Parameter",       "Working Coordinate System:=", "Global",       "RegionDepCSOk:=", False)</pre>

## ShowWindow

Opens the active 3D Modeler window.

<b>UI Access</b>	None.
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	ShowWindow
<b>Python Example</b>	<code>oEditor.ShowWindow()</code>

<b>VB Syntax</b>	ShowWindow
<b>VB Example</b>	<code>oEditor.ShowWindow</code>

## Split

Splits the specified object(s) along a plane.

<b>UI Access</b>	<b>Modeler &gt; Boolean &gt; Split.</b>		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;SelectionsArray&gt;</code>	Array	Structured array. See: <a href="#">SelectionsArray</a> .

<b>Parameters</b>	<table border="1"> <tr> <td><code>&lt;Parameters&gt;</code></td><td>Array</td><td>           Structured array.    <code>Array("NAME:SplitToParameters",</code>  <code>      "SplitPlane:=" , &lt;string "XY", "YZ", "ZX", or</code>  <code>      "Dummy"&gt;,</code>  <code>      "WhichSide:=" , &lt;string "PositiveOnly", "Neg-</code>  <code>      "ativeOnly" or "Both"&gt;,</code> </td></tr> </table>	<code>&lt;Parameters&gt;</code>	Array	Structured array.  <code>Array("NAME:SplitToParameters",</code> <code>      "SplitPlane:=" , &lt;string "XY", "YZ", "ZX", or</code> <code>      "Dummy"&gt;,</code> <code>      "WhichSide:=" , &lt;string "PositiveOnly", "Neg-</code> <code>      "ativeOnly" or "Both"&gt;,</code>
<code>&lt;Parameters&gt;</code>	Array	Structured array.  <code>Array("NAME:SplitToParameters",</code> <code>      "SplitPlane:=" , &lt;string "XY", "YZ", "ZX", or</code> <code>      "Dummy"&gt;,</code> <code>      "WhichSide:=" , &lt;string "PositiveOnly", "Neg-</code> <code>      "ativeOnly" or "Both"&gt;,</code>		

			<pre>"ToolType:=" , &lt;string "PlaneTool" or "FaceTool"&gt;, "ToolEntityID:=" , &lt;-1 if using SplitPlane; else FaceID&gt;, "SplitCrossingObjectsOnly:=" , &lt;boolean&gt;, "DeleteInvalidObjects:=" , &lt;boolean&gt;)</pre> <p>You can split an object either along an existing plane , or by creating a plane using an edge.</p> <p>For existing plane:</p> <ul style="list-style-type: none"> <li>• SplitPlane is "XY", "YZ", or "ZX"</li> <li>• ToolType is "PlaneTool"</li> <li>• ToolEntityID is -1</li> </ul> <p>Possible values for 2D XY designs are "YZ", "ZX".</p> <p>Possible values for 2D RZ designs are "XY", "YZ".</p> <p>For an edge:</p> <ul style="list-style-type: none"> <li>• SplitPlane is "Dummy"</li> <li>• ToolType is "EdgeTool"</li> <li>• ToolEntityID is the face ID</li> </ul>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	Split(<SelectionsArray>, <Parameters>)
----------------------	--

<b>Python Example</b> <pre> oEditor.Split(     ["NAME:Selections",         "Selections:="          , "Box1",         "NewPartsModelFlag:="    , "Model"     ],     ["NAME:SplitToParameters",         "SplitPlane:="          , "XY",         "WhichSide:="           , "PositiveOnly",         "ToolType:="            , "PlaneTool",         "ToolEntityID:="         , -1,         "SplitCrossingObjectsOnly:=", False,         "DeleteInvalidObjects:=", True     ] ) </pre>
--

<b>VB Syntax</b> <pre>Split &lt;SelectionsArray&gt; &lt;Parameters&gt;</pre>
<b>VB Example</b> <pre> oEditor.Split     Array("NAME:Selections",         "Selections:="          , "Box1",         "NewPartsModelFlag:="    , "Model")     Array("NAME:SplitToParameters",         "SplitPlane:="          , "XY", </pre>

```

    "WhichSide:="           , "PositiveOnly",
    "ToolType:="            , "PlaneTool",
    "ToolEntityID:="         , -1,
    "SplitCrossingObjectsOnly:=", False,
    "DeleteInvalidObjects:=", True)

```

## Subtract

Subtracts the specified object(s).

UI Access	<b>Modeler &gt; Boolean &gt; Subtract.</b>		
<b>Parameters</b>	Name <i>&lt;Selections&gt;</i>	Type Array	Description <p>Structured array.</p> <p>Array ("NAME:Selections",          "Blank Parts:=" , &lt;string object name(s)&gt;,          "Tool Parts:=" , &lt;string object name(s)&gt;)</p> <p>The Tool Parts object is the object being removed.</p> <p>The Blank Parts object has any Tool Parts overlap removed.</p> <p>Either string can contain more than one object.</p>
	<i>&lt;Parameters&gt;</i>	Array	Structured array. <p>Array ("NAME:SubtractParameters",          "KeepOriginals:=" , &lt;boolean&gt;)</p>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	Subtract(<Selections>, <Parameters>)
<b>Python Example</b>	<pre> oEditor.Subtract(     [         "NAME:Selections",         "Blank Parts:=", "Rectangle1",         "Tool Parts:=", "Rectangle2"     ],     [         "NAME:SubtractParameters",         "KeepOriginals:=", False     ] ) </pre>

<b>VB Syntax</b>	Subtract <Selections> <Parameters>
<b>VB Example</b>	<pre> oEditor.Subtract     Array("NAME:Selections",         "Blank Parts:=", "Rectangle1",         "Tool Parts:=", "Rectangle2")     Array("NAME:SubtractParameters",         "KeepOriginals:=", false) ) </pre>

## SweepFacesAlongNormal

Sweep the specified face(s) along normal.

<b>UI Access</b>	<b>Modeler &gt; Surface &gt; Sweep Faces Along Normal</b>
------------------	---

<b>Parameters</b>	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
	<parameters>	Array	Structured array. Array ("NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", <faceIDarray>, "LengthOfSweep:=", "<value><units>")
<b>Return Value</b>	None		

<b>Python Syntax</b>	SweepFacesAlongNormal(<SelectionsArray> <parameters>)
<b>Python Example</b>	<pre> oEditor.SweepFacesAlongNormal (     ["NAME:Selections",         "Selections:=", "Rectangle1",         "NewPartsModelFlag:=", "Model"],     ["NAME:Parameters",         "NAME:SweepFaceAlongNormalToParameters",         "FacesToDetach:=", [183],         "LengthOfSweep:=", "0.1mm"]) </pre>

<b>VB Syntax</b>	SweepFacesAlongNormal <SelectionsArray> <parameters>
<b>VB Example</b>	<pre> oEditor.SweepFacesAlongNormal      Array ("NAME:Selections",         "Selections:=", "Rectangle1",         "NewPartsModelFlag:=", "Model"),     Array ("NAME:Parameters",         "NAME:SweepFaceAlongNormalToParameters",         "FacesToDetach:=", Array(57),         "LengthOfSweep:=", "0.5mm") ) </pre>

## ThickenSheet

Thickens a sheet object to convert it to a 3D object.

<b>UI Access</b>	<b>Modeler &gt; Surface &gt; Thicken Sheet</b>		
<b>Parameters</b>	Name <SelectionsArray> <parameters>	Type Array Array	Description Structured array. See: <a href="#">SelectionsArray</a> . Structured array. Array ("NAME:SheetThickenParameters", "Thickness:=" , <string>, "BothSides:=" , <boolean>)
<b>Return Value</b>	None.		

---

<b>Python Syntax</b>	ThickenSheet (<SelectionsArray> <parameters>)
<b>Python Example</b>	<pre> oEditor.ThickenSheet(     [         "NAME:Selections",         "Selections:=" , "Rectangle3",         "NewPartsModelFlag:=" , "Model"     ],     [         "NAME:SheetThickenParameters",         "Thickness:=" , "0.01mm",         "BothSides:=" , False     ] ) </pre>

<b>VB Syntax</b>	ThickenSheet <SelectionsArray> <parameters>
<b>VB Example</b>	<pre> oEditor.ThickenSheet     Array("NAME:Selections",         "Selections:=" , "Rectangle3",         "NewPartsModelFlag:=" , "Model")     Array("NAME:SheetThickenParameters",         "Thickness:=" , "0.01mm",         "BothSides:=" , false) ) </pre>

## UncoverFaces

Uncovers the specified face(s).

UI Access	Modeler > Surface > Uncover Faces		
Parameters	Name <SelectionsArray> <parameters>	Type Array Array	Description Structured array. See: <a href="#">SelectionsArray</a> . Structured array. Array ("NAME:Parameters", Array ("NAME:UncoverFacesParameters", "FacesToUncover:=" , <array of face IDs>) )
Return Value	None.		

Python Syntax	UncoverFaces(<SelectionsArray> <parameters>)
Python Example	<pre> oEditor.UncoverFaces (     [ "NAME:Selections",         "Selections:="           , "Box1",         "NewPartsModelFlag:="     , "Model"     ],     [ "NAME:Parameters",         [ "NAME:UncoverFacesParameters",             "FacesToUncover:="      , [12,16,18]         ]     ] ) </pre>

	]
	])

<b>VB Syntax</b>	UncoverFaces <SelectionsArray> <parameters>
<b>VB Example</b>	<pre> oEditor.UncoverFaces  Array("NAME:Selections",       "Selections:=" , "Box1",       "NewPartsModelFlag:=" , "Model")  Array("NAME:Parameters",       Array("NAME:UncoverFacesParameters",             "FacesToUncover:=" , Array(12,16,18))) </pre>

## Unite

Unites the specified objects.

UI Access	Modeler > Boolean > Unite		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
	<parameters>	Array	Structured array. Array ("NAME:UniteParameters", "KeepOriginals:=" , <boolean>)

<b>Return Value</b>	None.
---------------------	-------

<b>Python Syntax</b>	Unite(<SelectionsArray>, <parameters>)
<b>Python Example</b>	<pre> oEditor.Unite(     [         "NAME:Selections",         "Selections:=", "Rectangle1,Rectangle2"     ],     [         "NAME:UniteParameters",         "KeepOriginals:=", False     ] ) </pre>

<b>VB Syntax</b>	Unite <SelectionsArray> <parameters>
<b>VB Example</b>	<pre> oEditor.Unite     Array("NAME:Selections",         "Selections:=", "Rectangle1,Rectangle2")     Array("NAME:UniteParameters",         "KeepOriginals:=", false) ) </pre>

## Ungroup

Ungroups a specified history tree group.

<b>UI Access</b>	<b>Modeler &gt; Group &gt; Ungroup</b>		
<b>Parameters</b>	Name <i>&lt;Groups&gt;</i>	Type Array	Description Structured array. Array("Groups:=", <array of group names to ungroup>)
<b>Return Value</b>	None		

<b>Python Syntax</b>	Ungroup(<Groups>)
<b>Python Example</b>	<code>oEditor.Ungroup(["Groups:=", ["Group1"]])</code>

<b>VB Syntax</b>	Ungroup <Groups>
<b>VB Example</b>	<code>oEditor.Ungroup Array("Groups:=", Array("Group1"))</code>

## WrapSheet

Wraps a sheet object to another object.

<b>UI Access</b>	None.		
<b>Parameters</b>	Name <i>&lt;SelectionsArray&gt;</i>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .

<b>Return Value</b>	None.
---------------------	-------

<b>Python Syntax</b>	WrapSheet (<SelectionsArray>, <Parameters>)
<b>Python Example</b>	<pre> oEditor.WrapSheet(     [         "NAME:Selections",         "Selections:=", "Rectangle1,Box1"     ],     [         "NAME:WrapSheetParameters",         "Imprinted:=", True     ] ) </pre>

<b>VB Syntax</b>	WrapSheet <SelectionsArray> <Parameters>
<b>VB Example</b>	<pre> oEditor.WrapSheet     Array("NAME:Selections",         "Selections:=", "Rectangle1,Box1"),     Array("NAME:WrapSheetParameters",         "Imprinted:=", true) </pre>

## Other oEditor Commands

[AddDefinitionFromBlock](#)

[AddDefinitionFromLibFile](#)

[AddViewOrientation](#)

[BreakUDMConnection](#)

[ChangeProperty](#)

[Delete](#)

[FitAll](#)

[GetBodyNamesByPosition](#)

[GetChildNames \[Modeler\]](#)

[GetChildObject \[Modeler\]](#)

[GetChildTypes \[Modeler\]](#)

[GetEdgeByPosition](#)

[GetEdgeIDsFromObject](#)

[GetEdgeIDsFromFace](#)

[GetEntityListIDByName](#)

[GetExtendedDefinitionObject](#)

[GetFaceArea](#)

[GetFaceByPosition](#)

[GetFaceCenter](#)

[GetFaceIDs](#)

[GetGeometryModelerMode](#)

[GetMatchedObjectName](#)  
[GetModelBoundingBox](#)  
[GetModelUnits](#)  
[GetNumObjects](#)  
[GetObjectIDByName](#)  
[GetObjectName](#)  
[GetObjectNameByFaceID](#)  
[GetObjectsByMaterial](#)  
[GetObjectsInGroup](#)  
[GetObjectVolume](#)  
[GetPropertyValue](#)  
[GetPropEvaluatedValue](#)  
[GetPropNames](#)  
[GetPropSIValue](#)  
[GetPropValue](#)  
[GetSelections](#)  
 [GetUserPosition](#)  
[GetVertexIDsFromEdge](#)  
[GetVertexIDsFromFace](#)  
[GetVertexIDsFromObject](#)  
[GetVertexPosition](#)

---

[OpenExternalEditor](#)[PageSetup](#)[RenamePart](#)[SetPropValue \[Modeler\]](#)

## AddDefinitionFromBlock

Adds a material definition from block text (same definition format as would be contained in the material library file) by library type (using definition folder name). This scripting command directly supports the .AMAT (or .ASURF) definition formats.

UI Access	N/A		
Parameters	Name	Type	Description
	<defBlock>	String	Text of the new material definition in block form.
	<defFolderName>	String	Library type (by definition folder name)
	<newTimeStamp>	String	New timestamp (time_t as integer number of seconds since 1/1/1970 12:00am, as string), default is current time
	<replaceExisting>	Boolean	True to replace existing, False to choose a new unique name if an existing definition is found
Return Value	A property scripting object for the definition.		

P-yt-h-o-n-S-

AddDefinitionFromBlock(&lt;defBlock&gt;,&lt;defFolderName&gt;,&lt;newTimeStamp&gt;,&lt;replaceExisting&gt;)

<b>y-</b> <b>nt-</b> <b>ax</b>	
<b>P-</b> <b>yt-</b> <b>h-</b> <b>o-</b> <b>n-</b> <b>E-</b> <b>x-</b> <b>a-</b> <b>m-</b> <b>pl-</b> <b>e</b>	<pre> oProject = oDesktop.NewProject()  oProject.InsertDesign("HFSS", "HFSSDesign1", "DrivenModal", "")  oDesign = oProject.SetActiveDesign("HFSSDesign1")  oEditor = oDesign.SetActiveEditor("3D Modeler")  oEditor.CreateBox()  [      "NAME:BoxParameters",          "XPosition:=" , "-0.4mm",         "YPosition:=" , "-1mm",         "ZPosition:=" , "0mm",         "XSize:=" , "1.4mm",         "YSIZE:=" , "1.6mm",         "ZSize:=" , "0.6mm"  ] , [      "NAME:Attributes", </pre>

```
"Name:=" , "Box1",
"Flags:=" , "", ,
"Color:=" , "(143 175 143)",
"Transparency:=" , 0,
"PartCoordinateSystem:=", "Global",
"UDMID:=" , "", ,
"MaterialValue:=" , "\\"vacuum\\",
"SurfaceMaterialValue:=" , "\\"\\",
"SolveInside:=" , True,
"ShellElement:=" , False,
"ShellElementThickness:=" , "0mm",
"IsMaterialEditable:=" , True,
"UseMaterialAppearance:=" , False,
```



```
[  
    "NAME:Geometry3DAttributeTab",  
    [  
        "NAME:PropServers",  
        "Box1"  
    ],  
    [  
        "NAME:ChangedProps",  
        [  
            "NAME:Material",  
            "Value:=", materialNameInQuotes  
        ]  
    ]
```

```
[ ]  
]  
])
```

## AddDefinitionFromLibFile

Adds a material definition from a library file (e.g. AMAT file), by name and library type (using definition folder name) . This scripting command directly supports the .AMAT (or .ASURF) definition formats.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<FilePath>	String	Path of the library file (i.e. AMAT file or ASURF file)
	<defName>	String	Which definition to use, required because a lib file can have multiple definitions.
	<defFolderName>	String	Library type (by definition folder name).
	<newTimeStamp>	String	New timestamp string (time_t as integer, number of seconds since 1/1/1970 12:00am), default is current time
	<replaceExisting>	Boolean	True to replace existing, False to choose a new unique name if an existing definition is found, default is False
<b>Return Value</b>	Property scripting object for the definition.		

<b>Python Syntax</b>	AddDefinitionFromLibFile(<FilePath>, <defName>, <defFolderName>, <newTimeStamp>,<replaceExisting>)
<b>Python Example</b>	<pre>oProject = oDesktop.NewProject() oProject.InsertDesign("HFSS", "HFSSDesign1", "DrivenModal", "") oDesign = oProject.SetActiveDesign("HFSSDesign1")</pre>

```
oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.CreateBox(
    [
        "NAME:BoxParameters",
        "XPosition:=" , "-0.4mm",
        "YPosition:=" , "-1mm",
        "ZPosition:=" , "0mm",
        "XSize:=" , "1.4mm",
        "YSIZE:=" , "1.6mm",
        "ZSize:=" , "0.6mm"
    ],
    [
        "NAME:Attributes",
        "Name:=" , "Box1",
        "Flags:=" , ""
    ]
)
```

```
"Color:="           , "(143 175 143)",  
  
"Transparency:="   , 0,  
  
"PartCoordinateSystem:=", "Global",  
  
"UDMId:="          , "",  
  
"MaterialValue:="   , "\\"vacuum\\\"",  
  
"SurfaceMaterialValue:=", "\\"\\\"",  
  
"SolveInside:="     , True,  
  
"ShellElement:="    , False,  
  
"ShellElementThickness:=", "0mm",  
  
"IsMaterialEditable:=" , True,  
  
"UseMaterialAppearance:=" , False,  
  
"IsLightweight:="    , False  
])  
oDefinitionManager = oProject.GetDefinitionManager()
```

```
scriptDir = os.path.dirname(os.path.realpath(__file__))

amatFilePath = scriptDir + '/material0.txt'

materialName = "material0"

materialNameInQuotes = "\"" + materialName + "\""

added = oDefinitionManager.AddDefinitionFromLibFile(amatFilePath, materialName, "Materials", "")

addedName = ''

    if isinstance(added, basestring):

addedName = added

    elif isinstance(added, list):

addedName = added[0]

else:

    addedName = added.GetName().replace("Materials:", "")

AddInfoMessage(os.path.basename(__file__) + " result: " + addedName)

oEditor.ChangeProperty(
    [
        "NAME:AllTabs",
        [
            "NAME:Geometry3DAttributeTab",
            [

```

```

        "NAME:PropServers",
        "Box1"
    ],
    [
        "NAME:ChangedProps",
        [
            "NAME:Material",
            "Value:=", materialNameInQuotes
        ]
    ]
]
)

```

## AddViewOrientation

Creates a new orientation using a script. Input parameters can be *either* vector components *or* angles. Local view orientation specific to a design can be saved in project by calling `Save()` on the project object.

**Note:**

You cannot use the name of an existing view. Default views include: top, bottom, right, left, front, back, trimetric, dimetric, and isometric.

UI Access	View > Modify Attributes > Orientation List. Enter parameters and click <b>Add</b> .		
Parameters	Name	Type	Description

	<code>&lt;orientationName&gt;</code>	String	Name of the new orientation.
	<code>&lt;isGlobalOrientation&gt;</code>	Long	1 to save global; else 0.
	<code>&lt;OrientationVectorComponents&gt;</code>	Array	Structured array. Use <u>either</u> <code>&lt;OrientationVectorComponents&gt;</code> <u>or</u> <code>&lt;OrientationAnglesArray&gt;</code> .  Array ("NAME:OrientationVectorComponents", Array ("NAME:Direction", <float>, <float>, <float>, Array ("NAME:Up", <float>, <float>, <float>))
	<code>&lt;OrientationAnglesArray&gt;</code>	Array	Structured array. Use <u>either</u> <code>&lt;OrientationVectorComponents&gt;</code> <u>or</u> <code>&lt;OrientationAnglesArray&gt;</code> .  Array ("NAME:OrientationAngles", "Psi:=" , <float>, "Theta:=" , <float>, "Phi:=" , <float>))
	<code>&lt;ProjectionParams&gt;</code>	Array	<i>Optional.</i> Structured array containing arrays of four optional parameters: Scaling, Translation, Projection Type, and Projection Limits.  Array ("NAME:ProjectionParams",

```
        Array("NAME:Scaling",
              <float>,
              <float>,
              <float>
            ),
        Array("NAME:Translation",
              <float>,
              <float>,
              <float>
            ),
        "ProjectionType:=" ,
<string "Orthographic" or "Perspective">,
        Array("NAME:ProjectionLimits",
              <float>,
              <float>,
              <float>,
              <float>,
              <float>,
              <float>
            )
      )
```

<b>Return Value</b>	None.
---------------------	-------

<b>Python Syntax</b>	AddViewOrientation(<orientationName>, <isGlobalOrientation>, <OrientationVectorComponents or OrientationAnglesArray>, <ProjectionParams>)
<b>Python Example</b>	<p><b>Example Using Angles and Projection Parameters:</b></p> <pre>oEditor.AddViewOrientation("orientation3", 0,                            ["NAME:OrientationAngles",                             "Psi:="                  , -161.565002441406,                             "Theta:="                 , 32.3115005493164,                             "Phi:="                  , 0      ],                            ["NAME:ProjectionParams",                             ["NAME:Scaling",                              0.591607987880707,                              0.591607987880707,                              0.591607928276062                             ],                             ["NAME:Translation",</pre>

```
        0,  
        2.38418579101563E-07,  
        -5.7486834526062  
    ],  
    "ProjectionType": "Orthographic",  
  
    ["NAME:ProjectionLimits",  
  
     -2.44362282752991,  
     2.44362282752991,  
  
     -1,  
  
     1,  
     0.625207424163818,  
     10.8721580505371  
    ]  
]  
)
```

**Example Using Vector without Projection Parameters:**

```
oEditor.AddViewOrientation("orientation6", 0,  
    ["NAME:OrientationVectorComponents",
```

```
[ "NAME:Direction",
    -0.801783978939056,
    -0.267262011766434,
    -0.534521996974945
],
[ "NAME:Up",
    -0.507091999053955,
    -0.169030994176865,
    0.845155000686646
]
])
```

<b>VB Syntax</b>	AddViewOrientation <orientationName>, <isGlobalOrientation>, <OrientationVectorComponents or OrientationAnglesArray>, <ProjectionParams>
<b>VB Example</b>	<b>Example Using Angles and Projection Parameters:</b> <pre>oEditor.AddViewOrientation "orientation3", 0,     Array("NAME:OrientationAngles",         "Psi:=" , -161.565002441406,         "Theta:=" , 32.3115005493164,         "Phi:=" , 0 )) ,</pre>

```
Array("NAME:ProjectionParams",
      Array("NAME:Scaling",
            0.591607987880707,
            0.591607987880707,
            0.591607928276062
      ),
      Array("NAME:Translation",
            0,
            2.38418579101563E-07,
            -5.7486834526062
      ),
      "ProjectionType:=" , "Orthographic",
      Array("NAME:ProjectionLimits",
            -2.44362282752991,
            2.44362282752991,
```

```
        -1,  
  
        1,  
        0.625207424163818,  
        10.8721580505371  
    )  
)
```

**Example Using Vector without Projection Parameters:**

```
oEditor.AddViewOrientation "orientation6", 0,  
Array("NAME:OrientationVectorComponents",  
    Array("NAME:Direction",  
        -0.801783978939056,  
        -0.267262011766434,  
        -0.534521996974945  
    ),  
    Array("NAME:Up",  
        -0.507091999053955,  
        -0.169030994176865,  
        0.845155000686646
```

	) )
--	--------

## BreakUDMConnection

Breaks a current UDM connection to SpaceClaim.

<b>UI Access</b>	Break Connection.						
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;SelectionsArray&gt;</td> <td>Array</td> <td>Structured array. See: <a href="#">SelectionsArray</a>.</td> </tr> </table>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
Name	Type	Description					
<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	BreakUDMConnection (<SelectionsArray>)
<b>Python Example</b>	<pre> oEditor.BreakUDMConnection (     [         "NAME:Selections",         "Selections:=" , "SpaceClaim1"     ] ) </pre>

<b>VB Syntax</b>	BreakUDMConnection <SelectionsArray>
<b>VB Example</b>	<pre> oEditor.BreakUDMConnection Array("NAME:Selections",     "Selections:=", "") </pre>

## ChangeProperty

Changes the properties of an object in the history tree.

<b>UI Access</b>	Right-click an object in the History Tree and select <b>Properties</b> .		
<b>Parameters</b>	Name <i>&lt;propertyArgs&gt;</i>	Type Array	Description Structured array. The properties vary depending on the object. Due to the number of potential configurations, it is recommended that you generate this script using the UI's <b>Automation</b> tab.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ChangeProperty( <i>&lt;propertyArgs&gt;</i> )
<b>Python Example</b>	<p><b>Example Changing the Position of a Box:</b></p> <pre>oEditor.ChangeProperty( [     "NAME:AllTabs",     [         "NAME:Geometry3DCmdTab",         [             "NAME:PropServers" ,             "Box1&gt;CreateBox:1"         ],         [             "NAME:ChangedProps",             [                 "NAME:Position",                 [                     "NAME:Y"                 ]             ]         ]     ] )</pre>

```
        "X:="           , "0.35in",
        "Y:="           , "0.55in",
        "Z:="           , "0in"
    ]
]
])
])
```

### **Example Changing a Box's Material and Wireframe Display:**

```
oEditor.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:Geometry3DAttributeTab",
        [
            "NAME:PropServers" ,
            "Box1"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Material",
                "Value:="           , "\\"vacuum\\\""
            ],
            [
                "NAME:Display Wireframe",
                "Value:="           , True
            ]
        ]
    ]
])
```

### **Example Changing the Model Depth:**

```
oEditor.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:Maxwell2D",
        [
            "NAME:PropServers",
            "Design Settings"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Model settings/Depth",
                "Value:=" , "2.4mm"
            ]
        ]
    ]
])
```

**Example Changing 2D Transient Use of Band Mapping Angle:**

```
oEditor.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:MeshSetupTab",
        [
            "NAME:PropServers",
            "MeshSetup:CylindricalGap1"
        ],
        [
            "NAME:ChangedProps",
            [

```

```
                "NAME:Use Band Mapping Angle",
                "Value:="      , True
            ]
        ]
    ]
])

oEditor.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:MeshSetupTab",
        [
            "NAME:PropServers",
            "MeshSetup:CylindricalGap1"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Band Mapping Angle",
                "Value:="      , "2deg"
            ]
        ]
    ]
])
```

**Example Edit VirtualPort Excitation by changing component properties:**

```
oEditor.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:Component Data",
        [
            "NAME:PropServers",
            "BoundarySetup:LC1_1_Port1"
        ]
    ]
])
```

```
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Voltage",
                "Value:=" , "10mV"
            ]
        ]
    )
...

```

VB Syntax	ChangeProperty <propertyArgs>
<b>VB Example</b>	<p><b>Example Changing the Position of a Box:</b></p> <pre>oEditor.ChangeProperty Array("NAME:AllTabs",     Array("NAME:Geometry3DCmdTab",         Array("NAME:PropServers" ,         "Box1&gt;CreateBox:1"     ),     Array("NAME:ChangedProps",         Array("NAME:Position",             "X:=" , "0.35in",             "Y:=" , "0.55in",             "Z:=" , "0in"         )     ) ) )</pre>

**Example Changing a Box's Material and Wireframe Display:**

```
oEditor.ChangeProperty(
    Array("NAME:AllTabs",
        Array("NAME:Geometry3DAttributeTab",
            Array("NAME:PropServers",
                "Box1"
            ),
            Array("NAME:ChangedProps",
                Array("NAME:Material",
                    "Value:=", "\vacuum\""
                ),
                Array("NAME:Display Wireframe",
                    "Value:=", True
                )
            )
        )
    )
)
```

**Example Changing the Model Depth:**

```
oEditor.ChangeProperty
Array("NAME:AllTabs",
    Array("NAME:Maxwell2D",
        Array("NAME:PropServers"
            "Design Settings"
        ),
        Array("NAME:ChangedProps",
            Array("NAME:Model settings/Depth",
                "Value:=", "2.4mm"
            )
        )
    )
)
```

**Example Changing 2D Transient Use of Band Mapping Angle:**

```
oEditor.ChangeProperty
Array("NAME:AllTabs",
    Array("NAME:MeshSetupTab",
        Array("NAME:PropServers"
            "MeshSetup:CylindricalGap1"
        ),
        Array("NAME:ChangedProps",
            Array("NAME:Use Band Mapping Angle",
                "Value:=", True
            )
        )
    )
)

oEditor.ChangeProperty
Array("NAME:AllTabs",
    Array("NAME:MeshSetupTab",
        Array("NAME:PropServers"
            "MeshSetup:CylindricalGap1"
        ),
        Array("NAME:ChangedProps",
            Array("NAME:Band Mapping Angle",
                "Value:=", "2deg"
            )
        )
    )
)
```

## Delete

Deletes the specified object(s).

**UI Access**

**Edit > Delete.**

<b>Parameters</b>	Name <i>&lt;SelectionsArray&gt;</i>	Type Array	Description Structured array. See: <a href="#">SelectionsArray</a> .
<b>Return Value</b>	None.		

<b>Python Syntax</b>	Delete(<SelectionsArray>)
<b>Python Example</b>	<pre>oEditor.Delete(     [ "NAME:Selections",       "Selections:=", "Rectangle1,Rectangle2"     ] )</pre>

<b>VB Syntax</b>	Delete <SelectionsArray>
<b>VB Example</b>	<pre>oEditor.Delete Array("NAME:Selections", "Selections:=", "Rectangle1,Rectangle2")</pre>

## GetBodyNamesByPosition

Returns the names of objects that contact a specified point.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;positionParameters&gt;</i>	Type Array	Description Structured array containing position coordinates for active coordinate system.  Array ("NAME:Parameters", "XPosition:=", <string>,

			"YPosition:=", <string>, "ZPosition:=", <string>)
<b>Return Value</b>	Array containing string object names.		

<b>Python Syntax</b>	GetBodyNamesByPosition (<positionParameters>)
<b>Python Example</b>	<pre>oEditor.GetBodyNamesByPosition(     ["NAME:Parameters",      "XPosition:=", "0mm",      "YPosition:=", "15mm",      "ZPosition:=", "0mm"     ] )</pre>

<b>VB Syntax</b>	GetBodyNamesByPosition <positionParameters>
<b>VB Example</b>	<pre>oEditor.GetBodyNamesByPosition Array("NAME:Parameters", _     "XPosition:=", "0mm", _     "YPosition:=", "15mm", _     "ZPosition:=", "0mm")</pre>

## GetEdgeByPosition

Returns the ID for edge(s) that contact a specified point.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;positionParameters&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array ("NAME:EdgeParameters",       "BodyName:=", &lt;string object name&gt;,       "Xposition:=", &lt;string&gt;,       "YPosition:=", &lt;string&gt;,       "ZPosition:=", &lt;string&gt;)</pre> <p><b>Note:</b></p> <p>For 2D XY Designs, ZPosition should be set to "0".</p> <p>For 2D RZ Designs, YPosition should be set to "0".</p> </td></tr> </table>	Name	Type	Description	<positionParameters>	Array	<p>Structured array.</p> <pre>Array ("NAME:EdgeParameters",       "BodyName:=", &lt;string object name&gt;,       "Xposition:=", &lt;string&gt;,       "YPosition:=", &lt;string&gt;,       "ZPosition:=", &lt;string&gt;)</pre> <p><b>Note:</b></p> <p>For 2D XY Designs, ZPosition should be set to "0".</p> <p>For 2D RZ Designs, YPosition should be set to "0".</p>		
Name	Type	Description							
<positionParameters>	Array	<p>Structured array.</p> <pre>Array ("NAME:EdgeParameters",       "BodyName:=", &lt;string object name&gt;,       "Xposition:=", &lt;string&gt;,       "YPosition:=", &lt;string&gt;,       "ZPosition:=", &lt;string&gt;)</pre> <p><b>Note:</b></p> <p>For 2D XY Designs, ZPosition should be set to "0".</p> <p>For 2D RZ Designs, YPosition should be set to "0".</p>							
<b>Return Value</b>	Array containing string edge IDs.								

<b>Python Syntax</b>	GetEdgeByPosition (<positionParameters>)
<b>Python Example</b>	<pre>oEditor.GetEdgeByPosition(     [ "NAME:EdgeParameters",       "BodyName:=", "Box1",       "Xposition:=", "10mm",</pre>

```
    "YPosition:=", "0mm",
    "ZPosition:=", "10mm"
]
)
```

<b>VB Syntax</b>	GetEdgeByPosition <positionParameters>
<b>VB Example</b>	<pre>oEditor.GetEdgeByPosition Array("NAME:EdgeParameters",     "BodyName:=", "Box1",     "Xposition:=", "10mm",     "YPosition:=", "0mm",     "ZPosition:=", "10mm")</pre>

## GetEdgeIDsFromFace

Returns the edge IDs for a specified Face ID.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;faceID&gt;</td><td>Integer</td><td>ID of the specified face.</td></tr></table>			Name	Type	Description	<faceID>	Integer	ID of the specified face.
Name	Type	Description							
<faceID>	Integer	ID of the specified face.							
<b>Return Value</b>	Array containing string edge IDs.								

---

<b>Python Syntax</b>	GetEdgeIDsFromFace (<faceID>)
<b>Python Example</b>	<code>oEditor.GetEdgeIDsFromFace (20)</code>

<b>VB Syntax</b>	GetEdgeIDsFromFace <faceID>
<b>VB Example</b>	<code>oEditor.GetEdgeIDsFromFace 20</code>

## GetEdgeIDsFromObject

Returns the edge IDs for a specified object.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <object>	Type String	Description Object name.
<b>Return Value</b>	Array containing string edge IDs.		

<b>Python Syntax</b>	GetEdgeIDsFromObject (<object>)
<b>Python Example</b>	<code>oEditor.GetEdgeIDsFromObject ("Box1")</code>

<b>VB Syntax</b>	GetEdgeIDsFromObject <object>
<b>VB Example</b>	<code>oEditor.GetEdgeIDsFromObject "Box1"</code>

## GetFaceArea

Returns the area of a specified face.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;faceID&gt;</td><td>Integer</td><td>ID of specified face.</td></tr></table>			Name	Type	Description	<faceID>	Integer	ID of specified face.
Name	Type	Description							
<faceID>	Integer	ID of specified face.							
<b>Return Value</b>	Long integer of face area.								

<b>Python Syntax</b>	GetFaceArea (<faceID>)
<b>Python Example</b>	<code>oEditor.GetFaceArea (19)</code>

<b>VB Syntax</b>	GetFaceArea <faceID>
<b>VB Example</b>	<code>oEditor.GetFaceArea 19</code>

## GetFaceCenter

Returns the center position of a specified face.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;planarFaceID&gt;</td><td>Long</td><td>Face ID</td></tr></table>			Name	Type	Description	<planarFaceID>	Long	Face ID
Name	Type	Description							
<planarFaceID>	Long	Face ID							
<b>Return Value</b>	Array containing string X, Y, Z coordinates.								

<b>Python Syntax</b>	GetFaceCenter (<planarFaceID>)
<b>Python Example</b>	<code>oEditor.GetFaceCenter(12)</code>

<b>VB Syntax</b>	GetFaceCenter <planarFaceID>
<b>VB Example</b>	<code>oEditor.GetFaceCenter 12</code>

## GetFaceByPosition

Returns the face ID located at a specified position.

### Note:

The coordinates must point to exactly one face, not a vertex or edge where two or more faces join.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;FaceParameters&gt;</td> <td>Array</td> <td> <p>Structured Array.</p> <pre>Array("NAME:FaceParameters",       "BodyName:=", &lt;string&gt;,       "XPosition:=", &lt;string&gt;,       "YPosition:=", &lt;string&gt;,       "ZPosition:=", &lt;string&gt;)</pre> <p>For 2D XY Designs, ZPosition should be set to "0".</p> <p>For 2D RZ Designs, YPosition should be set to "0".</p> </td> </tr> </tbody> </table>	Name	Type	Description	<FaceParameters>	Array	<p>Structured Array.</p> <pre>Array("NAME:FaceParameters",       "BodyName:=", &lt;string&gt;,       "XPosition:=", &lt;string&gt;,       "YPosition:=", &lt;string&gt;,       "ZPosition:=", &lt;string&gt;)</pre> <p>For 2D XY Designs, ZPosition should be set to "0".</p> <p>For 2D RZ Designs, YPosition should be set to "0".</p>		
Name	Type	Description							
<FaceParameters>	Array	<p>Structured Array.</p> <pre>Array("NAME:FaceParameters",       "BodyName:=", &lt;string&gt;,       "XPosition:=", &lt;string&gt;,       "YPosition:=", &lt;string&gt;,       "ZPosition:=", &lt;string&gt;)</pre> <p>For 2D XY Designs, ZPosition should be set to "0".</p> <p>For 2D RZ Designs, YPosition should be set to "0".</p>							

<b>Return Value</b>	Integer Face ID.
---------------------	------------------

<b>Python Syntax</b>	GetFaceByPosition(<FaceParameters>)
<b>Python Example</b>	<pre>oEditor.GetFaceByPosition(     [         "NAME:FaceParameters",         "BodyName:=", "Box1",         "XPosition:=", "0.2mm",         "YPosition:=", "-0.2mm",         "ZPosition:=", "0.4mm"     ] )</pre>

<b>VB Syntax</b>	GetFaceByPosition <FaceParameters>
<b>VB Example</b>	<pre>dim FaceID FaceID = oEditor.GetFaceByPosition Array("NAME:FaceParameters",     "BodyName:=", "Box1",     "XPosition:=", "0.2mm",     "YPosition:=", "-0.2mm",     "ZPosition:=", "0.4mm")</pre>

## GetFaceIDs

Returns the face IDs associated with a specified object.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;objectName&gt;</code></td> <td>String</td> <td>Object name.</td> </tr> </tbody> </table>	Name	Type	Description	<code>&lt;objectName&gt;</code>	String	Object name.
Name	Type	Description					
<code>&lt;objectName&gt;</code>	String	Object name.					
<b>Return Value</b>	Array containing string face IDs.						

<b>Python Syntax</b>	<code>GetFaceIDs (&lt;objectName&gt;)</code>
<b>Python Example</b>	<code>oEditor.GetFaceIDs ('Box1')</code>

<b>VB Syntax</b>	<code>GetFaceIDs &lt;objectName&gt;</code>
<b>VB Example</b>	<code>oEditor.GetFaceIDs "Box1"</code>

## GetGeometryModelerMode

Returns the modeler mode (3D or XY) for the current design. This lets you know whether the current model is 2D or 3D.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String containing the modeling mode ("3D" or "XY").

<b>Python Syntax</b>	GetGeometryModelerMode()
<b>Python Example</b>	<code>oEditor.GetGeometryModelerMode ()</code>

<b>VB Syntax</b>	GetGeometryModelerMode
<b>VB Example</b>	<code>oEditor.GetGeometryModelerMode</code>

## GetModelBoundingBox

Returns the bounding box of the current model.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array containing string Xmin, Ymin, Zmin, Xmax, Ymax, and Zmax values of the bounding box.

<b>Python Syntax</b>	GetModelBoundingBox()
<b>Python Example</b>	<code>oEditor.GetModelBoundingBox ()</code>

<b>VB Syntax</b>	GeModelBoundingBox
<b>VB Example</b>	<code>oEditor.GetModelBoundingBox</code>

## GetObjectIDByName

Given an object's name, returns its ID. IDs are used with [CreateEntityList](#).

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;objectName&gt;</i>	Type String	Description Object name.
<b>Return Value</b>	Integer object ID.		

<b>Python Syntax</b>	GetObjectIDByName(< <i>objectName</i> >)
<b>Python Example</b>	<code>oEditor.GetObjectIDByName ('Box1')</code>

<b>VB Syntax</b>	GetObjectIDByName < <i>objectName</i> >
<b>VB Example</b>	<code>oEditor.GetObjectIDByName "Box1"</code>

## GetObjectName

Returns an object's name from its specified base index (creation order).

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;index&gt;</i>	Type Integer	Description Base index (where '0' is the first item created)
<b>Return Value</b>	String containing object name.		

<b>Python Syntax</b>	GetObjectName(<index>)
<b>Python Example</b>	<code>oEditor.GetObjectName(3)</code>

<b>VB Syntax</b>	GetObjectName <index>
<b>VB Example</b>	<code>oEditor.GetObjectName 3</code>

### GetObjectByNameFaceID

Returns an object name given a face ID.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;FaceID&gt;</td><td>Integer</td><td>The face ID.</td></tr></table>			Name	Type	Description	<FaceID>	Integer	The face ID.
Name	Type	Description							
<FaceID>	Integer	The face ID.							
<b>Return Value</b>	String containing object name.								

<b>Python Syntax</b>	GetObjectByNameFaceID (<FaceID>)
<b>Python Example</b>	<code>oEditor.GetObjectByNameFaceID(88)</code>

<b>VB Syntax</b>	GetObjectByNameFaceID <FaceID>
<b>VB Example</b>	<code>oEditor.GetObjectByNameFaceID 10</code>

## GetObjectsByMaterial

Returns a list of objects of a specified material.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;material&gt;</td> <td>String</td> <td>Material name.</td> </tr> </tbody> </table>	Name	Type	Description	<material>	String	Material name.
Name	Type	Description					
<material>	String	Material name.					
<b>Return Value</b>	Array containing string object names.						

<b>Python Syntax</b>	GetObjectsByMaterial (<material>)
<b>Python Example</b>	<code>oEditor.GetObjectsByMaterial ('copper')</code>

<b>VB Syntax</b>	GetObjectsByMaterial <material>
<b>VB Example</b>	<code>oEditor.GetObjectsByMaterial "copper"</code>

## GetObjectsInGroup

Returns a list of objects in a specified group.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;group&gt;</td> <td>String</td> <td>Group name. One of:<ul style="list-style-type: none"><li>• "&lt;materialName&gt;"</li></ul></td> </tr> </tbody> </table>	Name	Type	Description	<group>	String	Group name. One of: <ul style="list-style-type: none"><li>• "&lt;materialName&gt;"</li></ul>
Name	Type	Description					
<group>	String	Group name. One of: <ul style="list-style-type: none"><li>• "&lt;materialName&gt;"</li></ul>					

			<ul style="list-style-type: none"><li>• "&lt;assignmentName&gt;"</li><li>• "Non Model"</li><li>• "Solids"</li><li>• "Unclassified"</li><li>• "Sheets"</li><li>• "Lines"</li></ul>
<b>Return Value</b>	Array containing string object names.		

<b>Python Syntax</b>	GetObjectsInGroup (<group>)
<b>Python Example</b>	<code>oEditor.GetObjectsInGroup ('Sheets')</code>

<b>VB Syntax</b>	GetObjectsInGroup <group>
<b>VB Example</b>	<code>oEditor.GetObjectsInGroup "Sheets"</code>

## GetMatchedObjectName

Returns all object names containing the input text string.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;wildcardText&gt;</td><td>String</td><td>Text string present in object name(s).</td></tr></tbody></table>			Name	Type	Description	<wildcardText>	String	Text string present in object name(s).
Name	Type	Description							
<wildcardText>	String	Text string present in object name(s).							

<b>Return Value</b>	Array containing string object names.
---------------------	---------------------------------------

<b>Python Syntax</b>	GetMatchedObjectName (<wildcardText>)
<b>Python Example</b>	<code>oEditor.GetMatchedObjectName ('Box*')</code> <code>oEditor.GetMatchedObjectName ('?ox?')</code>

<b>VB Syntax</b>	GetMatchedObjectName <wildcardText>
<b>VB Example</b>	<code>oEditor.GetMatchedObjectName "Box*"</code>

## GetModelUnits

Returns the model's unit of measure.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String containing unit of measure.

<b>Python Syntax</b>	GetModelUnits()
<b>Python Example</b>	<code>oEditor.GetModelUnits ()</code>

<b>VB Syntax</b>	GetModelUnits
------------------	---------------

<b>VB Example</b>	<code>oEditor.GetModelUnits</code>
-------------------	------------------------------------

## GetNumObjects

Returns the number of objects in a design.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Integer number of objects.

<b>Python Syntax</b>	<code>GetNumObjects()</code>
<b>Python Example</b>	<code>oEditor.GetNumObjects ()</code>

<b>VB Syntax</b>	<code>GetNumObjects</code>
<b>VB Example</b>	<code>oEditor.GetNumObjects ()</code>

## GetSelections [Model Editor]

Returns an array of currently selected objects.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array containing object IDs

---

<b>Python Syntax</b>	GetSelections()
<b>Python Example</b>	<code>oEditor.GetSelections ()</code>

<b>VB Syntax</b>	GetSelections
<b>VB Example</b>	<code>oEditor.GetSelections</code>

## GetUserPosition

Returns a user's current coordinates in the 3D Modeler window.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;prompt&gt;</code>	String	"Enter a point." Then click a point in the 3D Modeler window.
<b>Return Value</b>	Array containing X, Y, Z coordinates.		

<b>Python Syntax</b>	GetUserPosition(<prompt>)
<b>Python Example</b>	<code>oEditor.GetUserPosition("Enter a point.")</code>

<b>VB Syntax</b>	GetUserPosition <prompt>
<b>VB Example</b>	<code>oEditor.GetUserPosition "Enter a point."</code>

## GetVertexIDsFromEdge

Returns vertex IDs associated with a specified edge.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;edgeID&gt;</td><td>Integer</td><td>Edge ID.</td></tr></tbody></table>	Name	Type	Description	<edgeID>	Integer	Edge ID.
Name	Type	Description					
<edgeID>	Integer	Edge ID.					
<b>Return Value</b>	Array containing string vertex IDs.						

<b>Python Syntax</b>	GetVertexIDsFromEdge(<edgeID>)
<b>Python Example</b>	<code>oEditor.GetVertexIDsFromEdge(10)</code>

<b>VB Syntax</b>	GetVertexIDsFromEdge <edgeID>
<b>VB Example</b>	<code>oEditor.GetVertexIDsFromEdge 10</code>

## GetVertexIDsFromFace

Returns vertex IDs associated with a specified face.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;faceID&gt;</td><td>Integer</td><td>Face ID.</td></tr></tbody></table>	Name	Type	Description	<faceID>	Integer	Face ID.
Name	Type	Description					
<faceID>	Integer	Face ID.					
<b>Return Value</b>	Array containing string vertex IDs.						

---

<b>Python Syntax</b>	GetVertexIDsFromFace(<faceID>)
<b>Python Example</b>	<code>oEditor.GetVertexIDsFromFace(10)</code>

<b>VB Syntax</b>	GetVertexIDsFromFace <faceID>
<b>VB Example</b>	<code>oEditor.GetVertexIDsFromFace 10</code>

## GetVertexIDsFromObject

Returns vertex IDs associated with a specified object.

<b>UI Access</b>	N/A	
<b>Parameters</b>	Name <i>&lt;objectName&gt;</i>	Type String
<b>Return Value</b>	Description Object name.	
<b>Return Value</b>	Array containing string vertex IDs.	

<b>Python Syntax</b>	GetVertexIDsFromObject(<objectName>)
<b>Python Example</b>	<code>oEditor.GetVertexIDsFromObject('Box1')</code>

<b>VB Syntax</b>	GetVertexIDsFromObject <objectName>
<b>VB Example</b>	<code>oEditor.GetVertexIDsFromObject "Box1"</code>

## GetVertexPosition

Returns an array of coordinates for a specified vertex.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;vertexID&gt;</td><td>Integer</td><td>Vertex ID.</td></tr></tbody></table>	Name	Type	Description	<vertexID>	Integer	Vertex ID.
Name	Type	Description					
<vertexID>	Integer	Vertex ID.					
<b>Return Value</b>	Array containing X, Y, Z coordinates.						

<b>Python Syntax</b>	GetVertexPosition(<vertexID>)
<b>Python Example</b>	<code>oEditor.GetVertexPosition(1)</code>

<b>VB Syntax</b>	GetVertexPosition <vertexID>
<b>VB Example</b>	<code>oEditor.GetVertexPosition 1</code>

## OpenExternalEditor

Launches a SpaceClaim session.

<b>UI Access</b>	<b>Modeler &gt; SpaceClaim Link &gt; Connect to Active Session.</b>						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;SelectionsArray&gt;</td><td>Array</td><td>Structured array. See: <a href="#">SelectionsArray</a>.</td></tr></tbody></table>	Name	Type	Description	<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .
Name	Type	Description					
<SelectionsArray>	Array	Structured array. See: <a href="#">SelectionsArray</a> .					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	OpenExternalEditor (<SelectionsArray>)
<b>Python Example</b>	<pre> oEditor.OpenExternalEditor( [     "NAME:Selections",     "Selections:="           , "SpaceClaim1" ]) </pre>

<b>VB Syntax</b>	OpenExternalEditor <SelectionsArray>
<b>VB Example</b>	<pre> oEditor.OpenExternalEditor Array("NAME:Selections", "Selections:=", "") </pre>

## PageSetup

Specifies Page Setup settings for printing.

<b>UI Access</b>	File > Page Setup.		
<b>Parameters</b>	Name <Parameters>	Type Array	Description Structured array. <pre> [     "NAME:PageSetupData",     "margins:=", &lt;SetupArray&gt; ] </pre>

	<SetupArray>	Array	Structured Array [ "left:=", <string>, "right:=", <string>, "top:=", <string>, "bottom:=", <string>) ]
<b>Return Value</b>	None.		

<b>Python Syntax</b>	PageSetup(<Parameters>)
<b>Python Example</b>	<pre>oEditor.PageSetup([     "NAME:PageSetupData",     "margins:=",     [ "left:=", "10mm",       "right:=", "10mm",       "top:=", "10mm",       "bottom:=", "10mm" ] ])</pre>

<b>VB Syntax</b>	PageSetup <Parameters>
<b>VB Example</b>	<pre> oEditor.PageSetup Array("NAME:PageSetupData",     "margins:=",     Array("left:=", "10mm",         "right:=", "10mm",         "top:=", "10mm",         "bottom:=", "10mm"     ) ) </pre>

## RenamePart

Renames an object.

<b>UI Access</b>	Enter new name in <b>Name</b> field.								
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;renameParametersArray&gt;</td> <td>Array</td> <td>           Structured array.            Array("NAME:Rename Data",               "Old Name:=", &lt;string&gt;,               "New Name:=", &lt;string&gt;             )         </td> </tr> </table>	Name	Type	Description	<renameParametersArray>	Array	Structured array. Array("NAME:Rename Data",               "Old Name:=", <string>,               "New Name:=", <string>             )		
Name	Type	Description							
<renameParametersArray>	Array	Structured array. Array("NAME:Rename Data",               "Old Name:=", <string>,               "New Name:=", <string>             )							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	<code>oEditor.RenamePart(&lt;renameParametersArray&gt;)</code>
<b>Python Example</b>	<pre>oEditor.RenamePart(     [         'NAME:Rename Data',         'Old Name:=' , 'partname',         'New Name:=' , 'newpartname',     ] )</pre>

<b>VB Syntax</b>	<code>oEditor.RenamePart &lt;RenameParametersArray&gt;</code>
<b>VB Example</b>	<pre>oEditor.RenamePart Array("NAME:Rename Data",     "Old Name:=" , "partname",     "New Name:=" , "newpartname", )</pre>

# 11 - Output Variable Script Commands

Output variable commands should be executed by the "OutputVariable" module.

First, obtain the output variable module from oDesign and use it for output variable commands:

```
Set oModule = oDesign.GetModule("OutputVariable")
oModule.<CommandName><args>
```

The commands are:

[CreateOutputVariable](#)

[DeleteOutputVariable](#)

[DoesOutputVariableExist](#)

[EditOutputVariable](#)

[GetOutputVariableValue](#)

[GetOutputVariableValue](#)

[SimValueContext](#)

## CreateOutputVariable (Maxwell)

**Use:** Adds a new output variable to the output variable list. Output variables are associated with a name and an expression. The name of an output variable is not permitted to collide with design variables, Sim values, or other output variable names. It cannot have spaces or any arithmetic or other operators. The definitions cannot be cyclic. For example, A = 2\*B, B=3\*A is not allowed.

**Command:** Maxwell3D>Results>Output Variables, Maxwell2D>Results>Output Variables, or RMxprt>Results>Output Variables.

**Syntax:** CreateOutputVariable <OutputVarName>, <Expression>, <SolutionName>, <ReportTypeName>, <simValueContext>

**Return Value:** None

**Parameters:** <OutputVarName>

Type: <string>

Name of the output variable.

<Expression>

Type: <value>

Value to assign to the variable.

<SolutionName>

Type: <string>

Name of the solution as listed in the **output variable UI**.

For example: "Setup1 : Last Adaptive"

<ReportTypeName>

Type: <string>

The name of the report type as seen in the output variable UI.

<simValueContext>

Type: <variant>

Context for which the output variable expression is being evaluated

**Example:** Set oModule = oDesign.GetModule("OutputVariable")

```
oModule.CreateOutputVariable "magforce", "mag(Force1.Force_x)", _
```

```
"Setup1 : LastAdaptive", "Magnetostatic", Array()
```

## DeleteOutputVariable

Deletes an existing output variable. The variable can only be deleted if it is not in use by any traces.

<b>UI Access</b>	Maxwell > Results > Output Variables. In the <b>Output Variables</b> window, click <b>Delete</b> .			
<b>Parameters</b>	<table border="1"> <tr> <td>Name <i>&lt;OutputVarName&gt;</i></td> <td>Type String</td> <td>Description Name of the output variable.</td> </tr> </table>	Name <i>&lt;OutputVarName&gt;</i>	Type String	Description Name of the output variable.
Name <i>&lt;OutputVarName&gt;</i>	Type String	Description Name of the output variable.		
<b>Return Value</b>	None.			

<b>Python Syntax</b>	DeleteOutputVariable ( <i>&lt;OutputVarName&gt;</i> )
<b>Python Example</b>	<pre>oModule = oDesign.GetModule("OutputVariable") oModule.DeleteOutputVariable ("testNew")</pre>

<b>VB Syntax</b>	DeleteOutputVariable <i>&lt;OutputVarName&gt;</i>
<b>VB Example</b>	<pre>Set oModule = oDesign.GetModule("OutputVariable") oModule.DeleteOutputVariable "testNew"</pre>

## DoesOutputVariableExist

Verifies whether or not a named output variable exists.

<b>UI Access</b>	N/A			
<b>Parameters</b>	<table border="1"> <tr> <td>Name <i>&lt;outputVariableName&gt;</i></td> <td>Type String</td> <td>Description The output variable name.</td> </tr> </table>	Name <i>&lt;outputVariableName&gt;</i>	Type String	Description The output variable name.
Name <i>&lt;outputVariableName&gt;</i>	Type String	Description The output variable name.		

<b>Return Value</b>	Boolean True if the variable exists; False if it does not.
---------------------	--

<b>Python Syntax</b>	DoesOutputVariableExist(<outputVariableName>)
<b>Python Example</b>	<pre> oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() oModule = oDesign.GetModule("OutputVariable") oModule.DoesOutputVariableExist("MyTestVar") </pre>

<b>VB Syntax</b>	DoesOutputVariableExist(<outputVariableName>)
<b>VB Example</b>	<pre> Set oModule = oDesign.GetModule "OutputVariable" oModule.DoesOutputVariableExist "MyTestVar" </pre>

## EditOutputVariable

Changes the name or expression of an existing output variable.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <OrigVarName> <NewExpression> <NewVarName>	Type String String String	Description Name of the original output variable. New value to assign to the variable. New name of the variable if any, else pass empty string.

	<code>&lt;SolutionName&gt;</code>	String	Name of the solution as seen in the output variable UI. For example, "Setup1 : Last Adaptive".
	<code>&lt;ReportType&gt;</code>	String	The name of the report type as seen in the output variable UI.
	<code>&lt;ContextArray&gt;</code>	Array	Structured array containing context for which the output variable expression is being evaluated. <code>Array("Context:=", &lt;string&gt;)</code>
<b>Return Value</b>	None		

<b>Python Syntax</b>	<code>EditOutputVariable (&lt;OrigVarName&gt;, &lt;NewExpression&gt;, &lt;NewVarName&gt;, &lt;SolutionName&gt;, &lt;ReportType&gt;, &lt;ContextArray&gt;)</code>
<b>Python Example</b>	<pre>oModule = oDesign.GetModule("OutputVariable") oModule.EditOutputVariable ("test", "normalize(R1_0.V)", "testNew", "TR", "Standard", [ ])</pre>

<b>VB Syntax</b>	<code>EditOutputVariable &lt;OrigVarName&gt;, &lt;NewExpression&gt;, &lt;NewVarName&gt;, &lt;SolutionName&gt;, &lt;ReportType&gt;, &lt;ContextArray&gt;</code>
<b>VB Example</b>	<pre>Set oModule = oDesign.GetModule("OutputVariable") oModule.EditOutputVariable "test", "dB(S(WavePort1,WavePort1)) ", "testNew", "Setup1 : LastAdaptive", "Modal Solution Data", Array("Domain:=", "Sweep")</pre>

## GetOutputVariableValue

**Use:** Gets the double value of an output variable. Only those expressions that return a double value are supported. The expression is evaluated only for a single point.

**Syntax:** GetOutputVariableValue(<OutputVarName>, <IntrinsicVariation>, <SolutionName>, <ReportTypeName>, <SimValueContext>)

**Return Value:** Double value of the output variable .

**Parameters:** <OutputVarName>

Type: <string>

Name of the output variable.

<IntrinsicVariation>

Type: <string>

A set of intrinsic variable value pairs to use when evaluating the output expression. If no variables are present, a null string must be used. The Eddy Current solution type requires at least the specification of frequency as shown here while the transient solver will require Time as a minimum entry.

For example: "" (Null String)

"freq='60'" (Eddy current example)

<SolutionName>

Type: <string>

Name of the solution as listed in the output variables dialog box as shown in SimValueContext.

For example: "Setup1 : Last Adaptive" or "Setup1 : Transient"

<ReportTypeName>

Type: <string>

The name of the report type as seen in the output variable dialog box.

**Note:**

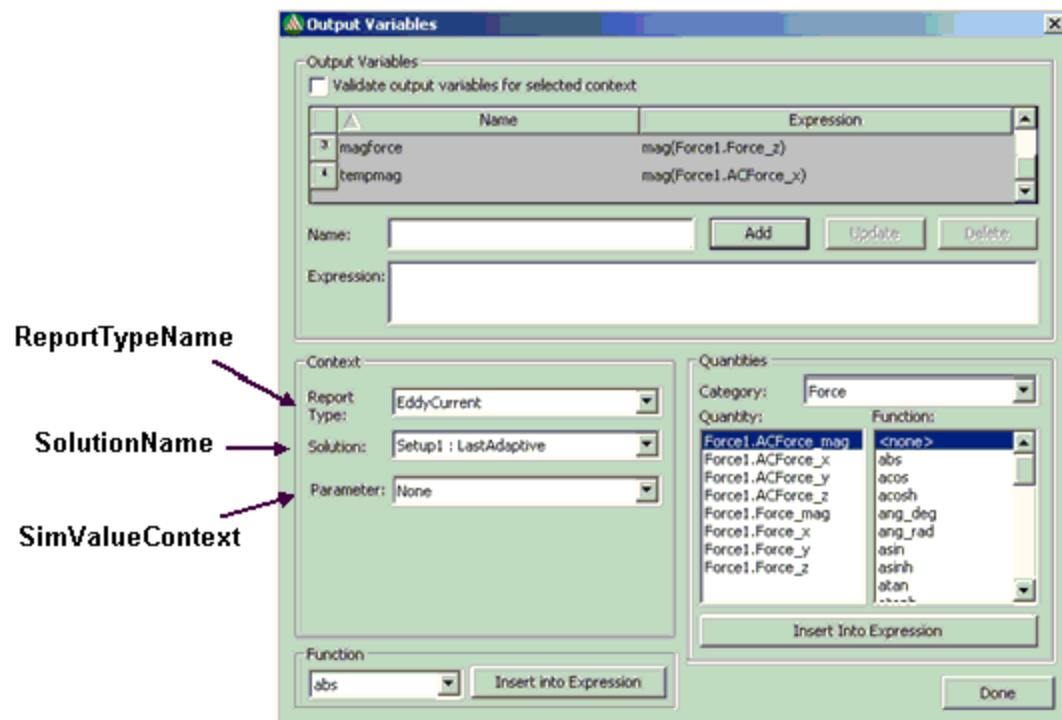
If the Output Variable is based upon a field quantity, then the ReportTypeName must be set to "**Fields**".

<SimValueContext>

Type: <Array>

Context for which the output variable expression is being evaluated. This section is related to the Parameters or Geometry fields in the UI that specify a context for extracting data quantities. This can be an empty string if there is no context.

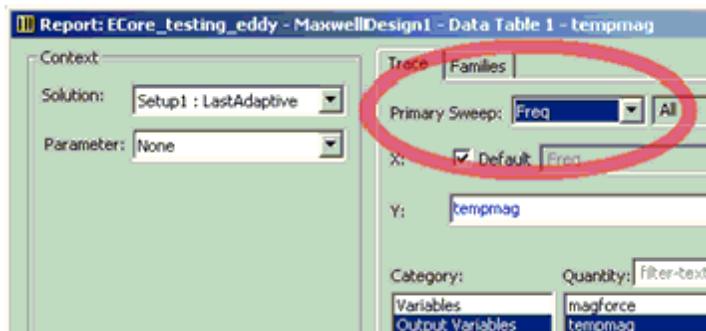
The **Context** section of the Output Variables dialog box contains the information that is required when getting the value of the Output Variable. Note these values when using the dialog box to create an Output Variable.

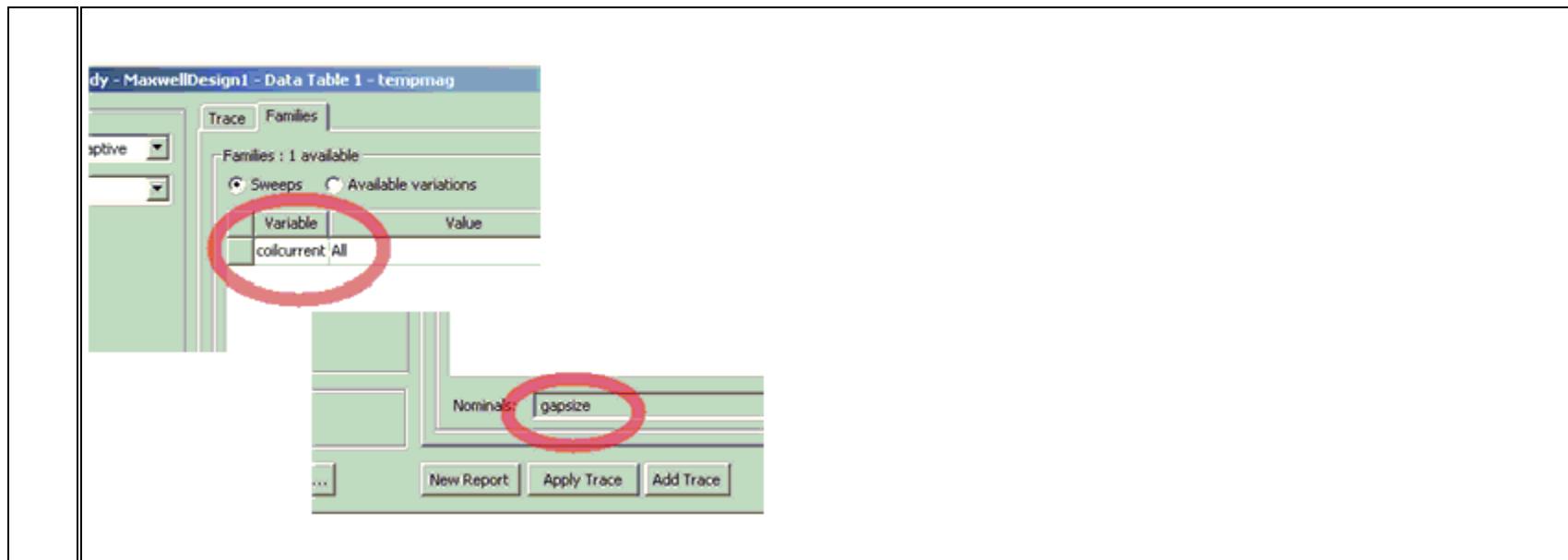
**Hint**

Variation variables can be identified by doing a **Data Table** report and viewing the variable specifications in the table header.  
For example:

tempmag coilcurrent='1000' gapsize='0.001' Setup1 : LastAdaptive		
	Freq [Hz]	
1	60.000000	1.538441

Also, as shown below, the **Trace** and **Families** tabs on the Report dialog box can be used as illustrated to identify variation variables.



**Example:**

```

Dim Val

Val= oModule.GetOutputVariableValue("tempmag", "freq='60' coilcurrent='1000' gapsize='-0.001'",
"Setup1 : LastAdaptive", "EddyCurrent", "")

Val= oModule.GetOutputVariableValue("tempmag", "coilcurrent='1000' gapsize='-0.001'", "Setup1 :
LastAdaptive", "Magnetostatic", "")

Val= oModule.GetOutputVariableValue("BFieldMag", "", "Setup1 : LastAdaptive", "Fields", "")

Val= oModule.GetOutputVariableValue("MagBatPoint1", "freq='60' coilcurrent='1000' gapsize='-
0.001' Phase='0deg'", "Setup1 : LastAdaptive", "Fields", Array("Context:", "Point1"))

```

# 12 - Reporter Editor Script Commands

Reporter commands should be executed by the oDesign object.

For example:

```
Set oDesign = Project.SetActiveDesign("MaxwellDesign1")
Set oModule = oDesign.GetModule("ReportSetup")
```

All Report and Trace properties can be edited using the **ChangeProperty** commands. This includes Title properties, General properties, and Background properties such as border color, fonts, X and Y axis scaling, and number display.

**Note:**

When you execute **Tools > Record Script**, operations performed in the Reporter are automatically recorded.

The list of commands is as follows:

[AddCartesianLimitLine](#)

[AddCartesianXMarker](#)

[AddCartesianYMarker](#)

[AddCartesianYMarkerToStack](#)

[AddDeltaMarker](#)

[AddMarker](#)

[AddNote](#)

[AddTraces](#)

[AddVerifEyeAnalysis](#)

[ClearAllMarkers](#)

[ClearAllTraceCharacteristics](#)

[CopyReportDefinitions](#)

[CopyReportData](#)

[CopyTraceDefinitions](#)

[CopyTracesData](#)

[CreateReport](#)

[CreateReportFromFile](#)

[CreateReportFromTemplate](#)

[CreateReportOfAllQuantities](#)

[DeleteMarker](#)

[DeleteAllReports](#)

[DeleteReports](#)

[DeleteTraces](#)

EditQuickEyeAnalysis

EditVerifEyeAnalysis

[FFTOnReport](#)

[ExportFieldsToFile](#)

[ExportImageToFile](#)

[ExportModelImageToFile](#)

[ExportModelMeshToFile](#)

[ExportToFile \[Reporter\]](#)

[ExportMarkerTable](#)

[GetAllCategories](#)

[GetAllQuantities](#)

[GetAllReportNames](#)

[GetAvailableDisplayTypes](#)

[GetAvailableReportTypes](#)

[GetAvailableSolutions](#)

[GetChildNames](#)

[GetChildObject](#)

[GetChildTypes](#)

[GetDisplayType](#)

[GetPropertyValue](#)

[GetReportTraceNames](#)

[GetSolutionContexts](#)

[GetSolutionDataPerVariation](#)

[GroupPlotCurvesByGroupingStrategy](#)

[ImportIntoReport](#)

[MovePlotCurvesToGroup](#)

[MovePlotCurvesToNewGroup](#)

[PasteReports](#)

[PasteTraces](#)  
[RenameReport](#)  
[RenameTrace](#)  
[ResetPlotSettings](#)  
[SavePlotSettingsAsDefault](#)  
[SetPropValue](#)  
[UpdateAllFieldsPlots](#)  
[UpdateAllReports](#)  
[UpdateQuantityFieldsPlots](#)  
[UpdateReports](#)  
[UpdateTraces](#)  
[UpdateTracesContextandSweeps](#)  
[UnGroupPlotCurvesInGroup](#)

## AddCartesianLimitLine

Adds a limit line to a report on the X axis.

UI Access	Report2D > Add Limit Line> Specify Points...		
Parameters	Name	Type	Description
	<ReportName>	String	Name of the report.

		<pre>         Array("NAME:XValues", &lt;integer X values&gt;),         "XUnits:=", &lt;string unit of measure for X&gt;,         Array("NAME:YValues", &lt;integer Y values&gt;),         "YUnits:=", "&lt;string unit of measure for Y&gt;",         "YAxis:=", &lt;string name of associated Y axis&gt;     ) </pre>
<b>Return Value</b>	None.	

<b>Python Syntax</b>	AddCartesianLimitLine (<ReportName>, <Def>)
<b>Python Example</b>	<pre> oModule.AddCartesianLimitLine ("Project Outputs",     [         "NAME:CartesianLimitLine",         [             ["NAME:XValues", 0, 2, 5, 7, 10, 15],             "XUnits:=", "s",             ["NAME:YValues", 0.05, 0.3, 0.65, 0.825, 0.95, 1],             "YUnits:=", "mV", "YAxis:=", "Y1"         ]     ] ) </pre>

<b>VB Syntax</b>	AddCartesianLimitLine <ReportName>, <Def>
<b>VB Example</b>	<pre> oModule.AddCartesianLimitLine "Project Outputs", </pre>

```
    Array("NAME:CartesianLimitLine",
          Array("NAME:XValues",0, 2, 5, 7, 10, 15),
          "XUnits:=", "s",
          Array("NAME:YValues",0.05, 0.3, 0.65, 0.825, 0.95, 1),
          "YUnits:=", "mV", "YAxis:=", "Y1"
        )
```

## AddCartesianXMarker

Adds a marker to a report on the X axis.

UI Access	Report2D > Marker > Add X Marker.		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<MarkerName>	String	Marker name, including any trailing number.
<XValue>	Double	X coordinate.	
Return Value	None		

Python Syntax	AddCartesianXMarker (<ReportName>, <MarkerName>, <XValue>)
Python Example	oModule.AddCartesianXMarker ("XY Plot 1", "MX1", 0)

<b>VB Syntax</b>	AddCartesianXMarker < <i>ReportName</i> >, < <i>MarkerName</i> >, < <i>XValue</i> >
<b>VB Example</b>	oModule.AddCartesianXMarker "XY Plot1", "MX1", 0

## AddCartesianYMarker

Adds a marker to a report on the Y axis.

UI Access	Report2D > Marker > Add Y Marker.		
<b>Parameters</b>	Name	Type	Description
	< <i>ReportName</i> >	String	Name of report.
	< <i>MarkerName</i> >	String	Marker name, including any trailing number.
	< <i>AxisName</i> >	String	Name of axis.
	< <i>YValue</i> >	Double	Y coordinate.
	< <i>CurveName</i> >	String	Name of curve.
<b>Return Value</b>	None		

<b>Python Syntax</b>	AddCartesianYMarker (< <i>ReportName</i> >, < <i>MarkerName</i> >, < <i>AxisName</i> >, < <i>YValue</i> >, < <i>CurveName</i> >)
<b>Python Example</b>	oModule.AddCartesianYMarker ("XY Plot 1", "MY1", "Y1", 0, "dB() : Setup1 : Sweep1")

<b>VB Syntax</b>	AddCartesianYMarker < <i>ReportName</i> >, < <i>MarkerName</i> >, < <i>AxisName</i> >, < <i>YValue</i> >, < <i>CurveName</i> >
<b>VB Example</b>	oModule.AddCartesianYMarker "XY Plot 1", "MY1", "Y1", 0, "dB() : Setup1 : Sweep1"

## AddDeltaMarker

Add markers to calculate differences between two trace points on a plot.

UI Access	Report2D > Marker > Add Delta Marker.		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<MarkerName1>	String	Marker name, including any trailing number, for the first marker.
	<CurveName1>	String	Full trace name for the first marker.
	<PrimarySweepValue1>	String	Frequency, including unit, for the first marker's sweep.
	<MarkerName2>	String	Marker name, including any trailing number, for the second marker.
	<CurveName2>	String	Full trace name for the second marker.
<PrimarySweepValue2>	String	Frequency, including unit, for the second marker's sweep.	
Return Value	None		

Python Syntax	AddDeltaMarker(<ReportName>, <MarkerName1>, <CurveName1>, <PrimarySweepValue1>, <MarkerName2>, <CurveName2>, <PrimarySweepValue2>)
Python Example	<pre>oModule.AddDeltaMarker("S Parameter Plot 1", "m1", "dB(S(Port1 Port1)) : Setup1 : Sweep1 : Cartesian", "8.45GHz", "m2", "dB(S(Port1 Port1)) : Setup1 : Sweep1 : Cartesian", "9.5GHz")</pre>

<b>VB Syntax</b>	AddDeltaMarker <ReportName>, <MarkerName1>, <CurveName1>, <PrimarySweepValue1>, <MarkerName2>, <CurveName2>, <PrimarySweepValue2>
<b>VB Example</b>	<pre> oModule.AddDeltaMarker "S Parameter Plot 1", "m1", "dB(S(Port1 Port1)) : Setup1 : Sweep1 : Cartesian", "8.45GHz", "m2", "dB(S(Port1 Port1)) : Setup1 : Sweep1 : Cartesian", "9.5GHz" </pre>

## AddMarker

Adds a marker to a trace on a report.

<b>UI Access</b>	Report2D > Marker > Add Marker.															
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ReportName&gt;</td> <td>String</td> <td>Name of report.</td> </tr> <tr> <td>&lt;MarkerName&gt;</td> <td>String</td> <td>Marker name, including any trailing number.</td> </tr> <tr> <td>&lt;CurveName&gt;</td> <td>String</td> <td>Full trace name.</td> </tr> <tr> <td>&lt;PrimarySweepValue&gt;</td> <td>String</td> <td>Primary sweep value, including unit.</td> </tr> </tbody> </table>	Name	Type	Description	<ReportName>	String	Name of report.	<MarkerName>	String	Marker name, including any trailing number.	<CurveName>	String	Full trace name.	<PrimarySweepValue>	String	Primary sweep value, including unit.
Name	Type	Description														
<ReportName>	String	Name of report.														
<MarkerName>	String	Marker name, including any trailing number.														
<CurveName>	String	Full trace name.														
<PrimarySweepValue>	String	Primary sweep value, including unit.														
<b>Return Value</b>	None															

<b>Python Syntax</b>	AddMarker( <ReportName>, <MarkerName>, <CurveName>, <PrimarySweepValue>)
<b>Python Example</b>	<pre> oModule.AddMarker("XY Plot 1", "m5", "GS1.VAL : TR4 : Cartesian", "3.6159999999997s") </pre>

<b>VB Syntax</b>	AddMarker <ReportName>, <MarkerName>, <CurveName>, <PrimarySweepValue>
<b>VB Example</b>	<pre>Set oModule = oDesign.GetModule("ReportSetup") oModule.AddMarker "XY Plot1", "m1", "mag(S(Port1 Port1)) : Setup1: LastAdaptive : Cartesian", "0.3in"</pre>

## AddNote

Adds a note at a specified location to a given report.

<b>UI Access</b>	Right-click on the plot and select <b>Add Note</b> .		
<b>Parameters</b>	Name <ReportName> <NotedataArray>  <NoteArray>	Type String Array  Array	Description Name of report Structured array.  Structured array:  "NAME:<StringDataName>", <NoteArray>  "SourceName:=", <string source name>,  "HaveDefaultPos:=", <boolean True for position 0,0; False to specify below>,  "DefaultXPos:=", <int X position for note; 0 for default>,  "DefaultYPos:=", <int Y position for note; 0 for default>,

		"String:=", <string note text>)
<b>Return Value</b>	None	

<b>Python Syntax</b>	AddNote (< <i>ReportName</i> >, < <i>NotedataArray</i> >)
<b>Python Example</b>	<pre> oModule.AddNote("XY Plot1",     [         "NAME:NoteDataSource",         "SourceName:=", "Note1",         "HaveDefaultPos:=", False,         "DefaultXPos:=", 1996,         "DefaultYPos:=", 3177,         "String:=", "This is a note."     ] ) </pre>

<b>VB Syntax</b>	AddNote < <i>ReportName</i> > < <i>NotedataArray</i> >
<b>VB Example</b>	<pre> oModule.AddNote "XY Plot1", _     Array("NAME:NoteDataSource", _         "SourceName:=", "Note1", _         "HaveDefaultPos:=", false, _         "DefaultXPos:=", 1996, _ ) </pre>

	<pre>"DefaultYPos:=", 3177, _ "String:=", "This is a note.")</pre>
--	--

## AddTraceCharacteristics

Adds a trace characteristics field to the legend on a report.

UI Access	Report2D > Trace Characteristics > All. This opens the <b>Add Trace Characteristics</b> dialog box.		
	Name	Type	Description
	<ReportName>	String	Name of report.
	<FunctionName>	String	The function name. See the <b>Functions</b> column of the <b>Add Trace Characteristics</b> dialog box.
<b>Parameters</b>	<FunctionArgs>	Array	<p>Array containing string values for any function arguments. Pass empty array if no arguments exist.</p> <p>To see which argument(s) a function takes, see the bottom of the <b>Add Trace Characteristics</b> dialog box.</p> <p>For function with one argument:</p> <pre>Array(&lt;value&gt;)</pre> <p>For function with multiple arguments:</p> <pre>Array(&lt;value&gt;, &lt;value&gt;, ...)</pre>
	<RangeArgs>	Array	<p>Required. Array containing either string "Full" for a full sweep range, or "Specified" and strings containing the start and end values for the frequency range.</p> <p>For example:</p> <pre>Array("Specified", "19.5GHz", "24.4GHz")</pre>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	AddTraceCharacteristics (<ReportName>, <FunctionName>, <FunctionArgs>, <RangeArgs>)
<b>Python Example</b>	<pre> oModule.AddTraceCharacteristics("XY Plot 2", "delaytime", ["0"], ["Full"]) oModule.AddTraceCharacteristics("Differential S-parameters", "prms", ["0", "0"], ["Full"]) oModule.AddTraceCharacteristics("Rept2DRectFreq", "distortion", ["0"], ["Specified", "19.5GHz", "20.4GHz"]) </pre>

<b>VB Syntax</b>	AddTraceCharacteristics <ReportName>, <FunctionName>, <FunctionArgs>, <RangeArgs>
<b>VB Example</b>	<pre> oModule.AddTraceCharacteristics "XY Plot 2", "delaytime", Array("0"), Array("Full") oModule.AddTraceCharacteristics "Differential S-parameters", "prms", Array("0", "0"), Array("Full") oModule.AddTraceCharacteristics "Rept2DRectFreq", "distortion", Array("0"), Array( "Specified", "19.5GHz", "20.4GHz") </pre>

## AddTraces

Creates a new trace and adds it to the specified report.

UI Access	Modify Report > Add Trace.		
Parameters	Name	Type	Description
	<ReportName>	String	Name of Report
	<SolutionName>	String	Name of the solution as listed in the <b>Modify Report</b> dialog box.

	<code>&lt;ContextArray&gt;</code>	Array	<p>Context for which the expression is being evaluated. This can be an empty string if there is no context.</p> <pre>Array("Domain:=", &lt;DomainType&gt;) &lt;DomainType&gt; ex. "Sweep" or "Time" Array("Context:=", &lt;GeometryType&gt;) &lt;GeometryType&gt; ex. "Infinite Spheren", "Spheren", "Polylinen"</pre>
	<code>&lt;FamiliesArray&gt;</code>	Array	<p>Contains sweep definitions for the report.</p> <pre>Array("&lt;VariableName&gt;:= ", &lt;ValueArray&gt;) &lt;ValueArray&gt; Array("All") or Array("Value1", "Value2", ..."Valuen") examples of &lt;VariableName&gt; "Freq", "Theta", "Distance"</pre>
	<code>&lt;ReportdataArray&gt;</code>	Array	<p>This array contains the report quantity and X, Y, and (Z) axis definitions.</p> <pre>Array("X Component:=", &lt;VariableName&gt;, "Y Component:=", &lt;VariableName&gt;   &lt;ReportQuantityArray&gt;) &lt;ReportQuantityArray&gt; ex. Array("dB(S(Port1, Port1))")</pre>
<b>Return Value</b>	None		

<b>Python Syntax</b>	Add Traces(<ReportName>, <SolutionName>, <ContextArray>, <FamiliesArray>, <ReportdataArray>)
<b>Python Example</b>	<pre> oModule.AddTraces("XY Plot1", "Setup1 : Sweep1", ["Domain:=", "Time", "HoldTime:=", 1, "RiseTime:=", 0, "StepTime:=", 6.24999373748E-012, "Step:=", False, "WindowWidth:=", 1, "WindowType:=", 0, "KaiserParameter:=", 1, "MaximumTime:=", 6.2437437437444E-009], ["Time:=", ["All"], "OverridingValues:=", ["0s", "6.24999373748188e-012s", ...]], ["X Component:=", "Time", "Y Component:=", ["TDRZ(WavePort1)"]], []) </pre>

<b>VB Syntax</b>	Add Traces <ReportName> <SolutionName> <ContextArray> <FamiliesArray> <ReportdataArray>
<b>VB Example</b>	<pre> oModule.AddTraces "XY Plot1", "Setup1 : Sweep1", _ Array("Domain:=", "Time", "HoldTime:=", 1, "RiseTime:=", 0, _ "StepTime:=", 6.24999373748E-012, "Step:=", false, _ "WindowWidth:=", 1, _ "WindowType:=", 0, "KaiserParameter:=", 1, _ "MaximumTime:=", 6.2437437437444E-009), _ Array("Time:=", Array("All"), "OverridingValues:=", _ Array("0s", "6.24999373748188e-012s", ...)), _ Array("X Component:=", "Time", _ </pre>

```
"Y Component:=", Array("TDRZ(WavePort1)"), _  
Array()
```

## ChangeProperty[ReportSetup]

Change the properties for a Report.

<b>UI Access</b>	Double-click on a report to change its properties.		
<b>Parameters</b>	Name <i>&lt;PropertyArray&gt;</i>	Type Array	Description Varies, depending on the properties associated with the select object command.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ChangeProperty( <i>&lt;PropertyArray&gt;</i> )
<b>Python Example</b>	<pre>oModule = oDesign.GetModule("ReportSetup") oModule.ChangeProperty( [     "NAME:AllTabs",     [         "NAME:General",         [             "NAME:PropServers",             [                 "NAME:FE_Solver"             ]         ]     ] ])</pre>

```
"XY Plot 1:General"
],
[
"NAME:ChangedProps",
[
"NAME:Use Scientific Notation",
"Value:=", True
]
]
]
])

oModule = oDesign.GetModule("ReportSetup")
oModule.ChangeProperty(
[
"NAME:AllTabs",
[
"NAME:Legend",
[
"NAME:PropServers",
"S Parameter Plot 1:Legend"
```

```
        ],
        [
        "NAME:ChangedProps",
        [
        "NAME:Legend Name",
        "Value:=" , "Ansys"
        ]
    ],
    [
    "NAME:AllTabs",
    [
    "NAME:General",
    [
    "NAME:PropServers",
    "S Parameter Plot 1:General"
    ],
    [

```

```

        "NAME:ChangedProps",
        [
            "NAME:Auto Scale Fonts",
            "Value:=" , False
        ]
    ]
)

```

VB Syntax	ChangeProperty <PropertyArray>
VB Example	<pre> oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Cartesian", Array ("NAME:PropServers", _ "XY Plot 1:CartesianDisplayTypeProperty"), Array("NAME:ChangedProps", Array("NAME&gt;Show X Scrollbar", "Value:=", _ true)))))  oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:General", Array("NAME:PropServ- ers", _ "XY Plot 1:General"), Array("NAME:ChangedProps", Array("NAME:Use Scientific Notation", "Value:=", _ true)))))  </pre>

## ClearAllMarkers

Clears all markers from a report.

<b>UI Access</b>	<b>Report2D &gt; Markers &gt; Clear All.</b>						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;ReportName&gt;</td><td>String</td><td>Name of Report</td></tr></table>	Name	Type	Description	<ReportName>	String	Name of Report
Name	Type	Description					
<ReportName>	String	Name of Report					
<b>Return Value</b>	None						

<b>Python Syntax</b>	ClearAllMarkers(<ReportName>)
<b>Python Example</b>	oModule.ClearAllMarkers("XY Plot 1")

<b>VB Syntax</b>	ClearAllMarkers <ReportName>
<b>VB Example</b>	oModule.ClearAllMarkers "XY Plot 1"

## ClearAllTraceCharacteristics

Clears all trace characteristics from the legend in a report.

<b>UI Access</b>	<b>Report2D &gt; Trace Characteristics &gt; Clear All.</b>						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;PlotName&gt;</td><td>String</td><td>Name of the plot</td></tr></table>	Name	Type	Description	<PlotName>	String	Name of the plot
Name	Type	Description					
<PlotName>	String	Name of the plot					

<b>Return Value</b>	None
---------------------	------

<b>Python Syntax</b>	ClearAllTraceCharacteristics(<PlotName>)
<b>Python Example</b>	<code>oModule.ClearAllTraceCharacteristics("XY Plot 1")</code>

<b>VB Syntax</b>	ClearAllTraceCharacteristics <PlotName>
<b>VB Example</b>	<code>oModule.ClearAllTraceCharacteristics "XY Plot 1"</code>

## CopyTracesData

Copies trace data for a paste operation.

<b>UI Access</b>	Select a trace in the Project tree, right-click and select <b>Copy Data</b> .									
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;ReportName&gt;</td> <td>String</td> <td>Name of Report.</td> </tr> <tr> <td>&lt;TraceArray&gt;</td> <td>Array</td> <td>Trace definitions from which to copy corresponding data.</td> </tr> </table>	Name	Type	Description	<ReportName>	String	Name of Report.	<TraceArray>	Array	Trace definitions from which to copy corresponding data.
Name	Type	Description								
<ReportName>	String	Name of Report.								
<TraceArray>	Array	Trace definitions from which to copy corresponding data.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	CopyTracesData(<ReportName>, <TracesArray>)
<b>Python Example</b>	<code>oModule.CopyTracesData ("Transmission", ["mag(S(Port1,Port2))"])</code>

<b>VB Syntax</b>	CopyTracesData < <i>ReportName</i> >, < <i>TracesArray</i> >
<b>VB Example</b>	<pre>oModule.CopyTracesData _     "C11", Array("mag(S(Port1,Port2))")</pre>

## CopyReportsData

Copy all data corresponding to the specified reports.

<b>UI Access</b>	Select a report in the Project tree, right-click and select <b>Copy Data</b>		
<b>Parameters</b>	Name < <i>ReportsArray</i> >	Type Array	Description Names of reports from which to copy data
<b>Return Value</b>	None		

<b>Python Syntax</b>	CopyReportsData (< <i>ReportsArray</i> >)
<b>Python Example</b>	<pre>oModule.CopyReportsData ([ "Transmission", "Reflection" ])</pre>

<b>VB Syntax</b>	CopyReportsData < <i>ReportsArray</i> >
<b>VB Example</b>	<pre>oModule.CopyReportsData _     Array("Transmission", "Reflection")</pre>

## CopyReportDefinitions

Copy the definition of a report for paste operations.

<b>UI Access</b>	Select a report in the Project tree, right-click and select <b>Copy Definition</b>		
<b>Parameters</b>	Name <i>&lt;ReportsArray&gt;</i>	Type Array	Description Names of reports from which to copy the definitions
<b>Return Value</b>	None		

<b>Python Syntax</b>	CopyReportDefinitions( <i>&lt;ReportsArray&gt;</i> )
<b>Python Example</b>	<code>oModule.CopyReportDefinitions(["Transmission", "Reflection"])</code>

<b>VB Syntax</b>	CopyReportDefinitions <i>&lt;ReportsArray&gt;</i>
<b>VB Example</b>	<code>oModule.CopyReportDefinitionsv _</code> <code>Array("Transmission", "Reflection")</code>

## CopyTraceDefinitions

Copy trace definitions for a paste operation.

<b>UI Access</b>	Select a trace in the Project tree, right-click and select <b>Copy Definition</b>		
<b>Parameters</b>	Name <i>&lt;ReportName&gt;</i>	Type String	Description Name of Report

	<TracesArray>	Array	Trace definitions to copy
Return Value	None		

Python Syntax	CopyTraceDefinitions(<ReportName>, <TracesArray>)
Python Example	<pre>oModule.CopyTraceDefinitions ("Transmission", ["mag(S(Port1,Port2))"])</pre>

VB Syntax	CopyTraceDefinitions <ReportName>, <TracesArray>
VB Example	<pre>oModule.CopyTraceDefinitions "Transmission", Array("mag(S(Port1,Port2))")</pre>

## CreateReportFromFile

Creates a new report from an .rdat file.

UI Access	Right-click on <b>Results</b> > <b>Create Report From File...</b>						
Parameters	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;FilePathName&gt;</td><td>String</td><td>Path to .rdat file.</td></tr></table>	Name	Type	Description	<FilePathName>	String	Path to .rdat file.
Name	Type	Description					
<FilePathName>	String	Path to .rdat file.					
Return Value	None.						

<b>Python Syntax</b>	CreateReportFromFile(<FilePathName>)
<b>Python Example</b>	oModule.CreateReportFromFile ("C:/Users/MyDir/Documents/Return_Loss.rdat")

<b>VB Syntax</b>	CreateReportFromFile <FilePathName>
<b>VB Example</b>	oModule.CreateReportFromFile "C:/Users/MyDir/Documents/Return_Loss.rdat"

## CreateReport (Maxwell)

**Use:** Creates a new report with a single trace and adds it to the **Results** branch in the project tree. To add more traces, use the **AddTraces** command. To edit the display properties, use the **ChangeProperty** Script command.

**Command:** Maxwell2D or Maxwell3D>Results>Create<type> Report

**Syntax:** CreateReport <ReportName> <ReportType> <DisplayType> <SolutionName> <ContextArray> <FamiliesArray><ReportdataArray>

**Return Value:** None

**Parameters:** <ReportName>

Type: <string>

Name of Report.

ReportType

*Some possible values are:*

"TransientAPhiFormulation"

"Fields"

"Noise Vibration"

"Time Averaged Fields"

"Matrix".  
"DC R/L Fields".  
"AC R/L Fields".  
"C Fields"  
"Harmonic Force"

DisplayType  
"Rectangular Plot", "Rectangular Stacked Plot", "Data Table", "3D Rectangular Plot", "Rectangular Contour Plot".

<TraceArray>  
    Array("NAME:Traces",  
          <OneTraceArray>, <OneTraceArray>, ...)

<OneTraceArray>  
    Array("NAME:<TraceName>,"  
          "SolutionName:=","string",  
          "Context:=","string",  
          <DisplayTypeDependentData>)

<SolutionName>  
    *Name of the solution as listed in the **Traces** dialog box.*  
    For example: "Setup1 : Last Adaptive"

<Context> Array  
    *Context for which the output variable expression is being evaluated.*

```
Array("Domain:=", <DomainType>
<DomainType>
ex. "Sweep" or "Time"
This can be an empty string if there is no context.
Example: "Line1" or ""
Field reports usually require a polyline (e.g. "Line1") unless they are integrations..
<DisplayTypeDependentData>
This data varies according to the display type. See the examples below.
<FamiliesArray>
Type: Array of strings
Contains sweep definitions for the report.
Array("<VariableName>:= ", <ValueArray>
<ValueArray>
Array("All") or Array("Value1", "Value2", ..."Valuen")
examples of <VariableName>
"Freq", "Theta", "Distance"
<ReportdataArray>
Type: Array of strings
This array contains the report quantity and X, Y, and (Z) axis definitions.
Array("X Component:=", <VariableName>, "Y Component:=", <VariableName> | <ReportQuantityArray>
<ReportQuantityArray>
```

ex. `Array("dB(S(Port1, Port1))")`

**Example:** `oDesign.CreateReport Array("NAME:Rept2DRectTime", _  
"ReportType:=", "Matrix",  
"DisplayType:=", "Rectangular Plot",  
Array("NAME:Traces",  
Array("NAME:Trace1",  
"SolutionName:=",  
"Setup1 : Adaptive_2",  
"Context:=", "Original",  
"XComponent:=", "Pass",  
"YComponent:=", "C(Box1, Box1)",  
"YAxis:=", 1)))`

**Python Example:** `oModule.CreateReport("Loss Plot 1", "Transient", "Rectangular Plot", "Setup1 : Transient", [  
"Domain:=" , "Average and RMS",  
"npoles:=" , "2",  
"alignment:=" , "0deg",  
"orientation:=" , "CounterClockwise",`

---

```
"losses:=" , "0W",
"speed:=" , "0rpm",
"Resistance(PhA):=" , "0ohm",
"Resistance(PhB):=" , "0ohm",
"Resistance(PhC):=" , "0ohm",
"Inductance(PhA):=" , "0H",
"Inductance(PhB):=" , "0H",
"Inductance(PhC):=" , "0H"
],
[
"Im:=" , ["All"],
"Gamma:=" , ["Nominal"],
"speed_rpm:=" , ["Nominal"],
"poles:=" , ["Nominal"]
],
[
"X Component:=" , "Im",
"Y Component:=" , ["AvgCoreLoss"]
], [])
```

```
Python Example for Transient A-phi Formulation: oModule.CreateReport("Loss Plot 1", "TransientAPhiFormulation", "Rectangular Plot", "Setup1 : Transient",
[
"Domain:=" , "Sweep"
],
[
"Time:=" , ["All"],
"$HcRef:=" , ["Nominal"],
"$MagnetTemperature:=" , ["Nominal"]
],
[
"X Component:=" , "Time",
"Y Component:=" , ["TerminalVoltage(Voltage1)", "TerminalVoltage(Voltage2)"]
])
```

```
[Beta] Python Example for Transient A-phi Formulation, LayoutForce plot: oModule.CreateReport
("LayoutForce Plot 1", "TransientAPhiFormulation", "Rectangular Plot", "Setup1 : Transient",
[
"Domain:=" , "Sweep"
],
[
```

```
"Time:=" , ["All"]  
],  
[  
"X Component:=" , "Time",  
"Y Component:=" , ["LayoutForcel.Force_mag"]  
)  
```
```

***Python Example for Average\_Surface\_Loss\_Density:***

```
oModule.CreateReport("Calculator Expressions Plot 1", "Time Averaged Fields", "Rectangular  
Plot", "Setup1 : Transient",  
[  
"Context:=" , "Point1",  
"PointCount:=" , 1001,  
"Time0:=" , "0s"  
,  
[  
"Time:=" , ["All"]  
,  
[  
"X Component:=" , "Time",  
"Y Component:=" , ["Average_Surface_Loss_Density"]
```

])

***Python Example for Transient and Eddy Current Per-Winding Losses:***

```
oModule.CreateReport("Loss Plot 4", "Transient", "Rectangular Plot", "Setup1 : Transient",
[
"Domain:=" , "Sweep"
],
[
"Time:=" , ["All"]
],
[
"X Component:=" , "Time",
"Y Component:=" , ["PerWindingSolidLoss(WindingV1_Solid)"]
])
oModule.AddTraces("Loss Plot 4", "Setup1 : Transient",
[
"Domain:=" , "Sweep"
],
[
"Time:=" , ["All"]]
```

```
],
[
"X Component:=" , "Time",
"Y Component:=" , ["PerWindingStrandedLoss(WindingI3_Litz)"]
])
oModule.AddTraces("Loss Plot 4", "Setup1 : Transient",
[
"Domain:=" , "Sweep"
],
[
"Time:=" , ["All"]
],
[
"X Component:=" , "Time",
"Y Component:=" , ["PerWindingStrandedLossAC(WindingI3_Litz)"]
])
```

#### ***Python Example for Harmonic Force Report Type:***

```
oModule.CreateReport("Harmonic Force Plot 5", "Harmonic Force", "3D Rectangular Bar Plot",
"Setup1 : Transient",
[
"Freq:=" , ["All"],
```

```
"SpatialOrder:=" , ["All"],  
"fractions:=" , ["Nominal"],  
"halfAxial:=" , ["Nominal"],  
"endRegion:=" , ["Nominal"],  
"RPM:=" , ["1500rpm"],  
"ppairs:=" , ["Nominal"]  
,  
[  
"X Component:=" , "Freq",  
"Y Component:=" , "SpatialOrder",  
"Z Component:=" , ["re(HarmonicForce_a)"]  
)
```

***Python Example for Thin Layer (Air Gap) Report Type:***

```
oModule.CreateReport("Magnetic Energy Table 1", "Magnetostatic", "Data Table", "Setup1 : LastAdaptive", [],  
[  
"$Current:=" , ["All"]  
,  
[  
"X Component:=" , "$Current",
```

```
"Y Component:=" , ["MagneticEnergy(ThinLayer1)"]
])
```

**VB Example for Thin Layer (Air Gap) Report Type:**

```
oModule.CreateReport "Magnetic Energy Table 1", "Magnetostatic", "Data Table" "Setup1 : LastAdaptive", Array(), Array("$Current:=", Array("All")), Array("X Component:=", "$Current", "Y Component:=", Array("MagneticEnergy(ThinLayer1)"))
```

## CreateReportFromTemplate

Creates a report from a saved template.

|                     |                                                                                                                                                                                   |                         |      |             |                |        |                         |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|------|-------------|----------------|--------|-------------------------|
| <b>UI Access</b>    | [product] > Results > Report Templates > PersonalLib > [TemplateName]                                                                                                             |                         |      |             |                |        |                         |
| <b>Parameters</b>   | <table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;TemplatePath&gt;</td> <td>String</td> <td>Path to report template</td> </tr> </table> | Name                    | Type | Description | <TemplatePath> | String | Path to report template |
| Name                | Type                                                                                                                                                                              | Description             |      |             |                |        |                         |
| <TemplatePath>      | String                                                                                                                                                                            | Path to report template |      |             |                |        |                         |
| <b>Return Value</b> | None                                                                                                                                                                              |                         |      |             |                |        |                         |

|                       |                                                                                                                |
|-----------------------|----------------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | CreateReportFromTemplate( <TemplatePath>)                                                                      |
| <b>Python Example</b> | <pre>oModule.CreateReportFromTemplate( "C:/MyHFSSProjects/PersonalLib/ReportTemplates/TestTemplate.rpt")</pre> |

|                   |                                                                                |
|-------------------|--------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | CreateReportFromTemplate <TemplatePath>                                        |
| <b>VB Example</b> | <pre>oModule.CreateReportFromTemplate "C:/MyHFSSProjects/PersonalLib/" _</pre> |

|  |                                    |
|--|------------------------------------|
|  | "ReportTemplates/TestTemplate.rpt" |
|--|------------------------------------|

## CreateReportOfAllQuantities

Creates a report including all quantities in a category. Cannot create a report with expressions.

|                     |                            |        |                                                                 |
|---------------------|----------------------------|--------|-----------------------------------------------------------------|
| <b>UI Access</b>    | NA                         |        |                                                                 |
| <b>Parameters</b>   | Name                       | Type   | Description                                                     |
|                     | <ReportType>               | String | Report type name as input parameter                             |
|                     | <DisplayType>              | String | Display type name as input parameter                            |
|                     | <SolutionName>             | String | Solution name as input parameter                                |
|                     | <SimValueCtx>              | String | A context name, or array of string that encoded the context(I). |
|                     | <CategoryName>             | String | Category name as input parameter                                |
|                     | <PointSet>                 | Array  | Array of strings(II)                                            |
|                     | <CommonComponentsOfTraces> | Array  | Array of strings(III)                                           |
|                     | <ExtTraceInfo>             | Array  | Array of strings(IV)                                            |
| <b>Return Value</b> | None.                      |        |                                                                 |

|                       |                                                                                                                                                                                                        |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | CreateReportOfAllQuantities(<ReportNameArg>, <ReportType>, <DisplayType>, <SolutionName>, <SimValueCtx>, <CategoryName>, <PointSet>, <CommonComponentsOfTraces>, <ExtTraceInfo>)                       |
| <b>Python Example</b> | <pre> oModule.CreateReportOfAllQuantities("Smith Chart all", "Modal Solution Data", "Smith Chart", "Setup1 : LastAdaptive", [], "S Parameter", ["Freq:=", ["All"]], "offset:=", ["All"], "a:=", </pre> |

|  |                                                           |
|--|-----------------------------------------------------------|
|  | <code>["Nominal"], "b:=", [ "Nominal" ]], [], [] )</code> |
|--|-----------------------------------------------------------|

|                   |                                                                                                                                                                                                                                                                                                 |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | <code>CreateReportOfAllQuantities &lt;ReportNameArg&gt;, &lt;ReportType&gt;, &lt;DisplayType&gt;, &lt;SolutionName&gt;, &lt;SimValueCtxt&gt;, &lt;CategoryName&gt;, &lt;PointSet&gt;, &lt;CommonComponentsOfTraces&gt;, &lt;ExtTraceInfo&gt;</code>                                             |
| <b>VB Example</b> | <pre>oModule.CreateReportOfAllQuantities "Smith Chart all", _ "Modal Solution Data", "Smith Chart", "Setup1 : LastAdaptive", _ Array(), "S Parameter", _ Array("Freq:=", Array("All")), "offset:=", Array("All"), _ "a:=", Array("Nominal"), "b:=", Array("Nominal")), _ Array(), Array()</pre> |

## DeleteMarker

*Use:* Deletes the specified marker.

|                                 |                                                                                                                                                                                          |                     |      |             |                                 |        |                     |
|---------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|------|-------------|---------------------------------|--------|---------------------|
| <b>UI Access</b>                | [product] > <b>Fields</b> > <b>Fields</b> > <b>Marker</b> > <b>Delete Marker</b> .                                                                                                       |                     |      |             |                                 |        |                     |
| <b>Parameters</b>               | <table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><code>&lt;MarkerName&gt;</code></td> <td>String</td> <td>Name of the marker.</td> </tr> </table> | Name                | Type | Description | <code>&lt;MarkerName&gt;</code> | String | Name of the marker. |
| Name                            | Type                                                                                                                                                                                     | Description         |      |             |                                 |        |                     |
| <code>&lt;MarkerName&gt;</code> | String                                                                                                                                                                                   | Name of the marker. |      |             |                                 |        |                     |
| <b>Return Value</b>             | None.                                                                                                                                                                                    |                     |      |             |                                 |        |                     |

|                       |                                          |
|-----------------------|------------------------------------------|
| <b>Python Syntax</b>  | DeleteMarker(<MarkerName>)               |
| <b>Python Example</b> | <code>oModule.DeleteMarker ("m1")</code> |

|                   |                                        |
|-------------------|----------------------------------------|
| <b>VB Syntax</b>  | DeleteMarker <MarkerName>              |
| <b>VB Example</b> | <code>oModule.DeleteMarker "m1"</code> |

## DeleteAllReports

Deletes all existing reports.

|                     |                                                                                                                      |
|---------------------|----------------------------------------------------------------------------------------------------------------------|
| <b>UI Access</b>    | Right-click the report to delete in the project tree, and then click <b>Delete All Reports</b> on the shortcut menu. |
| <b>Parameters</b>   | None.                                                                                                                |
| <b>Return Value</b> | None.                                                                                                                |

|                       |                                          |
|-----------------------|------------------------------------------|
| <b>Python Syntax</b>  | <code>DeleteAllReports()</code>          |
| <b>Python Example</b> | <code>oModule.DeleteAllReports ()</code> |

|                   |                                       |
|-------------------|---------------------------------------|
| <b>VB Syntax</b>  | <code>DeleteAllReports</code>         |
| <b>VB Example</b> | <code>oModule.DeleteAllReports</code> |

## DeleteReports

Deletes an existing report or reports.

|                     |                                                                                                         |               |                                                    |
|---------------------|---------------------------------------------------------------------------------------------------------|---------------|----------------------------------------------------|
| <b>UI Access</b>    | Right-click the report to delete in the project tree, and then click <b>Delete</b> on the shortcut menu |               |                                                    |
| <b>Parameters</b>   | Name<br><i>&lt;ReportNameArray&gt;</i>                                                                  | Type<br>Array | Description<br>Array of report names to be deleted |
| <b>Return Value</b> | None.                                                                                                   |               |                                                    |

|                       |                                                           |
|-----------------------|-----------------------------------------------------------|
| <b>Python Syntax</b>  | DeleteReports( <i>&lt;ReportNameArray&gt;</i> )           |
| <b>Python Example</b> | <code>oModule.DeleteReports ([ "Rept2DRectFreq" ])</code> |

|                   |                                                            |
|-------------------|------------------------------------------------------------|
| <b>VB Syntax</b>  | DeleteReports <i>&lt;ReportNameArray&gt;</i>               |
| <b>VB Example</b> | <code>oModule.DeleteReports Array("Rept2DRectFreq")</code> |

## DeleteTraces

Deletes an existing traces or traces.

|                   |                                                                                                        |               |                                                                                                                                                       |
|-------------------|--------------------------------------------------------------------------------------------------------|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>UI Access</b>  | Right-click the Trace to delete in the project tree, and then click <b>Delete</b> on the shortcut menu |               |                                                                                                                                                       |
| <b>Parameters</b> | Name<br><i>&lt;TraceSelection&gt;</i>                                                                  | Type<br>Array | Description<br>Structured array define selections.<br><br><code>Array ("&lt;ReportName&gt;:=", &lt;TracesArray&gt;, &lt;TracesArray&gt;, ... )</code> |

|                     |                                  |        |                                                                                                                             |
|---------------------|----------------------------------|--------|-----------------------------------------------------------------------------------------------------------------------------|
|                     | <code>&lt;ReportName&gt;</code>  | String | Name of Report                                                                                                              |
|                     | <code>&lt;TracesArray&gt;</code> | Array  | Contains the traces to delete within a report<br><br>Array ( <code>&lt;Trace&gt;</code> , <code>&lt;Trace&gt;</code> , ...) |
|                     | <code>&lt;Trace&gt;</code>       | String | A specific trace that the user wishes to delete                                                                             |
| <b>Return Value</b> | None.                            |        |                                                                                                                             |

|                       |                                                                                                                              |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | <code>DeleteTraces(&lt;TraceSelection&gt;)</code>                                                                            |
| <b>Python Example</b> | <code>oModule.DeleteTraces ([ "XY Plot 1:=",<br/>[ "dB (S (LumpPort1,LumpPort1)) "],<br/>"XY Plot 2:=", [ "Mag_E" ]])</code> |

|                   |                                                                                                                                          |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | <code>DeleteTraces &lt;TraceSelection&gt;</code>                                                                                         |
| <b>VB Example</b> | <code>oModule.DeleteTraces Array("XY Plot 1:=", _<br/>Array("dB (S (LumpPort1,LumpPort1))"), _<br/>"XY Plot 2:=", Array("Mag_E"))</code> |

## ExportImageToFile [Reporter]

Exports a report image in a specified format. In Release 23.1, this command is fully supports -ng (non-graphical) mode.

|                  |     |
|------------------|-----|
| <b>UI Access</b> | N/A |
|------------------|-----|

| <b>Parameters</b>   | Name         | Type    | Description                                                                                                                                |
|---------------------|--------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------|
|                     | <ReportName> | String  | Name of report to be exported.                                                                                                             |
|                     | <FileName>   | String  | Full path of the exported image file name; with extension of jpg, gif, tiff, tif, bmp, or wrl.                                             |
|                     | <Width>      | Integer | Image width in pixels; if width or height is less or equal to zero, use the report window width, or 500 pixels if report window is closed. |
|                     | <Height>     | Integer | Image height in pixels; if width or height is less or equal to zero, use the report window height, or 500 pixels report window is closed.  |
| <b>Return Value</b> | None.        |         |                                                                                                                                            |

|                       |                                                                                                                           |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | ExportImageToFile(<ReportName>, <FileName>, <Width>, <Height>)                                                            |
| <b>Python Example</b> | <pre> oModule.ExportImageToFile(     "Rectangular Contour Plot 1",     "D:/work/2015/UV-Export/pp1.gif",     0, 0) </pre> |

|                   |                                                                                                                         |
|-------------------|-------------------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | ExportImageToFile <ReportName>, <FileName>, <Width>, <Height>                                                           |
| <b>VB Example</b> | <pre> oModule.ExportImageToFile _     "Rectangular Contour Plot 1", _     "D:/work/2015/UV-Export/pp1.gif", 0, 0 </pre> |

## ExportPlotImageToFile

Deprecated. Use [ExportPlotImageWithViewToFile](#).

Creates field plot exports of existing field plots from a given view points, and with the model being auto-sized automatically for each view.

|                     |                                |        |                                                                         |
|---------------------|--------------------------------|--------|-------------------------------------------------------------------------|
| <b>UI Access</b>    | N/A                            |        |                                                                         |
| <b>Parameters</b>   | Name                           | Type   | Description                                                             |
|                     | <FileName>                     | String | Full path plus file name.                                               |
|                     | <FolderName>                   | String | Plot folder name.                                                       |
|                     | <ItemName>                     | String | Name of fields to plot.                                                 |
|                     | <SetViewTopDownDirectionByRCS> | String | Optional. Name of relative coordinate system to use for the field plot. |
| <b>Return Value</b> | None.                          |        |                                                                         |

|                       |                                                                                                              |
|-----------------------|--------------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | ExportPlotImageToFile(<FileName>, <FolderName>, <ItemName>, <SetViewTopDownDirectionByRCS>)                  |
| <b>Python Example</b> | <pre>oModule.ExportPlotImageToFile(<br/>    "C:\\TestEPITF2.jpg", "",<br/>    "Mag_E2", "RelativeCS1")</pre> |

|                   |                                                                                                                |
|-------------------|----------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | ExportPlotImageToFile <FileName>, <FolderName>, <ItemName>, <SetViewTopDownDirectionByRCS>                     |
| <b>VB Example</b> | <pre>oModule.ExportPlotImageToFile _<br/>    "C:\\TestEPITF2.jpg", "", _<br/>    "Mag_E2", "RelativeCS1"</pre> |

## ExportReport

**Note:**

The ExportReport script command has been replaced by the script command [ExportToFile](#). ExportReport remains in order to retain backward compatibility for existing scripts, but it is strongly recommended that you now use [ExportToFile](#).

Export a report to a data file.

*Command:* None

*Syntax:* ExportReport <ReportName>, <FileName>, <FileExtension>

*Return Value:* None

*Parameters:* <ReportName>

Type: string

<Filename>

Type: string

<FileExtension>

Type: string

*VB Example:*

```
Dim oAnsoftApp  
Dim oDesktop  
Dim oProject  
Dim oDesign  
Dim oEditor
```

```
Dim oModule

Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow

Set oProject = oDesktop.SetActiveProject("BJTinverter")
Set oDesign = oProject.SetActiveDesign("Nexxim1")
oDesign.ExportReport "Data Table 1", "table_test", "csv"
```

|                       |                                                                 |
|-----------------------|-----------------------------------------------------------------|
| <b>Python Syntax</b>  | ExportReport( <ReportName>, <FileName>)                         |
| <b>Python Example</b> | <pre>oDesign.ExportReport(     "Plot1", "c:\report1.dat")</pre> |

## ExportToFile

### Note:

The ExportToFile script command has replaced the script command [ExportReport](#). ExportReport remains in order to retain backward compatibility for existing scripts, but it is strongly recommended that you now use ExportToFile.

From a data table or plot, generates text format, comma delimited, tab delimited, or .dat type output files.

| <b>UI Access</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | Right-click on report name in the project tree and select <b>Export Data</b> . |                                                                                                                                                                                                   |      |             |              |        |                                |            |        |                                                                                                                                                                                                   |            |        |                            |                        |         |                 |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|-------------|--------------|--------|--------------------------------|------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|--------|----------------------------|------------------------|---------|-----------------|
| <b>Parameters</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                                                                                |                                                                                                                                                                                                   |      |             |              |        |                                |            |        |                                                                                                                                                                                                   |            |        |                            |                        |         |                 |
| <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ReportName&gt;</td> <td>String</td> <td>Name of report to be exported.</td> </tr> <tr> <td>&lt;FileName&gt;</td> <td>String</td> <td>Full path of the exported image file name; with extension of<br/>.txt - Post processor format file<br/>.csv - Comma-delimited data file<br/>.tab - Tab-separated file<br/>.dat - Ansys plot data file</td> </tr> <tr> <td>&lt;UnitSpec&gt;</td> <td>String</td> <td>For example, "kV, Mhz, yd"</td> </tr> <tr> <td>&lt;UseTraceNumberFormat&gt;</td> <td>Boolean</td> <td>"True", "False"</td> </tr> </tbody> </table> |                                                                                | Name                                                                                                                                                                                              | Type | Description | <ReportName> | String | Name of report to be exported. | <FileName> | String | Full path of the exported image file name; with extension of<br>.txt - Post processor format file<br>.csv - Comma-delimited data file<br>.tab - Tab-separated file<br>.dat - Ansys plot data file | <UnitSpec> | String | For example, "kV, Mhz, yd" | <UseTraceNumberFormat> | Boolean | "True", "False" |
| Name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Type                                                                           | Description                                                                                                                                                                                       |      |             |              |        |                                |            |        |                                                                                                                                                                                                   |            |        |                            |                        |         |                 |
| <ReportName>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | String                                                                         | Name of report to be exported.                                                                                                                                                                    |      |             |              |        |                                |            |        |                                                                                                                                                                                                   |            |        |                            |                        |         |                 |
| <FileName>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | String                                                                         | Full path of the exported image file name; with extension of<br>.txt - Post processor format file<br>.csv - Comma-delimited data file<br>.tab - Tab-separated file<br>.dat - Ansys plot data file |      |             |              |        |                                |            |        |                                                                                                                                                                                                   |            |        |                            |                        |         |                 |
| <UnitSpec>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | String                                                                         | For example, "kV, Mhz, yd"                                                                                                                                                                        |      |             |              |        |                                |            |        |                                                                                                                                                                                                   |            |        |                            |                        |         |                 |
| <UseTraceNumberFormat>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Boolean                                                                        | "True", "False"                                                                                                                                                                                   |      |             |              |        |                                |            |        |                                                                                                                                                                                                   |            |        |                            |                        |         |                 |
| <b>Return Value</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | None.                                                                          |                                                                                                                                                                                                   |      |             |              |        |                                |            |        |                                                                                                                                                                                                   |            |        |                            |                        |         |                 |

|                       |                                                                                                                                                                                       |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | ExportToFile(<ReportName>, <FileName>)                                                                                                                                                |
| <b>Python Example</b> | <pre>oModule.ExportToFile("rETotal", "C:/Users/Documents/rETotal.csv") oModule.ExportToFile("S Parameter Table 1", "D:/Users/Documents/cfft.csv", False, " kV, MHz, yd ", True)</pre> |

|                   |                                                                       |
|-------------------|-----------------------------------------------------------------------|
| <b>VB Syntax</b>  | ExportToFile <ReportName>, <FileName>                                 |
| <b>VB Example</b> | <pre>oModule.ExportToFile "rETotal", "C:/Documents/rETotal.csv"</pre> |

## ExportToFile [Reporter]

From a data table or plot, generates text format, comma delimited, tab delimited, .dat, or .rdat type output files.

|                     |                                                                           |        |                                             |
|---------------------|---------------------------------------------------------------------------|--------|---------------------------------------------|
| <b>UI Access</b>    | Right-click on report name in the Project tree and select <b>Export</b> . |        |                                             |
| <b>Parameters</b>   | Name                                                                      | Type   | Description                                 |
|                     | <ReportName>                                                              | String | Name of the report                          |
|                     | <FileName>                                                                | String | Path and File Name<br><br>Supported formats |
|                     | .txt                                                                      |        | Post processor format file                  |
|                     | .csv                                                                      |        | Comma-delimited data file                   |
|                     | .tab                                                                      |        | Tab-separated file                          |
| <b>Return Value</b> | .dat                                                                      |        | Ansys plot data file                        |
|                     | .rdat                                                                     |        | Ansys report data file                      |
| <b>Return Value</b> | None                                                                      |        |                                             |

|                       |                                                                     |
|-----------------------|---------------------------------------------------------------------|
| <b>Python Syntax</b>  | ExportToFile (<ReportName>, <FileName>)                             |
| <b>Python Example</b> | <pre>oModule.ExportToFile(<br/>    "Plot1", "c:\report1.dat")</pre> |

|                  |                                       |
|------------------|---------------------------------------|
| <b>VB Syntax</b> | ExportToFile <ReportName>, <FileName> |
|------------------|---------------------------------------|

|                   |                                                               |
|-------------------|---------------------------------------------------------------|
| <b>VB Example</b> | <pre>oModule.ExportToFile<br/>"Plot1", "c:\report1.dat"</pre> |
|-------------------|---------------------------------------------------------------|

## ExportMarkerTable

Exports the marker table to a .csv or .tab file.

|                     |                                                                                                                                                                                         |                                   |      |             |            |        |                                   |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|------|-------------|------------|--------|-----------------------------------|
| <b>UI Access</b>    | [product] > <b>Fields</b> > <b>Plot Fields</b> > <b>Marker</b> > <b>Export Marker Table</b> .                                                                                           |                                   |      |             |            |        |                                   |
| <b>Parameters</b>   | <table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;FileName&gt;</td> <td>String</td> <td>Name of export file include path.</td> </tr> </table> | Name                              | Type | Description | <FileName> | String | Name of export file include path. |
| Name                | Type                                                                                                                                                                                    | Description                       |      |             |            |        |                                   |
| <FileName>          | String                                                                                                                                                                                  | Name of export file include path. |      |             |            |        |                                   |
| <b>Return Value</b> | None.                                                                                                                                                                                   |                                   |      |             |            |        |                                   |

|                       |                                                                         |
|-----------------------|-------------------------------------------------------------------------|
| <b>Python Syntax</b>  | <code>ExportMarkerTable(&lt;FileName&gt;)</code>                        |
| <b>Python Example</b> | <code>oModule.ExportMarkerTable ("C:/work/FieldMarkerTable.csv")</code> |

|                   |                                                                       |
|-------------------|-----------------------------------------------------------------------|
| <b>VB Syntax</b>  | <code>ExportMarkerTable &lt;FileName&gt;</code>                       |
| <b>VB Example</b> | <code>oModule.ExportMarkerTable "C:/work/FieldMarkerTable.csv"</code> |

## FFTOnReport

Performs an FFT on a selected report.

|                   |                                                                                           |             |      |             |
|-------------------|-------------------------------------------------------------------------------------------|-------------|------|-------------|
| <b>UI Access</b>  | Right-click on <b>Results</b> in the project tree, select <b>Perform FFT On Report...</b> |             |      |             |
| <b>Parameters</b> | <table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> </table>   | Name        | Type | Description |
| Name              | Type                                                                                      | Description |      |             |

|                     |                           |        |                                                                                                                                                                                                            |
|---------------------|---------------------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     | <i>&lt;ReportName&gt;</i> | String | Name of specified report.                                                                                                                                                                                  |
|                     | <i>&lt;WindowName&gt;</i> | String | Name of window to apply for FFT. Possible values are "Rectangular", "Tri", "Van Hann", "Hamming", "Blackman", "Lanczos", "Weber", "Welch".                                                                 |
|                     | <i>&lt;Function&gt;</i>   | String | Function to apply on transformation values. Possible values are "none", "ang_deg", "ang_rad", "arg", "cang_deg", "cang_rad", "dB", "dB1 normalize", "dB2normalize", "dBc", "im", "mag", "normalize", "re". |
| <b>Return Value</b> | None.                     |        |                                                                                                                                                                                                            |

|                       |                                                                               |
|-----------------------|-------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | FFTOnReport < <i>ReportName</i> >, < <i>WindowName</i> >, < <i>Function</i> > |
| <b>Python Example</b> | <code>oModule.FFTOnReport("XY Plot 1", "Rectangular", "dB")</code>            |

|                   |                                                                                |
|-------------------|--------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | FFTOnReport(< <i>ReportName</i> >, < <i>WindowName</i> >, < <i>Function</i> >) |
| <b>VB Example</b> | <code>oModule.FFTOnReport "XY Plot 1", "Rectangular", "dB"</code>              |

## GetAllReportNames

Gets the names of existing reports in a design

|                     |                       |
|---------------------|-----------------------|
| <b>UI Access</b>    | N/A                   |
| <b>Parameters</b>   | None.                 |
| <b>Return Value</b> | Array of report names |

|                       |                                           |
|-----------------------|-------------------------------------------|
| <b>Python Syntax</b>  | GetAllReportNames()                       |
| <b>Python Example</b> | <code>oModule.GetAllReportNames ()</code> |

|                   |                                        |
|-------------------|----------------------------------------|
| <b>VB Syntax</b>  | GetAllReportNames                      |
| <b>VB Example</b> | <code>oModule.GetAllReportNames</code> |

## GetAllCategories

Get all available category names (not including variable and output-variables) in a solution for a report type and display type, returned as an array of text strings.

| <b>UI Access</b>                  | NA                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                                                               |      |             |                                 |        |                   |                                  |        |                       |                                   |        |                   |                                   |       |                                                               |
|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------|------|-------------|---------------------------------|--------|-------------------|----------------------------------|--------|-----------------------|-----------------------------------|--------|-------------------|-----------------------------------|-------|---------------------------------------------------------------|
| <b>Parameters</b>                 | <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;ReportType&gt;</code></td> <td>String</td> <td>Report type name.</td> </tr> <tr> <td><code>&lt;DisplayType&gt;</code></td> <td>String</td> <td>Name of display type.</td> </tr> <tr> <td><code>&lt;SolutionName&gt;</code></td> <td>String</td> <td>Name of solution.</td> </tr> <tr> <td><code>&lt;SimValueCtxt&gt;</code></td> <td>Array</td> <td>A context name, or array of strings that encode the contexts.</td> </tr> </tbody> </table> | Name                                                          | Type | Description | <code>&lt;ReportType&gt;</code> | String | Report type name. | <code>&lt;DisplayType&gt;</code> | String | Name of display type. | <code>&lt;SolutionName&gt;</code> | String | Name of solution. | <code>&lt;SimValueCtxt&gt;</code> | Array | A context name, or array of strings that encode the contexts. |
| Name                              | Type                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | Description                                                   |      |             |                                 |        |                   |                                  |        |                       |                                   |        |                   |                                   |       |                                                               |
| <code>&lt;ReportType&gt;</code>   | String                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Report type name.                                             |      |             |                                 |        |                   |                                  |        |                       |                                   |        |                   |                                   |       |                                                               |
| <code>&lt;DisplayType&gt;</code>  | String                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Name of display type.                                         |      |             |                                 |        |                   |                                  |        |                       |                                   |        |                   |                                   |       |                                                               |
| <code>&lt;SolutionName&gt;</code> | String                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Name of solution.                                             |      |             |                                 |        |                   |                                  |        |                       |                                   |        |                   |                                   |       |                                                               |
| <code>&lt;SimValueCtxt&gt;</code> | Array                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | A context name, or array of strings that encode the contexts. |      |             |                                 |        |                   |                                  |        |                       |                                   |        |                   |                                   |       |                                                               |
| <b>Return Value</b>               | Array of text strings                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                                                               |      |             |                                 |        |                   |                                  |        |                       |                                   |        |                   |                                   |       |                                                               |

|                       |                                                                                                                       |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | GetAllCategories(<ReportType>, <DisplayType>, <SolutionName>, <SimValueCtxt>)                                         |
| <b>Python Example</b> | <code>oModule.GetAllCategories ("Far Fields", "Rectangular Plot", "Setup1 : LastAdaptive", "Infinite Sphere1")</code> |

|                   |                                                                                                                         |
|-------------------|-------------------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | GetAllCategories <ReportType>, <DisplayType>, <SolutionName>, <SimValueCtxt>                                            |
| <b>VB Example</b> | <pre> oModule.GetAllCategories _ "Far Fields", "Rectangular Plot", _ "Setup1 : LastAdaptive", "Infinite Sphere1" </pre> |

## GetAllQuantities

Gets all available quantity names in category, returned as an array of text strings.

| <b>UI Access</b>    | NA                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                          |      |             |              |        |                   |               |        |                    |                |        |                   |                |       |                                                                  |                |        |                                                                                                                          |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------|------|-------------|--------------|--------|-------------------|---------------|--------|--------------------|----------------|--------|-------------------|----------------|-------|------------------------------------------------------------------|----------------|--------|--------------------------------------------------------------------------------------------------------------------------|
| <b>Parameters</b>   | <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ReportType&gt;</td> <td>String</td> <td>Report type name.</td> </tr> <tr> <td>&lt;DisplayType&gt;</td> <td>String</td> <td>Display type name.</td> </tr> <tr> <td>&lt;SolutionName&gt;</td> <td>String</td> <td>Name of solution.</td> </tr> <tr> <td>&lt;SimValueCtxt&gt;</td> <td>Array</td> <td>A context name, or array of string that encoded the contexts(l).</td> </tr> <tr> <td>&lt;CategoryName&gt;</td> <td>String</td> <td>A category name as input parameter. a category name returned in GetAllCategories() or "Variables", or "Output Variables"</td> </tr> </tbody> </table> | Name                                                                                                                     | Type | Description | <ReportType> | String | Report type name. | <DisplayType> | String | Display type name. | <SolutionName> | String | Name of solution. | <SimValueCtxt> | Array | A context name, or array of string that encoded the contexts(l). | <CategoryName> | String | A category name as input parameter. a category name returned in GetAllCategories() or "Variables", or "Output Variables" |
| Name                | Type                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Description                                                                                                              |      |             |              |        |                   |               |        |                    |                |        |                   |                |       |                                                                  |                |        |                                                                                                                          |
| <ReportType>        | String                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Report type name.                                                                                                        |      |             |              |        |                   |               |        |                    |                |        |                   |                |       |                                                                  |                |        |                                                                                                                          |
| <DisplayType>       | String                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Display type name.                                                                                                       |      |             |              |        |                   |               |        |                    |                |        |                   |                |       |                                                                  |                |        |                                                                                                                          |
| <SolutionName>      | String                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Name of solution.                                                                                                        |      |             |              |        |                   |               |        |                    |                |        |                   |                |       |                                                                  |                |        |                                                                                                                          |
| <SimValueCtxt>      | Array                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | A context name, or array of string that encoded the contexts(l).                                                         |      |             |              |        |                   |               |        |                    |                |        |                   |                |       |                                                                  |                |        |                                                                                                                          |
| <CategoryName>      | String                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | A category name as input parameter. a category name returned in GetAllCategories() or "Variables", or "Output Variables" |      |             |              |        |                   |               |        |                    |                |        |                   |                |       |                                                                  |                |        |                                                                                                                          |
| <b>Return Value</b> | Array of text strings                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                                                                                                                          |      |             |              |        |                   |               |        |                    |                |        |                   |                |       |                                                                  |                |        |                                                                                                                          |

|                       |                                                                                                   |
|-----------------------|---------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | GetAllQuantities(<ReportType>, <DisplayType>, <SolutionName>, <SimValueCtxt>, <CategoryName>)     |
| <b>Python Example</b> | <pre> oModule.GetAllQuantities( "Far Fields", "Rectangular Plot", "Setup1 : LastAdaptive", </pre> |

|  |                             |
|--|-----------------------------|
|  | "Infinite Spherel", "Gain") |
|--|-----------------------------|

|                   |                                                                                                                                       |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | GetAllQuantities < <i>ReportType</i> >,< <i>DisplayType</i> >,< <i>SolutionName</i> >,< <i>SimValueCtxt</i> >,< <i>CategoryName</i> > |
| <b>VB Example</b> | <pre> oModule.GetAllQuantities _ "Far Fields", "Rectangular Plot", _ "Setup1 : LastAdaptive", ) "Infinite Spherel", "Gain" </pre>     |

## GetAvailableDisplayTypes

Retrieves all supported display types in report type as an array of text strings.

| <b>UI Access</b>      | N/A                                                                                                                                                                              |                   |      |             |                       |        |                   |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|------|-------------|-----------------------|--------|-------------------|
| <b>Parameters</b>     | <table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;<i>ReportType</i>&gt;</td> <td>String</td> <td>Report type name.</td> </tr> </table> | Name              | Type | Description | < <i>ReportType</i> > | String | Report type name. |
| Name                  | Type                                                                                                                                                                             | Description       |      |             |                       |        |                   |
| < <i>ReportType</i> > | String                                                                                                                                                                           | Report type name. |      |             |                       |        |                   |
| <b>Return Value</b>   | Array of text strings                                                                                                                                                            |                   |      |             |                       |        |                   |

|                       |                                                              |
|-----------------------|--------------------------------------------------------------|
| <b>Python Syntax</b>  | GetAvailableDisplayTypes(< <i>ReportType</i> >)              |
| <b>Python Example</b> | <pre> oModule.GetAvailableDisplayTypes ("Far Fields") </pre> |

|                   |                                               |
|-------------------|-----------------------------------------------|
| <b>VB Syntax</b>  | GetAvailableDisplayTypes <ReportType>         |
| <b>VB Example</b> | oModule.GetAvailableDisplayTypes "Far Fields" |

## GetAvailableReportTypes

Retrieves all available report types in the current Design as an array of text string.

|                     |                       |
|---------------------|-----------------------|
| <b>UI Access</b>    | N/A                   |
| <b>Parameters</b>   | None.                 |
| <b>Return Value</b> | Array of text strings |

|                       |                                    |
|-----------------------|------------------------------------|
| <b>Python Syntax</b>  | GetAvailableReportTypes()          |
| <b>Python Example</b> | oModule.GetAvailableReportTypes () |

|                   |                                 |
|-------------------|---------------------------------|
| <b>VB Syntax</b>  | GetAvailableReportTypes         |
| <b>VB Example</b> | oModule.GetAvailableReportTypes |

## GetAvailableSolutions

Gets all available solutions in report type as an array of text strings.

|                  |     |
|------------------|-----|
| <b>UI Access</b> | N/A |
|------------------|-----|

|                     |                       |        |                   |
|---------------------|-----------------------|--------|-------------------|
| <b>Parameters</b>   | Name                  | Type   | Description       |
|                     | <ReportType>          | String | Report type name. |
| <b>Return Value</b> | Array of text strings |        |                   |

|                       |                                              |
|-----------------------|----------------------------------------------|
| <b>Python Syntax</b>  | GetAvailableSolutions(<ReportType>)          |
| <b>Python Example</b> | oModule.GetAvailableSolutions ("Far Fields") |

|                   |                                            |
|-------------------|--------------------------------------------|
| <b>VB Syntax</b>  | GetAvailableSolutions <ReportType>         |
| <b>VB Example</b> | oModule.GetAvailableSolutions "Far Fields" |

## GetDataExpressions

Returns data expressions.

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

|                     |                       |
|---------------------|-----------------------|
| <b>UI Access</b>    | N/A                   |
| <b>Parameters</b>   | N/A                   |
| <b>Return Value</b> | Array of text strings |

|                       |                                                       |
|-----------------------|-------------------------------------------------------|
| <b>Python Syntax</b>  | GetDataExpressions()                                  |
| <b>Python Example</b> | <pre>expressions = oModule.GetDataExpressions()</pre> |

|                   |                                                                       |
|-------------------|-----------------------------------------------------------------------|
| <b>VB Syntax</b>  | GetDataExpressions()                                                  |
| <b>VB Example</b> | <pre>dim expressions expressions = oModule.GetDataExpressions()</pre> |

## GetDataUnits

Returns text string containing units.

### Important:

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

|                     |                                         |        |                                                           |
|---------------------|-----------------------------------------|--------|-----------------------------------------------------------|
| <b>UI Access</b>    | N/A                                     |        |                                                           |
| <b>Parameters</b>   | Name                                    | Type   | Description                                               |
|                     | <expressionString>                      | String | Can be returned from <a href="#">GetDataExpressions()</a> |
| <b>Return Value</b> | Text string of units; empty if no units |        |                                                           |

|                      |                                  |
|----------------------|----------------------------------|
| <b>Python Syntax</b> | GetDataUnits(<expressionString>) |
|----------------------|----------------------------------|

|                       |                                                |
|-----------------------|------------------------------------------------|
| <b>Python Example</b> | <code>oModule.GetDataUnits(expressions)</code> |
|-----------------------|------------------------------------------------|

|                   |                                                     |
|-------------------|-----------------------------------------------------|
| <b>VB Syntax</b>  | <code>GetDataUnits(&lt;expressionString&gt;)</code> |
| <b>VB Example</b> | <code>oModule.GetDataUnits(expressions)</code>      |

## GetDesignVariableNames

Returns array of strings containing design variable names.

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

|                     |                  |
|---------------------|------------------|
| <b>UI Access</b>    | NA               |
| <b>Parameters</b>   | NA               |
| <b>Return Value</b> | Array of strings |

|                       |                                                       |
|-----------------------|-------------------------------------------------------|
| <b>Python Syntax</b>  | <code>GetDesignVariableNames()</code>                 |
| <b>Python Example</b> | <code>names = oModule.GetDesignVariableNames()</code> |

|                  |                                       |
|------------------|---------------------------------------|
| <b>VB Syntax</b> | <code>GetDesignVariableNames()</code> |
|------------------|---------------------------------------|

**VB Example**

```
dim names  
names = oModule.GetDesignVariableNames()
```

## GetDesignVariableUnits

Returns array of strings containing design variable units.

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

|                     |                                          |                |                                                                              |
|---------------------|------------------------------------------|----------------|------------------------------------------------------------------------------|
| <b>UI Access</b>    | N/A                                      |                |                                                                              |
| <b>Parameters</b>   | Name<br><varName>                        | Type<br>String | Description<br>Can be returned from <a href="#">GetDesignVariableNames()</a> |
| <b>Return Value</b> | Text string of units; empty if no units. |                |                                                                              |

**Python Syntax**

```
GetDesignVariableUnits(<varName>)
```

**Python Example**

```
units = oModule.GetDesignVariableUnits('Variable Name')
```

**VB Syntax**

```
GetDesignVariableUnits(<varName>)
```

**VB Example**

```
dim units
```

```
units = oModule.GetDesignVariableUnits("Variable Name")
```

## GetDesignVariableValue

Returns a design variable's value.

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

|                     |              |        |                                                               |
|---------------------|--------------|--------|---------------------------------------------------------------|
| <b>UI Access</b>    | N/A          |        |                                                               |
| <b>Parameters</b>   | Name         | Type   | Description                                                   |
|                     | <varName>    | String | Can be returned from <a href="#">GetDesignVariableNames()</a> |
| <b>Return Value</b> | Double value |        |                                                               |

|                       |                                                      |
|-----------------------|------------------------------------------------------|
| <b>Python Syntax</b>  | GetDesignVariableValue(<varName>, <siValue>)         |
| <b>Python Example</b> | value = oModule.GetDesignVariableValue('varName', 1) |

|                   |                                                                              |
|-------------------|------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | GetDesignVariableValue(<varName>, <siValue>)                                 |
| <b>VB Example</b> | <pre>dim value value = oModule.GetDesignVariableValue("varName", True)</pre> |

## GetDesignVariationKey

Returns a design's Variation Key.

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

|                     |                                  |
|---------------------|----------------------------------|
| <b>UI Access</b>    | N/A                              |
| <b>Parameters</b>   | N/A                              |
| <b>Return Value</b> | String containing variation key. |

|                       |                                              |
|-----------------------|----------------------------------------------|
| <b>Python Syntax</b>  | GetDesignVariationKey()                      |
| <b>Python Example</b> | <code>oModule.GetDesignVariationKey()</code> |

|                   |                                                                    |
|-------------------|--------------------------------------------------------------------|
| <b>VB Syntax</b>  | GetDesignVariationKey()                                            |
| <b>VB Example</b> | <code>dim key<br/>set key = oModule.GetDesignVariationKey()</code> |

## GetDisplayType (Maxwell)

**Use:** Get Display type of a report.

**Command:** None

**Syntax:** GetDisplayType <ReportName>

**Return Value:** String for display type

**Parameters:**

<ReportName>

Type: string

The name of report to find the display type for.

**Example:** Dim displayType

```
displayType = oModule.GetDisplayType("XY Plot 1")
```

## GetImagDataValues

Returns array of imaginary data values.

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

|                     |                    |        |                                                           |
|---------------------|--------------------|--------|-----------------------------------------------------------|
| <b>UI Access</b>    | N/A                |        |                                                           |
| <b>Parameters</b>   | Name               | Type   | Description                                               |
|                     | <expressionString> | String | Can be returned from <a href="#">GetDataExpressions()</a> |
| <b>Return Value</b> | Array of doubles   |        |                                                           |

|                       |                                                             |
|-----------------------|-------------------------------------------------------------|
| <b>Python Syntax</b>  | GetImagDataValues(<expressionString>,<siValue>)             |
| <b>Python Example</b> | imaginaryvalues = oModule.GetImagDataValues('expression',1) |

|                   |                                                                                       |
|-------------------|---------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | GetImagDataValues(<expressionString>,<siValue>)                                       |
| <b>VB Example</b> | dim imaginaryvalues<br>imaginaryvalues = oModule.GetImagDataValues("expression",True) |

## GetPerQuantityPrimarySweepValues

Returns per quantity primary sweep values.

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

|                     |                                                                                                        |        |                                                             |
|---------------------|--------------------------------------------------------------------------------------------------------|--------|-------------------------------------------------------------|
| <b>UI Access</b>    | N/A                                                                                                    |        |                                                             |
| <b>Parameters</b>   | Name                                                                                                   | Type   | Description                                                 |
|                     | <expressionString>                                                                                     | String | Can be returned from <a href="#">GetDataExpressions()</a> . |
| <b>Return Value</b> | Array of doubles if <a href="#">IsPerQuantityPrimarySweep()</a> returned True; error if returned False |        |                                                             |

|                       |                                                                                          |
|-----------------------|------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | GetPerQuantitySweepValues(<expressionString>, <siValue>)                                 |
| <b>Python Example</b> | <pre>sweepvalues = oModule.GetPerQuantitySweepValues('0.111,0.201,0.345,0.231', 1)</pre> |

|                   |                                                                                                             |
|-------------------|-------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | GetPerQuantitySweepValues(<expressionString>, <siValue>)                                                    |
| <b>VB Example</b> | <pre>dim sweepvalues sweepvalues = oModule.GetPerQuantitySweepValues("0.111,0.201,0.345,0.231", True)</pre> |

## GetPropertyValue

Returns the value of a single property belonging to a specific *<PropServer>* and *<PropTab>*. This function is available with the Project, Design or Editor objects, including definition editors.

**Tip:**

Use the script recording feature and edit a property, and then view the resulting script to see the format for that property.

|                   |                        |        |                                                                                                                                                                                                                                                                                                                                                                                      |
|-------------------|------------------------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>UI Access</b>  | N/A                    |        |                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>Parameters</b> | Name                   | Type   | Description                                                                                                                                                                                                                                                                                                                                                                          |
|                   | <i>&lt;PropTab&gt;</i> | String | <p>One of the following, where tab titles are shown in parentheses:</p> <ul style="list-style-type: none"> <li>• PassedParameterTab ("Parameter Values")</li> <li>• DefinitionParameterTab (Parameter Defaults")</li> <li>• LocalVariableTab ("Variables" or "Local Variables")</li> <li>• ProjectVariableTab ("Project variables")</li> <li>• ConstantsTab ("Constants")</li> </ul> |

|                     |                               |        |                                                                                                                                                                                                                                                                                     |
|---------------------|-------------------------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     |                               |        | <ul style="list-style-type: none"> <li>• BaseElementTab ("Symbol" or "Footprint")</li> <li>• ComponentTab ("General")</li> <li>• Component("Component")</li> <li>• CustomTab ("Intrinsic Variables")</li> <li>• Quantities ("Quantities")</li> <li>• Signals ("Signals")</li> </ul> |
|                     | <PropServer>                  | String | An object identifier, generally returned from another script method, such as<br>CompInst@R;2;3                                                                                                                                                                                      |
|                     | <PropName>                    | String | Name of the property.                                                                                                                                                                                                                                                               |
| <b>Return Value</b> | String value of the property. |        |                                                                                                                                                                                                                                                                                     |

|                       |                                                                                                                                                          |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | GetPropertyValue (<PropTab>, <PropServer>, <PropName>)                                                                                                   |
| <b>Python Example</b> | <pre>selectionArray = oEditor.GetSelections()  for k in selectionArray:     val = oEditor.GetPropertyValues("PassedParameterTab", k, "R")     ... </pre> |

|                   |                                                                             |
|-------------------|-----------------------------------------------------------------------------|
| <b>VB Syntax</b>  | GetPropertyValues (<PropTab>, <PropServer>, <PropName>)                     |
| <b>VB Example</b> | <pre>selectionArray = oEditor.GetSelections  for k in selectionArray:</pre> |

```
val = oEditor.GetPropertyValue("PassedParameterTab", k, "R")
...

```

## GetRealDataValues

Returns array of real data values.

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

|              |                    |        |                                                           |
|--------------|--------------------|--------|-----------------------------------------------------------|
| UI Access    | N/A                |        |                                                           |
| Parameters   | Name               | Type   | Description                                               |
|              | <expressionString> | String | Can be returned from <a href="#">GetDataExpressions()</a> |
| Return Value | Array of doubles   |        |                                                           |

|                |                                                        |
|----------------|--------------------------------------------------------|
| Python Syntax  | GetRealDataValues(<expressionString>,<siValue>)        |
| Python Example | realvalues = oModule.GetRealDataValues('expression',1) |

|           |                                                 |
|-----------|-------------------------------------------------|
| VB Syntax | GetRealDataValues(<expressionString>,<siValue>) |
|-----------|-------------------------------------------------|

**VB Example**

```
dim realvalues  
realvalues = oModule.GetRealDataValues("expression",True)
```

## GetReportTraceNames

Gets the names of existing trace names in a plot.

|                     |                                                                                                                                                                    |                         |      |             |            |        |                         |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|------|-------------|------------|--------|-------------------------|
| <b>UI Access</b>    | N/A                                                                                                                                                                |                         |      |             |            |        |                         |
| <b>Parameters</b>   | <table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;PlotName&gt;</td><td>String</td><td>Name of specified plot.</td></tr></table> | Name                    | Type | Description | <PlotName> | String | Name of specified plot. |
| Name                | Type                                                                                                                                                               | Description             |      |             |            |        |                         |
| <PlotName>          | String                                                                                                                                                             | Name of specified plot. |      |             |            |        |                         |
| <b>Return Value</b> | Array of strings containing trace names.                                                                                                                           |                         |      |             |            |        |                         |

**Python Syntax**

```
GetReportTraceNames(<PlotName>)
```

**Python Example**

```
oModule.GetReportTraceNames ("SParameter Plot 1")
```

**VB Syntax**

```
GetReportTraceNames <PlotName>
```

**VB Example**

```
oModule.GetReportTraceNames "SParameter Plot 1"
```

## GetSolutionContexts

Gets all available solution context names in a solution as an array of text strings.

**UI Access**

N/A

|                     |                       |        |                    |
|---------------------|-----------------------|--------|--------------------|
| <b>Parameters</b>   | Name                  | Type   | Description        |
|                     | <ReportType>          | String | Report type name.  |
|                     | <DisplayType>         | String | Display type name. |
|                     | <SolutionName>        | String | Name of solution.  |
| <b>Return Value</b> | Array of text strings |        |                    |

|                       |                                                                                                      |
|-----------------------|------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | GetSolutionContexts(<ReportType>, <DisplayType>, <SolutionName>)                                     |
| <b>Python Example</b> | <pre>oModule.GetSolutionContexts(     "Far Fields", "Rectangular Plot", "Setup1:LastAdaptive")</pre> |

|                   |                                                                                                            |
|-------------------|------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | GetSolutionContexts <ReportType>, <DisplayType>, <SolutionName>                                            |
| <b>VB Example</b> | <pre>oModule.GetSolutionContexts _     "Far Fields", "Rectangular Plot", _     "Setup1:LastAdaptive"</pre> |

## GetSolutionDataPerVariation

Obtains solution data for a given report type and solution. You must have already run a simulation.

|                   |                   |                  |                                                     |
|-------------------|-------------------|------------------|-----------------------------------------------------|
| <b>UI Access</b>  | N/A               |                  |                                                     |
| <b>Parameters</b> | Name              | Type             | Description                                         |
|                   | <reportTypeArg>   | String           | Report type name as input parameter.                |
|                   | <solutionNameArg> | String           | Solution name as input parameter.                   |
|                   | <simValueCtxtArg> | Structured Array | Same as ContextArray values created in the relevant |

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                 |                                                                                                                                      |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|--------------------------------------------------------------------------------------------------------------------------------------|
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                 | CreateReport script.                                                                                                                 |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | <familiesArg>   | Array of Strings<br>Same as FamiliesArray values created in the relevant CreateReport script.                                        |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | <expressionArg> | String or Array of Strings<br>Text string or array of text strings; valid expression, may validate it as the data-table Y-component. |
| <b>Return Value</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                 | ARRAY of ISolutionDataResultComInterface objects, containing:                                                                        |
| <ul style="list-style-type: none"> <li>• <a href="#">GetSweepNames()</a></li> <li>• <a href="#">GetSweepUnits()</a></li> <li>• <a href="#">GetSweepValues()</a></li> <li>• <a href="#">IsPerQuantityPrimarySweep()</a></li> <li>• <a href="#">GetDataExpressions()</a> – should be the same as &lt;expressionArg&gt;</li> <li>• <a href="#">GetPerQuantityPrimarySweepValues()</a></li> <li>• <a href="#">IsDataComplex()</a></li> <li>• <a href="#">GetDataUnits()</a></li> <li>• <a href="#">GetRealDataValues()</a></li> <li>• <a href="#">GetImagDataValues()</a></li> <li>• <a href="#">ReleaseData()</a></li> <li>• <a href="#">GetDesignVariableNames()</a></li> <li>• <a href="#">GetDesignVariableUnits()</a></li> <li>• <a href="#">GetDesignVariableValue()</a></li> <li>• <a href="#">GetDesignVariationKey()</a></li> </ul> |                 |                                                                                                                                      |

**Note:**

- This command is *not* recordable from the UI, but its parameters are similar to CreateReport, so you may record a CreateReport script to get the parameter values.
- For the returned ISolutionDataResultComInterface object, some of its functions have an optional boolean parameter: SIValue. SIValue defaults to True. When the pass in value is True, return data values will be in Standard International values; when False, return data values will be in the current units.

**Example:** Freq Sweep with [1GHz, 2GHz,3GHz], GetSweepUnits("Freq") return "GHz"; GetSweepValues("Freq", True) return [1000000,2000000,3000000]; GetSweepValues("Freq", False) return [1,2,3].

|                       |                                                                                                                                                                                                                                                              |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | GetSolutionDataPerVariation(reportTypeArg, solutionNameArg, simValueCtxtArg, familiesArg, expressionArg)                                                                                                                                                     |
| <b>Python Example</b> | <pre> oModule = oDesign.GetModule("ReportSetup") arr = oModule.GetSolutionDataPerVariation('Modal Solution Data', 'Setup1 : Sweep', ['Domain:=' , 'Sweep'], ['Freq:=' , ['All']] , 'offset:=' , ['All']] , ['S (Port1,Port1)', 'dB(S(Port1,Port3))']) </pre> |

|                   |                                                                                                                                                        |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | GetSolutionDataPerVariation(reportTypeArg, solutionNameArg, simValueCtxtArg, familiesArg, expressionArg)                                               |
| <b>VB Example</b> | <pre> set solutionData = oModule.GetSolutionDataPerVariation(     "Modal Solution Data",     "Setup1 : Sweep",     Array("Domain:=" , "Sweep"), </pre> |

```
    Array("Freq:=", Array("All"), "offset:=", Array("All")),  
    Array("S(Port1,Port1)", "dB(S(Port1,Port3))")  
)
```

## GetSweepNames

Returns array of text strings containing primary sweep name(s).

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

|                     |                       |
|---------------------|-----------------------|
| <b>UI Access</b>    | N/A                   |
| <b>Parameters</b>   | N/A                   |
| <b>Return Value</b> | Array of text strings |

|                       |                                       |
|-----------------------|---------------------------------------|
| <b>Python Syntax</b>  | GetSweepNames()                       |
| <b>Python Example</b> | sweepnames = oModule.GetSweepNames () |

|                   |                 |
|-------------------|-----------------|
| <b>VB Syntax</b>  | GetSweepNames() |
| <b>VB Example</b> | dim sweepnames  |

|  |                                                   |
|--|---------------------------------------------------|
|  | <code>sweepnames = oModule.GetSweepNames()</code> |
|--|---------------------------------------------------|

## GetSweepUnits

Returns text string containing units.

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

|                     |                              |                |                                   |
|---------------------|------------------------------|----------------|-----------------------------------|
| <b>UI Access</b>    | N/A                          |                |                                   |
| <b>Parameters</b>   | Name<br><sweepName>          | Type<br>String | Description<br>Primary sweep name |
| <b>Return Value</b> | Text string containing units |                |                                   |

|                       |                                                            |
|-----------------------|------------------------------------------------------------|
| <b>Python Syntax</b>  | <code>GetSweepUnits(&lt;sweepName&gt;)</code>              |
| <b>Python Example</b> | <code>sweepunits = oModule.GetSweepUnits('Sweep 1')</code> |

|                   |                                                                               |
|-------------------|-------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | <code>GetSweepUnits(&lt;sweepName&gt;)</code>                                 |
| <b>VB Example</b> | <code>dim sweepunits<br/>sweepunits = oModule.GetSweepUnits("Sweep 1")</code> |

## GetSweepValues

Returns sweep values.

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

|                     |                  |        |                    |
|---------------------|------------------|--------|--------------------|
| <b>UI Access</b>    | N/A              |        |                    |
| <b>Parameters</b>   | Name             | Type   | Description        |
|                     | <sweepName>      | String | Primary sweep name |
| <b>Return Value</b> | Array of doubles |        |                    |

|                       |                                                       |
|-----------------------|-------------------------------------------------------|
| <b>Python Syntax</b>  | GetSweepValues(<sweepName>, <siValue>)                |
| <b>Python Example</b> | sweepvalues = oModule.GetSweepValues('Sweep 1', True) |

|                   |                                                                          |
|-------------------|--------------------------------------------------------------------------|
| <b>VB Syntax</b>  | GetSweepValues(<sweepName>, <siValue>)                                   |
| <b>VB Example</b> | dim sweepvalues<br>sweepvalues = oModule.GetSweepValues("Sweep 1", True) |

## GroupPlotCurvesByGroupingStrategy

Groups curves in a Stacked Plot automatically based on a curve grouping strategy.

|                     |              |        |                 |
|---------------------|--------------|--------|-----------------|
| <b>UI Access</b>    | N/A          |        |                 |
| <b>Parameters</b>   | Name         | Type   | Description     |
|                     | <ReportName> | String | Name of report. |
| <b>Return Value</b> | None.        |        |                 |

|                       |                                                                            |
|-----------------------|----------------------------------------------------------------------------|
| <b>Python Syntax</b>  | GroupPlotCurvesByGroupingStrategy(<ReportName>, <GroupStrategy>)           |
| <b>Python Example</b> | oModule.GroupPlotCurvesByGroupingStrategy ("Transient Plot 1", "By Trace") |

|                   |                                                                          |
|-------------------|--------------------------------------------------------------------------|
| <b>VB Syntax</b>  | GroupPlotCurvesByGroupingStrategy <ReportName>, <GroupStrategy>          |
| <b>VB Example</b> | oModule.GroupPlotCurvesByGroupingStrategy "Transient Plot 1", "By Trace" |

## ImportIntoReport

Imports .tab, .csv, and .dat format files into a report.

|                   |                                                                             |        |                    |
|-------------------|-----------------------------------------------------------------------------|--------|--------------------|
| <b>UI Access</b>  | Right-click on report name in the Project tree and select <b>Import....</b> |        |                    |
| <b>Parameters</b> | Name                                                                        | Type   | Description        |
|                   | <ReportName>                                                                | String | Name of the Report |

|                     |      |  |      |                           |
|---------------------|------|--|------|---------------------------|
|                     |      |  | .csv | Comma-delimited data file |
|                     |      |  | .tab | Tab-separated file        |
|                     |      |  | .dat | Ansys plot data file      |
| <b>Return Value</b> | None |  |      |                           |

|                       |                                                                   |
|-----------------------|-------------------------------------------------------------------|
| <b>Python Syntax</b>  | ImportIntoReport (<ReportName>, <FileName>)                       |
| <b>Python Example</b> | <code>oDesign.ImportIntoReport ("Plot1", "c:\report1.dat")</code> |

|                   |                                                                 |
|-------------------|-----------------------------------------------------------------|
| <b>VB Syntax</b>  | ImportIntoReport <ReportName>, <FileName>                       |
| <b>VB Example</b> | <code>oDesign.ImportIntoReport "Plot1", "c:\report1.dat"</code> |

## IsDataComplex

Returns whether an expression is complex.

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

|                   |                                                                                         |             |      |             |
|-------------------|-----------------------------------------------------------------------------------------|-------------|------|-------------|
| <b>UI Access</b>  | NA                                                                                      |             |      |             |
| <b>Parameters</b> | <table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> </table> | Name        | Type | Description |
| Name              | Type                                                                                    | Description |      |             |

|                     |                                                            |                                                             |
|---------------------|------------------------------------------------------------|-------------------------------------------------------------|
|                     | <expressionString> String                                  | Can be returned from <a href="#">GetDataExpressions()</a> . |
| <b>Return Value</b> | Boolean (True if expression is Complex data; False if not) |                                                             |

|                       |                                                           |
|-----------------------|-----------------------------------------------------------|
| <b>Python Syntax</b>  | <code>IsDataComplex(&lt;expressionString&gt;)</code>      |
| <b>Python Example</b> | <code>oModule.IsDataComplex('.001,.234,.455,.434')</code> |

|                   |                                                           |
|-------------------|-----------------------------------------------------------|
| <b>VB Syntax</b>  | <code>IsDataComplex(&lt;expressionString&gt;)</code>      |
| <b>VB Example</b> | <code>oModule.IsDataComplex('.001,.234,.455,.434')</code> |

## IsPerQuantityPrimarySweep

Returns whether data expressions have different primary sweep values.

**Important:**

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

|                     |                                                                        |
|---------------------|------------------------------------------------------------------------|
| <b>UI Access</b>    | N/A                                                                    |
| <b>Parameters</b>   | N/A                                                                    |
| <b>Return Value</b> | Boolean (True if data expressions have different primary sweep values) |

|                       |                                                      |
|-----------------------|------------------------------------------------------|
| <b>Python Syntax</b>  | IsPerQuantityPrimarySweep()                          |
| <b>Python Example</b> | <pre>var = oModule.IsPerQuantityPrimarySweep()</pre> |

|                   |                                                              |
|-------------------|--------------------------------------------------------------|
| <b>VB Syntax</b>  | IsPerQuantityPrimarySweep()                                  |
| <b>VB Example</b> | <pre>dim var var = oModule.IsPerQuantityPrimarySweep()</pre> |

## MovePlotCurvesToGroup

In a Stacked Plot move curve(s) from its stack(s) to an existing stack. Here term ‘group’ is synonymous to ‘stack’ in the context of cartesian stacked plot.

|                     |                                                                                                                                                                                                                                                                                                                                          |                               |      |             |              |        |                 |              |       |                               |             |        |                           |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------|------|-------------|--------------|--------|-----------------|--------------|-------|-------------------------------|-------------|--------|---------------------------|
| <b>UI Access</b>    | N/A                                                                                                                                                                                                                                                                                                                                      |                               |      |             |              |        |                 |              |       |                               |             |        |                           |
| <b>Parameters</b>   | <table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;ReportName&gt;</td><td>String</td><td>Name of report.</td></tr><tr><td>&lt;CurveArray&gt;</td><td>Array</td><td>Array of curve names to move.</td></tr><tr><td>&lt;StackName&gt;</td><td>String</td><td>Name of stack to move to.</td></tr></table> | Name                          | Type | Description | <ReportName> | String | Name of report. | <CurveArray> | Array | Array of curve names to move. | <StackName> | String | Name of stack to move to. |
| Name                | Type                                                                                                                                                                                                                                                                                                                                     | Description                   |      |             |              |        |                 |              |       |                               |             |        |                           |
| <ReportName>        | String                                                                                                                                                                                                                                                                                                                                   | Name of report.               |      |             |              |        |                 |              |       |                               |             |        |                           |
| <CurveArray>        | Array                                                                                                                                                                                                                                                                                                                                    | Array of curve names to move. |      |             |              |        |                 |              |       |                               |             |        |                           |
| <StackName>         | String                                                                                                                                                                                                                                                                                                                                   | Name of stack to move to.     |      |             |              |        |                 |              |       |                               |             |        |                           |
| <b>Return Value</b> | None.                                                                                                                                                                                                                                                                                                                                    |                               |      |             |              |        |                 |              |       |                               |             |        |                           |

|                      |                                                                                                  |
|----------------------|--------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b> | MovePlotCurvesToGroup(<ReportName>, <CurveArray>, <StackName>)                                   |
| <b>Python</b>        | <pre>oModule.MovePlotCurvesToGroup("XY Stacked Plot 1", ["R2.V : TR", "R2.I : TR"], "Stack</pre> |

|                |     |
|----------------|-----|
| <b>Example</b> | 2") |
|----------------|-----|

|                   |                                                                                                                  |
|-------------------|------------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | MovePlotCurvesToGroup <ReportName>, <CurveArray>, <StackName>                                                    |
| <b>VB Example</b> | <pre> oModule.MovePlotCurvesToGroup _ "XY Stacked Plot 1", _ Array("R2.V : TR", "R2.I : TR"), _ "Stack 2" </pre> |

## MovePlotCurvesToNewGroup

Move curve(s) from its stack(s) to a new stack. Here term ‘group’ is synonymous to ‘stack’ in the context of Cartesian stacked plot.

| <b>UI Access</b>    | N/A                                                                                                                                                                                                                                                                                                    |                               |      |             |              |        |                 |              |       |                               |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------|------|-------------|--------------|--------|-----------------|--------------|-------|-------------------------------|
| <b>Parameters</b>   | <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ReportName&gt;</td> <td>String</td> <td>Name of report.</td> </tr> <tr> <td>&lt;CurveArray&gt;</td> <td>Array</td> <td>Array of curve names to move.</td> </tr> </tbody> </table> | Name                          | Type | Description | <ReportName> | String | Name of report. | <CurveArray> | Array | Array of curve names to move. |
| Name                | Type                                                                                                                                                                                                                                                                                                   | Description                   |      |             |              |        |                 |              |       |                               |
| <ReportName>        | String                                                                                                                                                                                                                                                                                                 | Name of report.               |      |             |              |        |                 |              |       |                               |
| <CurveArray>        | Array                                                                                                                                                                                                                                                                                                  | Array of curve names to move. |      |             |              |        |                 |              |       |                               |
| <b>Return Value</b> | None.                                                                                                                                                                                                                                                                                                  |                               |      |             |              |        |                 |              |       |                               |

|                       |                                                                                                  |
|-----------------------|--------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | MovePlotCurvesToNewGroup (<ReportName>, <CurveArray>)                                            |
| <b>Python Example</b> | <pre> oModule.MovePlotCurvesToNewGroup ( "XY Stacked Plot 1", ["R2.V : TR", "R2.I : TR"]) </pre> |

|                   |                                                                                                              |
|-------------------|--------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | MovePlotCurvesToNewGroup <ReportName>, <CurveArray>                                                          |
| <b>VB Example</b> | <pre>oModule.MovePlotCurvesToNewGroup _     "XY Stacked Plot 1", _     Array("R2.V : TR", "R2.I : TR")</pre> |

## PasteReports

Paste copied reports to results in the current project.

|                     |              |
|---------------------|--------------|
| <b>UI Access</b>    | Edit > Paste |
| <b>Parameters</b>   | None.        |
| <b>Return Value</b> | None.        |

|                       |                                    |
|-----------------------|------------------------------------|
| <b>Python Syntax</b>  | PasteReports ()                    |
| <b>Python Example</b> | <pre>oModule.PasteReports ()</pre> |

|                   |                                 |
|-------------------|---------------------------------|
| <b>VB Syntax</b>  | PasteReports                    |
| <b>VB Example</b> | <pre>oModule.PasteReports</pre> |

## PasteTraces

Pastes copied traces to a named plot.

|                     |                                   |                |                             |
|---------------------|-----------------------------------|----------------|-----------------------------|
| <b>UI Access</b>    | Paste                             |                |                             |
| <b>Parameters</b>   | Name<br><i>&lt;ReportName&gt;</i> | Type<br>String | Description<br>Name of plot |
| <b>Return Value</b> | None                              |                |                             |

|                       |                                               |
|-----------------------|-----------------------------------------------|
| <b>Python Syntax</b>  | PasteTraces ( <i>&lt;ReportName&gt;</i> )     |
| <b>Python Example</b> | <code>oModule.PasteTraces ("XY Plot1")</code> |

|                   |                                             |
|-------------------|---------------------------------------------|
| <b>VB Syntax</b>  | PasteTraces <i>&lt;ReportName&gt;</i>       |
| <b>VB Example</b> | <code>oModule.PasteTraces "XY Plot1"</code> |

## Release Data

Releases all cached data. After this function is called, all subsequent function calls to the object will fail.

### Important:

This script does not function on its own, but as part of [GetSolutionDataPerVariation](#).

|                  |     |
|------------------|-----|
| <b>UI Access</b> | N/A |
|------------------|-----|

|              |     |
|--------------|-----|
| Parameters   | N/A |
| Return Value | N/A |

|                |                                    |
|----------------|------------------------------------|
| Python Syntax  | ReleaseData()                      |
| Python Example | <code>oModule.ReleaseData()</code> |

|            |                                    |
|------------|------------------------------------|
| VB Syntax  | ReleaseData()                      |
| VB Example | <code>oModule.ReleaseData()</code> |

## RenameReport

Renames an existing report.

|                                    |                                                                                                                                                                                                                                                                         |                 |      |             |                                    |        |                 |                                    |        |                 |
|------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|------|-------------|------------------------------------|--------|-----------------|------------------------------------|--------|-----------------|
| UI Access                          | Select a report on the Project tree, right-click and select <b>Rename</b>                                                                                                                                                                                               |                 |      |             |                                    |        |                 |                                    |        |                 |
| Parameters                         | <table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><code>&lt;OldReportName&gt;</code></td><td>String</td><td>Old Report Name</td></tr><tr><td><code>&lt;NewReportName&gt;</code></td><td>String</td><td>New Report Name</td></tr></table> | Name            | Type | Description | <code>&lt;OldReportName&gt;</code> | String | Old Report Name | <code>&lt;NewReportName&gt;</code> | String | New Report Name |
| Name                               | Type                                                                                                                                                                                                                                                                    | Description     |      |             |                                    |        |                 |                                    |        |                 |
| <code>&lt;OldReportName&gt;</code> | String                                                                                                                                                                                                                                                                  | Old Report Name |      |             |                                    |        |                 |                                    |        |                 |
| <code>&lt;NewReportName&gt;</code> | String                                                                                                                                                                                                                                                                  | New Report Name |      |             |                                    |        |                 |                                    |        |                 |
| Return Value                       | None.                                                                                                                                                                                                                                                                   |                 |      |             |                                    |        |                 |                                    |        |                 |

|               |                                                                          |
|---------------|--------------------------------------------------------------------------|
| Python Syntax | <code>RenameReport (&lt;OldReportName&gt;, &lt;NewReportName&gt;)</code> |
|---------------|--------------------------------------------------------------------------|

|                       |                                                             |
|-----------------------|-------------------------------------------------------------|
| <b>Python Example</b> | <code>oModule.RenameReport("XY Plot1", "Reflection")</code> |
|-----------------------|-------------------------------------------------------------|

|                   |                                                                        |
|-------------------|------------------------------------------------------------------------|
| <b>VB Syntax</b>  | <code>RenameReport &lt;OldReportName&gt;, &lt;NewReportName&gt;</code> |
| <b>VB Example</b> | <code>oModule.RenameReport "XY Plot1", "Reflection"</code>             |

## RenameTrace

To rename a trace in a plot

| <b>UI Access</b>                | N/A                                                                                                                                                                                                                                                                                                                                                                                                        |                 |      |             |                                 |        |                 |                                |        |               |                              |        |                 |
|---------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|------|-------------|---------------------------------|--------|-----------------|--------------------------------|--------|---------------|------------------------------|--------|-----------------|
| <b>Parameters</b>               | <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;ReportName&gt;</code></td> <td>String</td> <td>Name of report.</td> </tr> <tr> <td><code>&lt;TraceName&gt;</code></td> <td>String</td> <td>Name of Trace</td> </tr> <tr> <td><code>&lt;NewName&gt;</code></td> <td>String</td> <td>New trace name.</td> </tr> </tbody> </table> | Name            | Type | Description | <code>&lt;ReportName&gt;</code> | String | Name of report. | <code>&lt;TraceName&gt;</code> | String | Name of Trace | <code>&lt;NewName&gt;</code> | String | New trace name. |
| Name                            | Type                                                                                                                                                                                                                                                                                                                                                                                                       | Description     |      |             |                                 |        |                 |                                |        |               |                              |        |                 |
| <code>&lt;ReportName&gt;</code> | String                                                                                                                                                                                                                                                                                                                                                                                                     | Name of report. |      |             |                                 |        |                 |                                |        |               |                              |        |                 |
| <code>&lt;TraceName&gt;</code>  | String                                                                                                                                                                                                                                                                                                                                                                                                     | Name of Trace   |      |             |                                 |        |                 |                                |        |               |                              |        |                 |
| <code>&lt;NewName&gt;</code>    | String                                                                                                                                                                                                                                                                                                                                                                                                     | New trace name. |      |             |                                 |        |                 |                                |        |               |                              |        |                 |
| <b>Return Value</b>             | None.                                                                                                                                                                                                                                                                                                                                                                                                      |                 |      |             |                                 |        |                 |                                |        |               |                              |        |                 |

|                       |                                                                                              |
|-----------------------|----------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | <code>RenameTrace(&lt;ReportName&gt;, &lt;TraceName&gt;, &lt;NewName&gt;)</code>             |
| <b>Python Example</b> | <code>oModule.RenameTrace ("XY Plot1",<br/>"dB(S(WavePort1,WavePort1))1", "Port1dBs")</code> |

|                   |                                                                                 |
|-------------------|---------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | <code>RenameTrace &lt;ReportName&gt;, &lt;TraceName&gt;, &lt;NewName&gt;</code> |
| <b>VB Example</b> | <code>oModule.RenameTrace "XY Plot1", _</code>                                  |

|  |                                                  |
|--|--------------------------------------------------|
|  | "dB (S (WavePort1, WavePort1)) 1",<br>"Port1dbS" |
|--|--------------------------------------------------|

## ResetPlotSettings

Resets plot settings to defaults.

|                     |                                                                                                |
|---------------------|------------------------------------------------------------------------------------------------|
| <b>UI Access</b>    | Right-click on a plot, select <b>Edit &gt; Reset Plot Settings</b>                             |
| <b>Parameters</b>   | Name      Type      Description<br><i>&lt;PlotName&gt;</i> String      Name of specified plot. |
| <b>Return Value</b> | None.                                                                                          |

|                       |                                                         |
|-----------------------|---------------------------------------------------------|
| <b>Python Syntax</b>  | ResetPlotSettings( <i>&lt;PlotName&gt;</i> )            |
| <b>Python Example</b> | oModule.ResetPlotSettings ("Differential S-parameters") |

|                   |                                                         |
|-------------------|---------------------------------------------------------|
| <b>VB Syntax</b>  | ResetPlotSettings < <i>PlotName</i> >                   |
| <b>VB Example</b> | oModule.ResetPlotSettings ("Differential S-parameters") |

## SavePlotSettingsAsDefault

Saves report plot settings as default.

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <b>UI Access</b> | <b>Report Templates &gt; Save Settings as Default</b> |
|------------------|-------------------------------------------------------|

|                     |                                 |                |                                                       |
|---------------------|---------------------------------|----------------|-------------------------------------------------------|
| <b>Parameters</b>   | Name<br><i>&lt;PlotName&gt;</i> | Type<br>String | Description<br>Name of plot to use for plot defaults. |
| <b>Return Value</b> | None.                           |                |                                                       |

|                       |                                                |
|-----------------------|------------------------------------------------|
| <b>Python Syntax</b>  | SavePlotSettingsAsDefault ("<PlotName>")       |
| <b>Python Example</b> | oModule.SavePlotSettingsAsDefault ("XY Plot1") |

|                   |                                               |
|-------------------|-----------------------------------------------|
| <b>VB Syntax</b>  | SavePlotSettingsAsDefault "<PlotName>"        |
| <b>VB Example</b> | oModule.SavePlotSettingsAsDefault "XY Plot 1" |

## UpdateAllReports

Updates all reports in the **Results** branch in the project tree.

|                     |                                                                                     |
|---------------------|-------------------------------------------------------------------------------------|
| <b>UI Access</b>    | Right-click on <b>Results</b> in the project tree, select <b>Update All Reports</b> |
| <b>Parameters</b>   | None                                                                                |
| <b>Return Value</b> | None                                                                                |

|                       |                             |
|-----------------------|-----------------------------|
| <b>Python Syntax</b>  | UpdateAllReports()          |
| <b>Python Example</b> | oModule.UpdateAllReports () |

|                   |                                       |
|-------------------|---------------------------------------|
| <b>VB Syntax</b>  | UpdateAllReports                      |
| <b>VB Example</b> | <code>oModule.UpdateAllReports</code> |

## UpdateReports

Updates specified reports.

|                     |                                          |               |                                                          |
|---------------------|------------------------------------------|---------------|----------------------------------------------------------|
| <b>UI Access</b>    | N/A                                      |               |                                                          |
| <b>Parameters</b>   | Name<br><code>&lt;ReportNames&gt;</code> | Type<br>Array | Description<br>Array of strings containing report names. |
| <b>Return Value</b> | None.                                    |               |                                                          |

|                       |                                                                 |
|-----------------------|-----------------------------------------------------------------|
| <b>Python Syntax</b>  | UpdateReports( <i>&lt;ReportNames&gt;</i> )                     |
| <b>Python Example</b> | <code>oModule.UpdateReports (["XY Plot 1", "XY Plot 4"])</code> |

|                   |                                                                    |
|-------------------|--------------------------------------------------------------------|
| <b>VB Syntax</b>  | UpdateReports <i>&lt;ReportNames&gt;</i>                           |
| <b>VB Example</b> | <code>oModule.UpdateReports Array("XY Plot 1", "XY Plot 4")</code> |

## UpdateTraces

Update the traces in a report for which traces are not automatically updated by the Report Traces dialog box, Update Report, Real Time selection.

| UI Access    | In <b>Report</b> dialog, click <b>Apply Traces</b> button |        |                                                                                                                                                                                                                                                                                                        |
|--------------|-----------------------------------------------------------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameters   | Name                                                      | Type   | Description                                                                                                                                                                                                                                                                                            |
|              | <ReportName>                                              | String | Name of Report.                                                                                                                                                                                                                                                                                        |
|              | <TraceNames>                                              | Array  | Array of strings containing trace names.                                                                                                                                                                                                                                                               |
|              | <SolutionName>                                            | String | Name of the solution.                                                                                                                                                                                                                                                                                  |
|              | <ContextArray>                                            | Array  | Context for which the expression is being evaluated. This can be an empty string if there is no context.<br><br>Array("Domain:=", <DomainType><br><br><DomainType> ex. "Sweep" or "Time"<br><br>Array("Context:=", <GeometryType><br><br><GeometryType> ex. "Infinite Spheren", "Spheren", "Polylinen" |
|              | <FamiliesArray>                                           | Array  | Contains sweep definitions for the report.<br><br>Array("<VariableName>:= ", <ValueArray><br><br><ValueArray><br><br>Array("All") or Array("Value1", "Value2", ... "Valuen")<br><br>examples of <VariableName> "Freq", "Theta", "Distance"                                                             |
|              | <ReportdataArray>                                         | Array  | This array contains the report quantity and X, Y, and (Z) axis definitions.<br><br>Array("X Component:=", <VariableName>,<br><br>"Y Component:=", <VariableName>   <ReportQuantityArray><br><br><ReportQuantityArray> ex. Array("dB(S(Port1, Port1))")                                                 |
|              | <ExtTraceInfo>                                            | Array  | Optional. Array defines extended trace information.                                                                                                                                                                                                                                                    |
| Return Value | None.                                                     |        |                                                                                                                                                                                                                                                                                                        |

|                       |                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | UpdateTraces(<ReportName>, <SolutionName>, <ContextArray>, <FamiliesArray>, <ReportdataArray>)                                                                                                                                                                                                                                                                                                                                     |
| <b>Python Example</b> | <pre>oModule.UpdateTraces ("XY Plot 1", ["NEG1.VAL"], "TR4", [     "NAME:Context",     "SimValueContext:=", [2,0,2,0,False,False, -1,1,0,1,1,"",0,0,"CG",False,"0","KP",False,"0","MH",False, "100","TE",False,"100s","TH",False,"40", "TS",False,"0ns","UF",False, "0","WT",False,"0","WW",False,"100"] ], [     "Spectrum:=", ["All"] ], [     "X Component:=", "Spectrum",     "Y Component:=", ["mag (NEG1.VAL)"] ], [])</pre> |

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | UpdateTraces <ReportName>, <SolutionName>, <ContextArray>, <FamiliesArray>, <ReportdataArray>                                                                                                                                                                                                                                                                                                                                    |
| <b>VB Example</b> | <pre> oModule.UpdateTraces "XY Plot1", Array("dB(S(WavePort1,WavePort1))"), _ "Setup1 : Sweep1", Array("Domain:=", "Time", "HoldTime:=", 1, _ "RiseTime:=", 0, "StepTime:=", 0, "Step:=", false, _ "WindowWidth:=", 1, "WindowType:=", 0, "KaiserParameter:=",1, _ "MaximumTime:=", 0), Array("Time:=", Array("All")), _ Array("X Component:=", "Time", _ "Y Component:=", Array("dB(S(WavePort1,WavePort1)))) ,_ Array() </pre> |

## UpdateTracesContextAndSweeps

Edits sweeps and context of multiple traces without affecting their component expressions.

|                   |                                                     |        |                                                                                                                                       |
|-------------------|-----------------------------------------------------|--------|---------------------------------------------------------------------------------------------------------------------------------------|
| <b>UI Access</b>  | <b>Modify Report</b> with multiple traces selected. |        |                                                                                                                                       |
| <b>Parameters</b> | Name                                                | Type   | Description                                                                                                                           |
|                   | <ReportName>                                        | String | Name of Report.                                                                                                                       |
|                   | <TraceNames>                                        | Array  | Array of strings containing trace names.                                                                                              |
|                   | <SolutionName>                                      | String | Name of the solution as listed in the <b>Modify Report</b> dialog box.<br><br>For example: "Setup1 : Last Adaptive"                   |
|                   | <ContextArray>                                      | Array  | Context for which the expression is being evaluated. This can be an empty string if there is no context.<br><br>ex. "Sweep" or "Time" |
|                   | <PointSet>                                          | Array  | Point set for the selected traces, for example, X and Y values for the plot.                                                          |

|                     |      |
|---------------------|------|
| <b>Return Value</b> | None |
|---------------------|------|

|                       |                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | UpdateTracesContextAndSweeps(<ReportName>, <TraceNames>, <SolutionName>, <ContextArray>, <PointSet>)                                                                                                                                                                                                                                                                                      |
| <b>Python Example</b> | <pre>oModule.UpdateTracesContextAndSweeps_ ("Active S Parameter Quick Report", ["dB(ActiveS(Port1:1))", "dB(ActiveS(Port2:1))"], "Setup1 : Sweep1", [], ["Freq:=", ["9GHz", "9.05GHz", "9.1GHz", "9.15GHz", "9.2GHz", "9.25GHz", "9.3GHz", "9.35GHz", "9.4GHz", "9.45GHz", "9.5GHz", "9.55GHz", "9.6GHz", "9.65GHz", "9.7GHz", "9.9GHz", "9.95GHz", "10GHz"], "offset:=", ["All"]])</pre> |

|                  |                                                                                                     |
|------------------|-----------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b> | UpdateTracesContextAndSweeps <ReportName>, <TraceNames>, <SolutionName>, <ContextArray>, <PointSet> |
|------------------|-----------------------------------------------------------------------------------------------------|

**VB Example**

```

oModule.UpdateTracesContextAndSweeps _
    "Active S Parameter Quick Report",_
    Array("dB(ActiveS(Port1:1))", "dB(ActiveS(Port2:1))"),_
    "Setup1 : Sweep1", Array(), _
    Array("Freq:=", _
        Array("9GHz", "9.05GHz", "9.1GHz", _
            "9.15GHz", "9.2GHz", _
            "9.25GHz", "9.3GHz", "9.35GHz", _
            "9.4GHz", "9.45GHz", "9.5GHz", _
            "9.55GHz", _
            "9.6GHz", "9.65GHz", "9.7GHz", _
            "9.75GHz", "9.8GHz", "9.85GHz", _
            "9.9GHz", "9.95GHz", "10GHz"),_
        "offset:=", Array("All"))

```

## UnGroupPlotCurvesInGroup

From a Stacked Plot, ungroups curves in a stack.

| <b>UI Access</b>    | N/A                                                                                                                                                                                                                                                                                        |                   |      |             |              |        |                 |             |        |                   |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|------|-------------|--------------|--------|-----------------|-------------|--------|-------------------|
| <b>Parameters</b>   | <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ReportName&gt;</td> <td>String</td> <td>Name of report.</td> </tr> <tr> <td>&lt;GroupName&gt;</td> <td>String</td> <td>Stack group name.</td> </tr> </tbody> </table> | Name              | Type | Description | <ReportName> | String | Name of report. | <GroupName> | String | Stack group name. |
| Name                | Type                                                                                                                                                                                                                                                                                       | Description       |      |             |              |        |                 |             |        |                   |
| <ReportName>        | String                                                                                                                                                                                                                                                                                     | Name of report.   |      |             |              |        |                 |             |        |                   |
| <GroupName>         | String                                                                                                                                                                                                                                                                                     | Stack group name. |      |             |              |        |                 |             |        |                   |
| <b>Return Value</b> | None.                                                                                                                                                                                                                                                                                      |                   |      |             |              |        |                 |             |        |                   |

|                       |                                                                   |
|-----------------------|-------------------------------------------------------------------|
| <b>Python Syntax</b>  | UnGroupPlotCurvesInGroup(<ReportName>, <GroupName>)               |
| <b>Python Example</b> | oModule.UngroupPlotCurvesInGroup("S Parameter Plot 3", "Stack 1") |

|                   |                                                                  |
|-------------------|------------------------------------------------------------------|
| <b>VB Syntax</b>  | UnGroupPlotCurvesInGroup <ReportName>, <GroupName>               |
| <b>VB Example</b> | oModule.UngroupPlotCurvesInGroup "S Parameter Plot 3", "Stack 1" |

# 13 - Boundary and Excitation Module Script Commands in Maxwell

Boundary and excitation commands should be executed by the **BoundarySetup** module.

```
Set oModule = oDesign.GetModule("BoundarySetup")
oModule.CommandName <args>
```

## Conventions Used in this Chapter

<BoundName>

Type: string.

Name of a boundary.

<BoundNameArray>

Type: Array of strings

An array of the names in a boundary/excitation group.

<AssignmentObjects>

**Type: Array of strings.**

An array of object names.

<AssignmentFaces>

Type: Array of integers.

An array of face IDs. The ID of a face can be determined through the user interface using the **Modeler>Measure>Area** command. The face ID is given in the **Measure Information** dialog box.

<LineEndPoint>

---

```
Array(<double>, <double>, <double>)  
<CoordSysArray>  
    Array("NAME:CoordSysVector",  
        "Origin:=", <CoordSysPoint>,  
        "UPos:=", <LineEndPoint>)
```

### Related Topics

[General Commands Recognized by the Boundary/Excitations Module](#)

[Script Commands for Creating and Modifying Boundaries in Maxwell](#)

[Script Commands for Creating and Modifying Excitations in Maxwell](#)

## General Commands Recognized by the Boundary/Excitations Module

[AddAssignmentToBoundary](#)

[DeleteAllBoundaries](#)

[DeleteAllExcitations](#)

[DeleteBoundaries](#)

[GetBoundaryAssignment](#)

[GetBoundaries](#)

[GetBoundariesOfType](#)

[GetCoreLossEffect](#)

[GetCoreLossEffectOnField](#)

[GetDisplacementCurrent](#)[GetEddyEffect](#)[GetExcitations](#)[GetExcitationsOfType](#)[GetHybridRegions](#)[GetHybridRegionsOfType](#)[GetNumBoundaries](#)[GetNumBoundariesOfType](#)[GetNumExcitations](#)[GetNumExcitationsOfType](#)[ReassignBoundary](#)[RemoveAssignmentFromBoundary](#)[RenameBoundary](#)[ReprioritizeBoundaries](#)

## AddAssignmentToBoundary

Adds a new geometry assignment to a boundary.

|            |                                   |               |                                                                                                          |
|------------|-----------------------------------|---------------|----------------------------------------------------------------------------------------------------------|
| UI Access  | N/A                               |               |                                                                                                          |
| Parameters | Name<br><i>&lt;Assignment&gt;</i> | Type<br>Array | Description<br>Structured array.<br><br>Array ("Name:<BoundName>",<br>"Objects:=", <AssignmentObjects>), |

|                     |       |                               |
|---------------------|-------|-------------------------------|
|                     |       | "Faces:=", <AssignmentFaces>) |
| <b>Return Value</b> | None. |                               |

|                       |                                                                                          |
|-----------------------|------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | AddAssignmentToBoundary(<Assignment>)                                                    |
| <b>Python Example</b> | <pre>oModule.AddAssignmentToBoundary( (     ["NAME:PerfE1",      "Faces:=", [12]])</pre> |

|                   |                                                                                                                                           |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | AddAssignmentToBoundary <Assignment>                                                                                                      |
| <b>VB Example</b> | <pre>oModule.AddAssignmentTOBoundary Array("NAME:PerfE1", _     "Objects:=", Array("Box2", "Box3"), _     "Faces:=", Array(12, 11))</pre> |

## DeleteAllBoundaries

Deletes all boundaries.

|                     |                                                   |
|---------------------|---------------------------------------------------|
| <b>UI Access</b>    | [product] > <b>Boundaries</b> > <b>Delete All</b> |
| <b>Parameters</b>   | None.                                             |
| <b>Return Value</b> | None.                                             |

|                       |                                            |
|-----------------------|--------------------------------------------|
| <b>Python Syntax</b>  | DeleteAllBoundaries()                      |
| <b>Python Example</b> | <code>oModule.DeleteAllBoundaries()</code> |

|                   |                                          |
|-------------------|------------------------------------------|
| <b>VB Syntax</b>  | DeleteAllBoundaries                      |
| <b>VB Example</b> | <code>oModule.DeleteAllBoundaries</code> |

## DeleteAllExcitations

Deletes all excitations.

|                     |                                                    |
|---------------------|----------------------------------------------------|
| <b>UI Access</b>    | [product] > <b>Excitations</b> > <b>Delete All</b> |
| <b>Parameters</b>   | None.                                              |
| <b>Return Value</b> | None.                                              |

|                       |                                             |
|-----------------------|---------------------------------------------|
| <b>Python Syntax</b>  | DeleteAllExcitations()                      |
| <b>Python Example</b> | <code>oModule.DeleteAllExcitations()</code> |

|                   |                                           |
|-------------------|-------------------------------------------|
| <b>VB Syntax</b>  | DeleteAllExcitations                      |
| <b>VB Example</b> | <code>oModule.DeleteAllExcitations</code> |

## DeleteBoundaries

Deletes the specified boundaries and excitations.

|                     |                                                                                                                           |               |                                                   |
|---------------------|---------------------------------------------------------------------------------------------------------------------------|---------------|---------------------------------------------------|
| <b>UI Access</b>    | <b>Delete</b> command in the <b>List</b> dialog box. Click <b>[product] &gt; List</b> to open the <b>List</b> dialog box. |               |                                                   |
| <b>Parameters</b>   | Name<br><i>&lt;NameArray&gt;</i>                                                                                          | Type<br>Array | Description<br>Array of boundary condition names. |
| <b>Return Value</b> | None.                                                                                                                     |               |                                                   |

|                       |                                                                |
|-----------------------|----------------------------------------------------------------|
| <b>Python Syntax</b>  | DeleteBoundaries( <i>&lt;NameArray&gt;</i> )                   |
| <b>Python Example</b> | <code>oModule.DeleteBoundaries(["PerfE1", "WavePort1"])</code> |

|                   |                                                                    |
|-------------------|--------------------------------------------------------------------|
| <b>VB Syntax</b>  | DeleteBoundaries <i>&lt;NameArray&gt;</i>                          |
| <b>VB Example</b> | <code>oModule.DeleteBoundaries Array("PerfE1", "WavePort1")</code> |

## GetBoundaryAssignment

Gets a list of face IDs associated with the given boundary or excitation assignment.

|                     |                                     |                |                                                              |
|---------------------|-------------------------------------|----------------|--------------------------------------------------------------|
| <b>UI Access</b>    | N/A                                 |                |                                                              |
| <b>Parameters</b>   | Name<br><i>&lt;BoundaryName&gt;</i> | Type<br>String | Description<br>Name of the specified boundary or excitation. |
| <b>Return Value</b> | Array of face IDs or object IDs.    |                |                                                              |

|                      |                                                      |
|----------------------|------------------------------------------------------|
| <b>Python Syntax</b> | GetBoundaryAssignment( <i>&lt;BoundaryName&gt;</i> ) |
|----------------------|------------------------------------------------------|

|                       |                                                     |
|-----------------------|-----------------------------------------------------|
| <b>Python Example</b> | <code>oModule.GetBoundaryAssignment ("Rad1")</code> |
|-----------------------|-----------------------------------------------------|

|                  |                                                         |
|------------------|---------------------------------------------------------|
| <b>VB Syntax</b> | <code>GetBoundaryAssignment &lt;BoundaryName&gt;</code> |
|------------------|---------------------------------------------------------|

|                   |                                                   |
|-------------------|---------------------------------------------------|
| <b>VB Example</b> | <code>oModule.GetBoundaryAssignment "Rad1"</code> |
|-------------------|---------------------------------------------------|

## GetBoundaries

Gets boundary names in the current design.

|                     |                          |
|---------------------|--------------------------|
| <b>UI Access</b>    | N/A                      |
| <b>Parameters</b>   | None.                    |
| <b>Return Value</b> | Array of boundary names. |

|                      |                              |
|----------------------|------------------------------|
| <b>Python Syntax</b> | <code>GetBoundaries()</code> |
|----------------------|------------------------------|

|                       |                                       |
|-----------------------|---------------------------------------|
| <b>Python Example</b> | <code>oModule.GetBoundaries ()</code> |
|-----------------------|---------------------------------------|

|                  |                            |
|------------------|----------------------------|
| <b>VB Syntax</b> | <code>GetBoundaries</code> |
|------------------|----------------------------|

|                   |                                    |
|-------------------|------------------------------------|
| <b>VB Example</b> | <code>oModule.GetBoundaries</code> |
|-------------------|------------------------------------|

## GetBoundariesOfType

Gets boundary names of the given type.

|                  |     |
|------------------|-----|
| <b>UI Access</b> | N/A |
|------------------|-----|

| Parameters   | Name                                       | Type   | Description            |
|--------------|--------------------------------------------|--------|------------------------|
|              | <BoundaryType>                             | String | Name of boundary type. |
| Return Value | Array of boundary names of the given type. |        |                        |

|                |                                          |
|----------------|------------------------------------------|
| Python Syntax  | GetBoundariesOfType(<BoundaryType>)      |
| Python Example | oModule.GetBoundariesOfType ("PerfectE") |

|            |                                        |
|------------|----------------------------------------|
| VB Syntax  | GetBoundariesOfType <BoundaryType>     |
| VB Example | oModule.GetBoundariesOfType "PerfectE" |

## GetCoreLossEffect

**Use:** Returns whether the core loss is included in the simulation for the object. Applicable to 2D and 3D Transient and to 3D Eddy Current solution types.

**Command:** None

**Syntax:** GetCoreLossEffect(<ObjectName>)

**Return Value:** Returns true or false based on the check box selection. True if checked.

**Parameters:** <ObjectName>

Type:<string>

Object name.

For example: "inner\_arm"

**Example:** oModule.GetCoreLossEffect("inner\_arm")

### GetCoreLossEffectOnField

**Use:** Returns whether the core loss effect on the field is being considered. Applicable to 2D and 3D Transient solution types.

**Command:** None

**Syntax:** GetCoreLossEffectOnField()

**Return Value:** Returns true or false based on the check box selection. True if checked.

**Parameters:** None

**Example:** oModule.GetCoreLossEffectOnField()

### GetDisplacementCurrent

**Use:** Returns whether the displacement current calculation is activated for the object. Applicable to Eddy Current solution type.

**Command:** None

**Syntax:** GetDisplacementCurrent(<ObjectName>)

**Return Value:** Returns true or false based on the check box selection. True if checked.

**Parameters:** <ObjectName>

Type:<string>

Object name.

For example: "coil1"

**Example:** oModule.GetDisplacementCurrent("coil1")

### GetEddyEffect

**Use:** Returns whether the eddy effect is activated for the object. Applicable to 2D and 3D Transient and to Eddy Current solution types.

**Command:** None

**Syntax:** GetEddyEffect(<ObjectName>)

**Return Value:** Returns true or false based on the given object name. True if checked.

**Parameters:** <ObjectName>

Type:<string>

Object name.

For example: "inner\_arm"

**Example:** oModule.GetEddyEffect ("outer\_arm")

## GetExcitations

Gets excitation port and terminal names for a model.

|                     |                                                       |
|---------------------|-------------------------------------------------------|
| <b>UI Access</b>    | N/A                                                   |
| <b>Parameters</b>   | None.                                                 |
| <b>Return Value</b> | Array of excitation name paired with excitation type. |

|                       |                           |
|-----------------------|---------------------------|
| <b>Python Syntax</b>  | GetExcitations()          |
| <b>Python Example</b> | oModule.GetExcitations () |

|                   |                        |
|-------------------|------------------------|
| <b>VB Syntax</b>  | GetExcitations         |
| <b>VB Example</b> | oModule.GetExcitations |

## GetExcitationsOfType

Gets excitation names of the given type.

| <b>UI Access</b>    | N/A                                                                                                                                                                                                                    |                          |      |             |                  |        |                          |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|------|-------------|------------------|--------|--------------------------|
| <b>Parameters</b>   | <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ExcitationType&gt;</td> <td>String</td> <td>Name of excitation type.</td> </tr> </tbody> </table> | Name                     | Type | Description | <ExcitationType> | String | Name of excitation type. |
| Name                | Type                                                                                                                                                                                                                   | Description              |      |             |                  |        |                          |
| <ExcitationType>    | String                                                                                                                                                                                                                 | Name of excitation type. |      |             |                  |        |                          |
| <b>Return Value</b> | Array of excitation names of the given type.                                                                                                                                                                           |                          |      |             |                  |        |                          |

|                       |                                                         |
|-----------------------|---------------------------------------------------------|
| <b>Python Syntax</b>  | GetExcitationsOfType(<ExcitationType>)                  |
| <b>Python Example</b> | <code>oModule.GetExcitationsOfType ("Wave Port")</code> |

|                   |                                                       |
|-------------------|-------------------------------------------------------|
| <b>VB Syntax</b>  | GetExcitationsOfType <ExcitationType>                 |
| <b>VB Example</b> | <code>oModule.GetExcitationsOfType "Wave Port"</code> |

## GetNumBoundaries

Gets the number of boundaries in a design.

|                     |                               |
|---------------------|-------------------------------|
| <b>UI Access</b>    | N/A                           |
| <b>Parameters</b>   | None.                         |
| <b>Return Value</b> | Integer number of boundaries. |

|                       |                                         |
|-----------------------|-----------------------------------------|
| <b>Python Syntax</b>  | GetNumBoundaries()                      |
| <b>Python Example</b> | <code>oModule.GetNumBoundaries()</code> |

|                   |                                       |
|-------------------|---------------------------------------|
| <b>VB Syntax</b>  | GetNumBoundaries                      |
| <b>VB Example</b> | <code>oModule.GetNumBoundaries</code> |

## GetNumBoundariesOfType

Gets the number of boundaries of the given type.

| <b>UI Access</b>                  | N/A                                                                                                                                                                                                                |                          |  |      |      |             |                                   |        |                          |
|-----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|--|------|------|-------------|-----------------------------------|--------|--------------------------|
| <b>Parameters</b>                 | <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><code>&lt;BoundaryType&gt;</code></td><td>String</td><td>Specified boundary type.</td></tr></tbody></table> |                          |  | Name | Type | Description | <code>&lt;BoundaryType&gt;</code> | String | Specified boundary type. |
| Name                              | Type                                                                                                                                                                                                               | Description              |  |      |      |             |                                   |        |                          |
| <code>&lt;BoundaryType&gt;</code> | String                                                                                                                                                                                                             | Specified boundary type. |  |      |      |             |                                   |        |                          |
| <b>Return Value</b>               | Integer number of boundaries.                                                                                                                                                                                      |                          |  |      |      |             |                                   |        |                          |

|                       |                                                          |
|-----------------------|----------------------------------------------------------|
| <b>Python Syntax</b>  | GetNumBoundariesOfType(<BoundaryType>)                   |
| <b>Python Example</b> | <code>oModule.GetNumBoundariesOfType ("PerfectE")</code> |

|                   |                                                        |
|-------------------|--------------------------------------------------------|
| <b>VB Syntax</b>  | GetNumBoundariesOfType <BoundaryType>                  |
| <b>VB Example</b> | <code>oModule.GetNumBoundariesOfType "PerfectE"</code> |

## GetNumExcitations

Gets the number of excitations in a design.

|                     |                                |
|---------------------|--------------------------------|
| <b>UI Access</b>    | N/A                            |
| <b>Parameters</b>   | None.                          |
| <b>Return Value</b> | Integer number of excitations. |

|                       |                                          |
|-----------------------|------------------------------------------|
| <b>Python Syntax</b>  | <code>GetNumExcitations()</code>         |
| <b>Python Example</b> | <code>oModule.GetNumExcitations()</code> |

|                   |                                        |
|-------------------|----------------------------------------|
| <b>VB Syntax</b>  | <code>GetNumExcitations</code>         |
| <b>VB Example</b> | <code>oModule.GetNumExcitations</code> |

## GetNumExcitationsOfType

Gets the number of excitations of the given type, including all defined modes and terminals of ports.

|                     |                                       |                |                                              |
|---------------------|---------------------------------------|----------------|----------------------------------------------|
| <b>UI Access</b>    | N/A                                   |                |                                              |
| <b>Parameters</b>   | Name<br><i>&lt;ExcitationType&gt;</i> | Type<br>String | Description<br>Specified type of excitation. |
| <b>Return Value</b> | Integer number of excitations.        |                |                                              |

|                       |                                                          |
|-----------------------|----------------------------------------------------------|
| <b>Python Syntax</b>  | GetNumExcitationsOfType(< <i>ExcitationType</i> >)       |
| <b>Python Example</b> | <code>oModule.GetNumExcitationsOfType ("Voltage")</code> |

|                   |                                                        |
|-------------------|--------------------------------------------------------|
| <b>VB Syntax</b>  | GetNumExcitationsOfType < <i>ExcitationType</i> >      |
| <b>VB Example</b> | <code>oModule.GetNumExcitationsOfType "Voltage"</code> |

## ReassignBoundary

Specifies a new geometry assignment for a boundary.

|                                    |                                                                                                                                                                                                                                                                                                  |                                                                                                                                                            |               |                                                                                                                                                            |
|------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>UI Access</b>                   | Right-click <b>Boundaries</b> > <b>Reassign or Excitations</b> > <b>Reassign</b>                                                                                                                                                                                                                 |                                                                                                                                                            |               |                                                                                                                                                            |
| <b>Parameters</b>                  | <table border="1"><tr><td>Name<br/>&lt;<i>AssignmentArray</i>&gt;</td><td>Type<br/>Array</td><td>Description<br/>Structured array.<br/><br/>Array ("Name:&lt;BoundName&gt;",<br/>"Objects:=", &lt;<i>AssignmentObjects</i>&gt;,<br/>"Faces:=", &lt;<i>AssignmentFaces</i>&gt;)</td></tr></table> | Name<br>< <i>AssignmentArray</i> >                                                                                                                         | Type<br>Array | Description<br>Structured array.<br><br>Array ("Name:<BoundName>",<br>"Objects:=", < <i>AssignmentObjects</i> >,<br>"Faces:=", < <i>AssignmentFaces</i> >) |
| Name<br>< <i>AssignmentArray</i> > | Type<br>Array                                                                                                                                                                                                                                                                                    | Description<br>Structured array.<br><br>Array ("Name:<BoundName>",<br>"Objects:=", < <i>AssignmentObjects</i> >,<br>"Faces:=", < <i>AssignmentFaces</i> >) |               |                                                                                                                                                            |
| <b>Return Value</b>                | None.                                                                                                                                                                                                                                                                                            |                                                                                                                                                            |               |                                                                                                                                                            |

|                       |                                              |
|-----------------------|----------------------------------------------|
| <b>Python Syntax</b>  | ReassignBoundary(< <i>AssignmentArray</i> >) |
| <b>Python Example</b> | <code>oModule.ReassignBoundary (</code><br>[ |

```

    "NAME:PerfH12",
    "Faces:=", [17]
)

```

|                   |                                                                                       |
|-------------------|---------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | ReassignBoundary <AssignmentArray>                                                    |
| <b>VB Example</b> | <pre> oModule.ReassignBoundary _ Array("NAME:PerfH12", _ "Faces:=", Array(17)) </pre> |

## RemoveAssignmentFromBoundary

Removes a geometry assignment from a boundary.

| <b>UI Access</b>    | Right-click on a boundary or an excitation, select <b>Remove from Assignment</b> .                                                                                                                                                                                                                                                                  |                                                                                                                                                     |      |             |                   |       |                                                                                                                                                     |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|------|-------------|-------------------|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Parameters</b>   | <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;AssignmentArray&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre> Array ("Name:&lt;BoundName&gt;", "Objects:=", &lt;AssignmentObjects&gt;, "Faces:=", &lt;AssignmentFaces&gt; ) </pre> </td> </tr> </tbody> </table> | Name                                                                                                                                                | Type | Description | <AssignmentArray> | Array | <p>Structured array.</p> <pre> Array ("Name:&lt;BoundName&gt;", "Objects:=", &lt;AssignmentObjects&gt;, "Faces:=", &lt;AssignmentFaces&gt; ) </pre> |
| Name                | Type                                                                                                                                                                                                                                                                                                                                                | Description                                                                                                                                         |      |             |                   |       |                                                                                                                                                     |
| <AssignmentArray>   | Array                                                                                                                                                                                                                                                                                                                                               | <p>Structured array.</p> <pre> Array ("Name:&lt;BoundName&gt;", "Objects:=", &lt;AssignmentObjects&gt;, "Faces:=", &lt;AssignmentFaces&gt; ) </pre> |      |             |                   |       |                                                                                                                                                     |
| <b>Return Value</b> | None.                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                     |      |             |                   |       |                                                                                                                                                     |

|                      |                                                 |
|----------------------|-------------------------------------------------|
| <b>Python Syntax</b> | RemoveAssignmentFromBoundary(<AssignmentArray>) |
|----------------------|-------------------------------------------------|

**Python Example**

```
oModule.RemoveAssignmentFromBoundary(  
    [  
        "NAME:Rad1",  
        "Faces:=", [11]  
    ])
```

**VB Syntax**

```
RemoveAssignmentFromBoundary <AssignmentArray>
```

**VB Example**

```
oModule.RemoveAssignmentFromBoundary _  
    Array("NAME:Rad1", _  
        "Faces:=", Array(11))
```

## RenameBoundary

Renames a boundary or excitation.

|                     |                                                                                                                                                                                                                                                                  |                                     |      |             |           |        |                                     |           |        |                            |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|------|-------------|-----------|--------|-------------------------------------|-----------|--------|----------------------------|
| <b>UI Access</b>    | Right-click a boundary in the project tree, and then click <b>Rename</b> on the shortcut menu.                                                                                                                                                                   |                                     |      |             |           |        |                                     |           |        |                            |
| <b>Parameters</b>   | <table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;OldName&gt;</td><td>String</td><td>Name of the boundary to be renamed.</td></tr><tr><td>&lt;NewName&gt;</td><td>String</td><td>New name for the boundary.</td></tr></table> | Name                                | Type | Description | <OldName> | String | Name of the boundary to be renamed. | <NewName> | String | New name for the boundary. |
| Name                | Type                                                                                                                                                                                                                                                             | Description                         |      |             |           |        |                                     |           |        |                            |
| <OldName>           | String                                                                                                                                                                                                                                                           | Name of the boundary to be renamed. |      |             |           |        |                                     |           |        |                            |
| <NewName>           | String                                                                                                                                                                                                                                                           | New name for the boundary.          |      |             |           |        |                                     |           |        |                            |
| <b>Return Value</b> | None.                                                                                                                                                                                                                                                            |                                     |      |             |           |        |                                     |           |        |                            |

**Python Syntax**

```
RenameBoundary(<OldName>, <NewName>)
```

|                       |                                                      |
|-----------------------|------------------------------------------------------|
| <b>Python Example</b> | <code>oModule.RenameBoundary ("Rad2", "Rad3")</code> |
|-----------------------|------------------------------------------------------|

|                  |                                                              |
|------------------|--------------------------------------------------------------|
| <b>VB Syntax</b> | <code>RenameBoundary &lt;OldName&gt;, &lt;NewName&gt;</code> |
|------------------|--------------------------------------------------------------|

|                   |                                                    |
|-------------------|----------------------------------------------------|
| <b>VB Example</b> | <code>oModule.RenameBoundary "Rad2", "Rad3"</code> |
|-------------------|----------------------------------------------------|

## ReprioritizeBoundaries

Specifies the order in which the boundaries and excitations are recognized by the solver. The first boundary in the list has the highest priority.

**Note:** this command is only valid if all defined boundaries and excitations appear in the list. All ports must be listed before any other boundary type.

|                     |                                                    |               |                                                                                                                                    |
|---------------------|----------------------------------------------------|---------------|------------------------------------------------------------------------------------------------------------------------------------|
| <b>UI Access</b>    | <b>[product] &gt; Boundaries &gt; Reprioritize</b> |               |                                                                                                                                    |
| <b>Parameters</b>   | Name<br><code>&lt;NewOrderArray&gt;</code>         | Type<br>Array | Description<br>Structured array.<br><br><code>Array ("NAME:NewOrder",<br/>       &lt;BoundName&gt;, &lt;BoundName&gt;, ...)</code> |
| <b>Return Value</b> | None.                                              |               |                                                                                                                                    |

|                      |                                                            |
|----------------------|------------------------------------------------------------|
| <b>Python Syntax</b> | <code>ReprioritizeBoundaries(&lt;NewOrderArray&gt;)</code> |
|----------------------|------------------------------------------------------------|

|                       |                                                                                           |
|-----------------------|-------------------------------------------------------------------------------------------|
| <b>Python Example</b> | <code>oModule.ReprioritizeBoundaries(<br/>    ["NAME:NewOrder",<br/>     "Imped1",</code> |
|-----------------------|-------------------------------------------------------------------------------------------|

```
"PerfE1",
"PerfH1"])
```

|                   |                                                                                                  |
|-------------------|--------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | ReprioritizeBoundaries <NewOrderArray>                                                           |
| <b>VB Example</b> | <pre>oModule.ReprioritizeBoundaries Array("NAME:NewOrder", _ "Imped1", "PerfE1", "PerfH1")</pre> |

## Script Commands for Creating and Modifying Boundaries in Maxwell

Following are script commands for creating and modifying boundaries that are recognized by the "BoundarySetup" module. In the following commands, all named data can be included/excluded as desired and may appear in any order.

- [AssignZeroTangentialHField](#)
- [EditZeroTangentialHField](#)
- [AssignFluxTangential](#)
- [EditFluxTangential](#)
- [AssignInsulating](#)
- [EditInsulating](#)
- [AssignSymmetry](#)
- [EditSymmetry](#)
- [AssignIndependent](#)
- [AssignDependent \(2D\)](#)
- [AssignDependent \(3D\)](#)

- [EditIndependent](#)
- [EditDependent](#)
- [AssignRadiation](#)
- [EditRadiation](#)
- [AssignImpedance](#)
- [EditImpedance](#)
- [AssignTangentialHField](#)
- [EditTangentialHField](#)
- [AssignVectorPotential](#)
- [EditVectorPotential](#)
- [AssignCylindricalHField](#)
- [EditCylindricalHField](#)
- [AssignResistiveSheet](#)
- [EditResistiveSheet](#)
- [AssignTouching](#)
- [EditTouching](#)
- [AssignThinLayer](#)
- [EditThinLayer](#)

## AssignCylindricalHField

**Use:** Creates an H-Field boundary on a cylindrical face.

**Command:** Maxwell>Boundaries>Assign>Tangential H Field

**Syntax:** AssignCylindricalHField <CylindricalHFieldArray>

**Return Value:** None

**Parameters:** <CylindricalHFieldArray>

```
Array("NAME:<BoundName>",
"ComponentPhiReal:=", <value>,
"ComponentPhiImg:=", <value>,
"ComponentZReal:=", <value>,
"ComponentZImg:=", <value>,
"ReverseZ:=", <bool>,
"Origin:=", <LineEndPoint>)
"Objects:=", <AssignmentObjects>,
"Faces:=", <AssignmentObjects>)
```

**Example:** oModule.AssignCylindricalHField Array("NAME:TangentialHField4", "ComponentPhiReal:=", \_  
"1", "ComponentPhiImg:=", "2", "ComponentZReal:=", "3", \_ "ComponentZImg:=", "4", "ReverseZ:=", \_  
false, \_ "ReferencePosDefined:=", true, "Origin:=", \_ Array("1.36568542494924mm", "0.6mm",  
"0mm"), \_  
"Faces:=", Array(75))

## AssignDependent (2D)

**Use:** Creates a dependent boundary.

**Command:** Maxwell2D>Boundaries>Assign>Matching>Dependent

**Syntax:** AssignDependent ([DependentArray])

**Return Value:** None

**Parameters:** <DependentArray>

```
[ "NAME:<BoundName>",
```

Either:

- "Edges:=" , <Edge ID list>, the IDs of the edges which the boundary is assigned to.
- "Objects:=" , <list of objects>, the name(s) of the polylines the boundary is assigned to.

```
"ReverseU:=" , <bool>,  
"Independent:=" , <string, name of independent>,  
"RelationIsSame:=" , <bool>  
(where:bool == True for Hdep = Hind, False for Hdep = -Hind, and so on)  
]
```

***Example command applied to polyline:***

```
oModule.AssignDependent(["NAME:Dependent1",  
"Objects:=" , ["Dependent"],  
"ReverseU:=" , False,  
"Independent:=" , "Independent1",  
"RelationIsSame:=" , False])
```

***Example command applied to edge:***

```
oModule.AssignDependent(["NAME:Dependent1",  
"Edges:=" , [8],  
"ReverseU:=" , True,  
"Independent:=" , "Independent1",  
"RelationIsSame:=" , False])
```

## AssignDependent (3D)

**Use:** Creates a dependent boundary.

**Command:** Maxwell3D>Boundaries>Assign>Matching>Dependent

**Syntax:** AssignDependent ([DependentArray])

**Return Value:** None

**Parameters:** <DependentArray>

```
[ ("NAME:<BoundName>",

Either:
  • "Faces:=" , <Face ID list>, the IDs of the faces which the boundary is assigned to.
  • "Objects:=" <list of objects>, the name(s) of the sheets the boundary is assigned to.

<CoordSysArray>, ["NAME:CoordSysVector", "Origin:=", <list of coordinates>, "UPos:=", <list of
coordinates>,]

"ReverseV:=" , <bool>,
"Independent:=" , <string, name of independent>,
"RelationIsSame:=" , <bool>,
//(bool == True for Hdep = Hind, bool == False for Hdep = -Hind, and so on)]
```

**Example command applied to sheet:**

```
oModule.AssignDependent(["NAME:Dependent1",
"Objects:=" , ["DependentSheet"],
["NAME:CoordSysVector","Origin:=" , ["0mm","0mm","0mm"],
"UPos:=" , ["3.67394039744206e-15mm","60mm","0mm"] ],
"ReverseV:=" , True,
"Independent:=" , "Independent1",
"RelationIsSame:=" , False])
```

**Example command applied to face:**

```
oModule.AssignDependent(["NAME:Dependent1",
"Faces:=" , [9],
["NAME:CoordSysVector","Origin:=" , ["0.6mm",-0.6mm,"1mm"],
"UPos:=" , [-0.8mm,-0.6mm,"0mm"] ],
"ReverseV:=" , False,
"Independent:=" , "Independent1",
"RelationIsSame:=" , False])
```

## AssignFluxTangential

Assigns a Flux Tangential boundary condition to the selected face(s).

|                     |                                   |        |                                      |
|---------------------|-----------------------------------|--------|--------------------------------------|
| <b>UI Access</b>    | Boundaries>Assign>Flux Tangential |        |                                      |
| <b>Parameters</b>   | Name                              | Type   | Description                          |
|                     | <NAME>                            | string | Name of the flux tangential boundary |
| <b>Return Value</b> | None                              |        |                                      |

|                       |                                                                                                                     |
|-----------------------|---------------------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | AssignFluxTangential (<NAME>, <Faces>)                                                                              |
| <b>Python Example</b> | <pre>oModule.AssignFluxTangential(     [         "NAME:FluxTangential2",         "Faces:=", [285, 313]     ])</pre> |

|                   |                                                                                                       |
|-------------------|-------------------------------------------------------------------------------------------------------|
| <b>VB Example</b> | <pre>oModule.AssignFluxTangential Array("NAME:FluxTangential2",     "Faces:=", Array(285, 313))</pre> |
|-------------------|-------------------------------------------------------------------------------------------------------|

## AssignImpedance (Maxwell)

**Use:** Creates an impedance boundary.

**Command:** Maxwell3D or Maxwell2D>Boundaries>Assign>Impedance

**Syntax:** AssignImpedance <ImpedanceArray>

**Return Value:** None

**Parameters:** <ImpedanceArray>

```
Array("NAME:<BoundName>",
"Faces:=", <AssignmentFaces>)
"Conductivity:=", <value>,
"Permeability:=", <value>,
"UseMaterial:=", <true or false>,
"MaterialName:=", <Material name>,
"NonlinearCoefficient:=", <number>,
"IsPermeabilityNonlinear:=", <true or false>,
"Objects:=", <AssignmentObjects>,
```

**Example:** oModule.AssignImpedance Array("NAME:Impedance5", "Faces:=", Array(29), "Conductivity:=",
"2128000", "Permeability:=", "1", "UseMaterial:=", true, "MaterialName:=", \_"Alnico5", "Non-
linearCoefficient:=", 5, "IsPermeabilityNonlinear:=", true)

## AssignIndependent (Maxwell)

**Use:** Creates an independent boundary.

**Command:** Maxwell3D or Maxwell2D>Boundaries>Assign>Matching>Independent

**Syntax:** AssignIndependent <IndependentArray>

**Return Value:** None

**Parameters:** <IndependentArray>

```
Array("NAME:<BoundName>",
<CoordSysArray>,
"ReverseV:=", <bool>,
Either:
```

- For 2D - <Edge ID list>: the IDs of the edges which the boundary is assigned to. The value is a list of edge numbers.
- For 3D - "Faces:=", <AssignmentFaces>,
- <Object>: the name(s) of the objects the boundary is assigned to.

For 2D, objects are Polylines. For 3D, objects can be sheets.

***Example AssignIndependent 2D:***

```
oModule.AssignIndependent(["NAME:Independent1",
"Edges:=", [20], "ReverseU:=", False])
```

***Example command applied to polyline:***

```
oModule.AssignIndependent(["NAME:Independent1",
"Objects:=", ["Independent"], "ReverseU:=", False])
```

***Example AssignIndependent 3D:***

```
oModule.AssignIndependent(["NAME:Independent1",
"Faces:=", [25], ["NAME:CoordSysVector",
"Origin:=", ["0mm", "0mm", "82.1090449266692mm"],
"UPos:=", ["0mm", "60mm", "82.1090449266692mm"]],
"ReverseV:=", False])
```

***Example command applied to sheet:***

```
oModule.AssignIndependent(["NAME:Independent1",
"Objects:=", ["IndependentSheet"],
["NAME:CoordSysVector", "Origin:=", ["0mm", "0mm", "0mm"],
"UPos:=", ["60mm", "0mm", "0mm"]],
"ReverseV:=", True])
```

## AssignInsulating

**Use:** Creates an insulating boundary.

**Command:** Maxwell3D>Boundaries>Assign>Insulating

**Syntax:** AssignInsulating <InsulatingArray>

**Return Value:** None

**Parameters:** <InsulatingArray>

```
Array("Name:<BoundName>",
"Objects:=", <AssignmentObjects>,
"Faces:=", <AssignmentFaces>
"Permittivity:=", <double value>
"Thickness:=", <value><unit>
```

**Example:** oModule.AssignInsulating Array("NAME:Insulating2", "Faces:=", Array(9), "Permittivity:=", "3", "Thickness:=", "5mm")

## AssignRadiation (Maxwell)

**Use:** Creates a radiation boundary.

**Command:** Maxwell3D>Boundaries>Assign>Radiation

**Syntax:** AssignRadiation <RadiationArray>

**Return Value:** None

**Parameters:** <RadiationArray>

```
Array("NAME:<BoundName>",


```

```
"Objects:=", <AssignmentObjects>,
"Faces:=", <AssignmentFaces>)
```

**Example:** oModule.AssignRadiation Array("NAME:Radiation1", \_ "Faces:=", Array(12, 11))

## AssignResistiveSheet

**Use:** For Maxwell 3D Eddy Current and Transient designs, creates a resistive sheet boundary between two conductors.

| UI Access    | Maxwell 3D>Boundaries>Assign>Resistive Sheet |               |                                                                                                                                                       |
|--------------|----------------------------------------------|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameters   | Name<br><Res-<br>istiveSheetArray>           | Type<br>Array | Description<br>For 3D Eddy Current and Transient designs: Array containing the boundary name, boundary face ID array, and resistance value with unit. |
| Return Value | None                                         |               |                                                                                                                                                       |

### For 3D Eddy Current and Transient designs

|                |                                                                                                                                               |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| Python Syntax  | AssignResistiveSheet ([<NAME>, <Faces>, <Resistance>])                                                                                        |
| Python Example | <pre>oModule.AssignResistiveSheet(     [         "NAME:ResistiveSheet2",         "Faces:=", [93],         "Resistance:=", "2ohm"     ])</pre> |

|            |                                                                                                                          |
|------------|--------------------------------------------------------------------------------------------------------------------------|
| VB Syntax  | AssignResistiveSheet Array (<NAME>, <Faces>, <Resistance>)                                                               |
| VB Example | <pre>oModule.AssignResistiveSheet Array ("NAME:ResistiveSheet4", "Faces:=", Array(93),     "Resistance:=", "2ohm")</pre> |

## AssignSymmetry (Maxwell)

**Use:** Creates a symmetry boundary.

**Command:** Maxwell3D or Maxwell2D>Boundaries>Assign>Symmetry

**Syntax:** AssignSymmetry <SymmetryArray>

**Return Value:** None

**Parameters:** <SymmetryArray>

```
Array( "NAME:<BoundName>",
"IsOdd:=",<bool>, //true for odd, false for even
"Objects:=", <AssignmentObjects>,
"Faces:=", <AssignmentFaces>)
```

**Example:** oModule.AssignSymmetry Array("NAME:Symmetry1", \_ "IsOdd:=", true, "Faces:=", Array(35))

## AssignTangentialHField

**Use:** Creates an H-Field boundary on a planar face.

**Command:** Maxwell3D>Boundaries>Assign>Tangential H Field

**Syntax:** AssignTangentialHField <TangentialHFieldArray>

**Return Value:** None

**Parameters:** <TangentialHFieldArray>

```
Array("NAME:<BoundName>",
"ComponentXReal:=", <value>,
"ComponentXImg:=", <value>,
```

```

"ComponentYReal:=", <value>,
"ComponentYImg:=", <value>,
<CoordSysArray>
"ReverseV:=", <bool>,
"Objects:=", <AssignmentObjects>,
"Faces:=", <AssignmentObjects>

```

**Example:** oModule.AssignTangentialHField Array("NAME:TangentialHField2", "ComponentXReal:=", \_  
 "1", "ComponentXImg:=", "2", "ComponentYReal:=", "3", "ComponentYImg:=", "4", Array  
 ("NAME:CoordSysVector", "Origin:=", Array( "-0.4mm", "-1.4mm", "1mm"), "UPos:=", \_  
 Array("-0.4mm", "-0.8mm", "1mm")), "ReverseV:=", true, "Faces:=", Array(7))

## AssignThinLayer

**Use:** For Maxwell 3D AC Conduction, 3D DC Conduction, 3D Magnetostatic, 3D Eddy Current, and 3D Transient designs, creates a thin layer boundary between two conductors.

| UI Access  | Maxwell 3D>Boundaries>Assign>Thin Layer |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|------------|-----------------------------------------|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameters | Name<br><ThinLayerArray>                | Type<br>Array | <p>Description</p> <ul style="list-style-type: none"> <li>For 3D AC Conduction designs: Array containing the boundary name, boundary face ID array, and permittivity, conductivity, thickness value with unit.</li> <li>For 3D DC Conduction designs: Array containing the boundary name, boundary face ID array, and conductivity and thickness value with unit.</li> <li>For 3D Magnetostatic, 3D Eddy Current, and 3D Transient (T-Omega only): Create an air gap boundary between two non-conductors. Array containing the boundary name, and thickness value with unit.</li> </ul> |

|                     |      |
|---------------------|------|
| <b>Return Value</b> | None |
|---------------------|------|

**For 3D AC Conduction designs**

|                       |                                                                                                                                                                                                           |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | AssignThinLayer ([<NAME>, <Faces>, <Permittivity>, <Conductivity>, <Thickness>])                                                                                                                          |
| <b>Python Example</b> | <pre> oModule.AssignThinLayer(     [         "NAME:ThinLayer1",         "Faces:=", [40],         "Permittivity:=", "1",         "Conductivity:=", "1S_per_m",         "Thickness:=", "1mm"     ] ) </pre> |

|                   |                                                                                                                                                               |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | AssignThinLayer Array (<NAME>, <Faces>, <Permittivity>, <Conductivity>, <Thickness>)                                                                          |
| <b>VB Example</b> | <pre> oModule.AssignThinLayer Array("NAME:ThinLayer1", "Faces:=", Array40), "Permittivity:=", "1", "Conductivity:=", "1S_per_m", "Thickness:=", "1mm") </pre> |

**For 3D DC Conduction designs**

|                       |                                                                                                                                                                            |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | AssignThinLayer ([<NAME>, <Faces>, <Conductivity>, <Thickness>])                                                                                                           |
| <b>Python Example</b> | <pre> oModule.AssignThinLayer(     [         "NAME:ThinLayer1",         "Faces:=", [40],         "Conductivity:=", "1S_per_m",         "Thickness:=", "1mm"     ] ) </pre> |

|                   |                                                                                                                           |
|-------------------|---------------------------------------------------------------------------------------------------------------------------|
|                   | ])                                                                                                                        |
| <b>VB Syntax</b>  | AssignThinLayer Array (<NAME>, <Faces>, <Conductivity>, <Thickness>)                                                      |
| <b>VB Example</b> | oModule.AssignThinLayer Array("NAME:ThinLayer1", "Faces:=", Array40), "Conductivity:=", "1S_per_m", "Thickness:=", "1mm") |

### For 3D Magnetostatic, 3D Eddy Current, and 3D Transient designs

|                       |                                                                                    |
|-----------------------|------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | AssignThinLayer ([<NAME>, <Thickness>])                                            |
| <b>Python Example</b> | oModule.AssignThinLayer (<br>[<br>"NAME:ThinLayer1",<br>"Thickness:=", "1mm"<br>]) |
| <b>VB Syntax</b>      | AssignThinLayer Array (<NAME>, <Thickness>)                                        |
| <b>VB Example</b>     | oModule.AssignThinLayer Array("NAME:ThinLayer1", "Thickness:=", "1mm")             |

### AssignTouching

**Use:** For Maxwell 3D Transient designs with translational motion with a moving part touching stationary part, assigns a touching boundary condition to a face of the moving part that touches a stationary part.

|                   |                                                       |        |                          |
|-------------------|-------------------------------------------------------|--------|--------------------------|
| <b>UI Access</b>  | <b>Maxwell 3D&gt;Boundaries&gt;Assign&gt;Touching</b> |        |                          |
| <b>Parameters</b> | Name                                                  | Type   | Description              |
|                   | <NAME>                                                | string | The name of the boundary |
|                   | <Faces>                                               | list   | Array of face IDs        |

|                     |      |
|---------------------|------|
| <b>Return Value</b> | None |
|---------------------|------|

|                       |                                                                                                       |
|-----------------------|-------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | AssignTouching (<NAME>, <Faces>)                                                                      |
| <b>Python Example</b> | <pre> oModule.AssignTouching(     [         "NAME:Touching3",         "Faces:=", [210]     ] ) </pre> |

|                   |                                                                                    |
|-------------------|------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | AssignTouching (<NAME>, <Faces>)                                                   |
| <b>VB Example</b> | <pre> oModule.AssignTouching Array("NAME:Touching3", "Faces:=", Array(210)) </pre> |

## AssignVectorPotential

**Use:** For Maxwell 2D Eddy Current, Magnetostatic, and Transient designs, assigns a vector potential boundary condition to the specified edges.

| UI Access         | Maxwell 2D>Boundaries>Assign>Vector Potential |        |                                                                                 |
|-------------------|-----------------------------------------------|--------|---------------------------------------------------------------------------------|
| <b>Parameters</b> | Name                                          | Type   | Description                                                                     |
|                   | <NAME>                                        | string | The name of the boundary                                                        |
|                   | <Edges>                                       | list   | Array of edge IDs                                                               |
|                   | <Phase>                                       | string | Phase angle with unit (for 2D Eddy Current designs only)                        |
|                   | <Value>                                       | string | Vector potential value in weber/m                                               |
|                   | <Coordin-                                     | string | Name of the reference coordinate system if the vector potential value is a spa- |

|                     |                                |                                     |
|---------------------|--------------------------------|-------------------------------------|
|                     | <code>&lt;ateSystem&gt;</code> | tial dependent function such as z+2 |
| <b>Return Value</b> | None                           |                                     |

|                       |                                                                                                                                                                                                                |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | <code>AssignVectorPotential (&lt;NAME&gt;, &lt;Edges&gt;, &lt;Phase&gt;, &lt;Value&gt;, &lt;CoordinateSystem&gt;)</code>                                                                                       |
| <b>Python Example</b> | <pre>oModule.AssignVectorPotential(     [         "NAME:VectorPotential2",         "Edges:=", [21],         "Phase:=", "60deg",         "Value:=", "Z+2",         "CoordinateSystem:=", "Global"     ] )</pre> |

|                   |                                                                                                                                                                             |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>VB Syntax</b>  | <code>AssignVectorPotential (&lt;NAME&gt;, &lt;Edges&gt;, &lt;Phase&gt;, &lt;Value&gt;, &lt;CoordinateSystem&gt;)</code>                                                    |
| <b>VB Example</b> | <pre>oModule.AssignVectorPotential Array("NAME:VectorPotential3",     "Edges:=", Array(21), "Phase:=", "60deg", "Value:=", "Z+2"),     "CoordinateSystem:=", "Global"</pre> |

## AssignZeroTangentialHField

**Use:** Creates a zero tangential H-Field boundary.

**Command:** Maxwell>Boundaries>Assign>Zero Tangential H Field

**Syntax:** `AssignZeroTangentialHField <ZeroTangentialHFieldArray>`

**Return Value:** None

**Parameters:** <ZeroTangentialHFieldArray>

```
Array("NAME:<BoundName>",
"Objects:=", <AssignmentObjects>,
"Faces:=", <AssignmentFaces>)
```

**Example:** oModule.AssignZeroTangentialHField Array("NAME:ZeroTangentialHField1", "Faces:=", Array(7))

## EditCylindricalHField

**Use:** Edits a cylindrical H-Field boundary.

**Command:** Double-click the boundary in the project tree to edit it.

**Syntax:** EditCylindricalHField <BoundName> <CylindricalHFieldArray>

**Return Value:** None

## Script Commands for Creating and Modifying Excitations

### EditDependent (Maxwell)

**Use:** Modifies a dependent boundary.

**Command:** Double-click the boundary in the project tree to modify its settings.

**Syntax:** EditDependent <BoundName> <DependentArray>

**Return Value:** None

**Note:** Refer to [AssignDependent \(2D\)](#) and [AssignDependent \(3D\)](#) examples for syntax details.

## EditFluxTangential

Edits a Flux Tangential boundary condition name.

| <b>UI Access</b>    | Double-click the Flux Tangential Boundaries item in the Project tree to edit the name.                                                                                                              |                 |  |      |      |             |        |        |                 |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|--|------|------|-------------|--------|--------|-----------------|
| <b>Parameters</b>   | <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;NAME&gt;</td> <td>string</td> <td>The edited name</td> </tr> </tbody> </table> |                 |  | Name | Type | Description | <NAME> | string | The edited name |
| Name                | Type                                                                                                                                                                                                | Description     |  |      |      |             |        |        |                 |
| <NAME>              | string                                                                                                                                                                                              | The edited name |  |      |      |             |        |        |                 |
| <b>Return Value</b> | None                                                                                                                                                                                                |                 |  |      |      |             |        |        |                 |

|                       |                                                                                                         |
|-----------------------|---------------------------------------------------------------------------------------------------------|
| <b>Python Syntax</b>  | EditFluxTangential_1 (<NAME>)                                                                           |
| <b>Python Example</b> | <pre>oModule.EditFluxTangential("FluxTangential1",     [         "NAME:FluxTangential_New"     ])</pre> |

|                   |                                                                                           |
|-------------------|-------------------------------------------------------------------------------------------|
| <b>VB Example</b> | <pre>oModule.EditFluxTangential "FluxTangential1", Array("NAME:FluxTangential_New")</pre> |
|-------------------|-------------------------------------------------------------------------------------------|

## EditImpedance (Maxwell)

**Use:** Edits an impedance boundary.

**Command:** Double-click the boundary in the project tree to edit it.

**Syntax:** EditImpedance <BoundName> <ImpedanceArray>

**Return Value:** None

**Parameters:** <BoundName>, string name of boundary

```
<ImpedanceArray>
  Array("NAME:<BoundName>",
    "Faces:=", <AssignmentFaces>)
    "Conductivity:=", <value>,
    "Permeability:=", <value>,
    "UseMaterial:=", <true or false>,
    "MaterialName:=", <Material name>,
    "NonlinearCoefficient:=", <number>,
    "IsPermeabilityNonlinear:=", <true or false>,
    "Objects:=", <AssignmentObjects>,
```

**Example:** (UseMaterial= true)

```
oModule.EditImpedance "Impedance5", Array("NAME:Impedance5", "UseMaterial:=", true, "MaterialName:=", _"Ceramic8D", "IsPermeabilityNonlinear:=", true, "NonlinearCoefficient:=", 6, "Conductivity:=", _"0.0001")
```

**Example:** (use constant; UseMaterial false)

```
oModule.EditImpedance "Impedance5", Array("NAME:Impedance5", "UseMaterial:=", false, "Permeability:=", _"2", "Conductivity:=", "3")
```

## EditIndependent (Maxwell)

**Use:** Modifies an independent boundary.

**Command:** Double-click the boundary in the Project Manager to modify its settings.

**Syntax:** Edit <BoundName> <PrimaryArray>

**Return Value:** None

**Note:** Refer to [AssignIndependent](#) examples for syntax details.

## EditInsulating

**Use:** Edits an insulating boundary.

**Command:** Double-click the boundary in the project tree to edit it.

**Syntax:** EditInsulating <BoundName> <InsulatingArray>

**Return Value:** None

**Parameters:** <BoundName> , string name of boundary  
<InsulatingArray>  
Array ("Name:<BoundName> ",  
"Permittivity:=", <double value>  
"Thickness:=", <value><unit>

**Example:** oModule.EditInsulating "Insulating2", Array("NAME:Insulating2", "Permittivity:=", "5", "Thickness:=", "7meter")

## EditRadiation (Maxwell)

**Use:** Edits a radiation boundary.

**Command:** Double-click the boundary in the project tree to edit it.

**Syntax:** EditRadiation <BoundName> <RadiationArray>

**Return Value:** None

## EditResistiveSheet

**Use:** For Maxwell 3D Eddy Current, Transient designs, edits a resistive sheet boundary.

<b>UI Access</b>	Double-click the boundary in the project tree to edit it.		
<b>Parameters</b>	Name	Type	Description
	<BoundName>	String	The name of the boundary.
<b>Return Value</b>	None		

### For 3D Eddy Current and Transient designs

<b>Python Syntax</b>	EditResistiveSheet (<BoundName>, [<NAME>, <Resistance>])
<b>Python Example</b>	<pre>oModule.EditResistiveSheet ("ResistiveSheet1",     [         "NAME:ResistiveSheet1",         "Resistance:=", "3ohm"     ] )</pre>

<b>VB Syntax</b>	EditResistiveSheet <BoundName>, Array (<NAME>, <Resistance>)
<b>VB Example</b>	<pre>oModule.EditResistiveSheet "ResistiveSheet2", Array ("NAME:ResistiveSheet2",     "Resistance:=", "3ohm")</pre>

### EditSymmetry

Modifies a symmetry boundary.

<b>UI Access</b>	Double-click the symmetry boundary in the project tree to edit its settings.
------------------	------------------------------------------------------------------------------

<b>Parameters</b>	Name	Type	Description
	<BoundaryName>	String	Name of the symmetry boundary to be edited.
	<SymmetryArray>	Array	Structured array.  Array ("NAME:<BoundName>", "IsPerfectE:=", <boolean>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EditSymmetry(<BoundaryName>, <SymmetryArray>)
<b>Python Example</b>	<pre>oModule.EditSymmetry("Sym1", ["NAME:Sym1After",  "IsPerfectE:=", True ])</pre>

<b>VB Syntax</b>	EditSymmetry <BoundaryName>, <SymmetryArray>
<b>VB Example</b>	<pre>oModule.EditSymmetry "Sym1", _ Array("NAME:Sym1After",_ "IsPerfectE:=", true)</pre>

## EditTangentialHField

**Use:** Edits a tangential H-Field boundary.

**Command:** Double-click the boundary in the project tree to edit it.

**Syntax:** EditTangentialHField <BoundName> <TangentialHFieldArray>

**Return Value:** None

## EditThinLayer

**Use:** For Maxwell 3D AC Conduction, 3D DC Conduction, 3D Magnetostatic, 3D Eddy Current, and 3D Transient designs, edit a thin layer boundary.

UI Access	Double-click the boundary in the project tree to edit it.		
Parameters	Name <BoundName>	Type String	Description The name of the boundary.
	<ThinLayerArray>	Array	<ul style="list-style-type: none"><li>For 3D AC Conduction designs: Array containing the boundary name, boundary face ID array, and permittivity, conductivity, thickness value with unit.</li><li>For 3D DC Conduction designs: Array containing the boundary name, boundary face ID array, and conductivity and thickness value with unit.</li><li>For 3D Magnetostatic, 3D Eddy Current, and 3D Transient (T-Omega only): Create an air gap boundary between two non-conductors. Array containing the boundary name, and thickness value with unit.</li></ul>
Return Value	None		
Python Syntax	EditThinLayer ( <i>BoundName</i> [<NAME>, <Faces>, <Permittivity>, <Conductivity>, <Thickness>])		
Python Example	<pre>oModule.EditThinLayer("ThinLayer1"     [         "NAME:ThinLayer1",         "Faces:=", [40],</pre>		

---

```

    "Permittivity:=", "1"
    "Conductivity:=", "1S_per_m"
    "Thickness:=", "1mm"
  ] )

```

<b>VB Syntax</b>	EditThinLayer <i>BoundName</i> Array (<NAME>, <Faces>, <Permittivity>, <Conductivity>, <Thickness>)
<b>VB Example</b>	<pre> oModule.EditThinLayer Array("NAME:ThinLayer1", "Faces:=", Array40), "Permittivity:=", "1", "Conductivity:=", "1S_per_m", "Thickness:=", "1mm") </pre>

### For 3D Magnetostatic, 3D Eddy Current, and 3D Transient designs

<b>Python Syntax</b>	EditThinLayer ( <i>BoundName</i> [<NAME>, <Thickness>])
<b>Python Example</b>	<pre> oModule.EditThinLayer ("ThinLayer1" [     "NAME:ThinLayer1",     "Thickness:=", "1mm" ]) </pre>
<b>VB Syntax</b>	EditThinLayer <i>BoundName</i> Array (<NAME>, <Thickness>)
<b>VB Example</b>	<pre> oModule.EditThinLayer Array("NAME:ThinLayer1", "Thickness:=", "1mm") </pre>

## EditTouching

**Use:** For Maxwell 3D Transient designs with translational motion with a moving part touching stationary part, edits a touching boundary condition.

<b>UI Access</b>	Double-click a touching boundary condition in the Project Manager to open the <b>Touching Boundary</b> dialog box.
------------------	--------------------------------------------------------------------------------------------------------------------

Parameters	Name	Type	Description
	<NAME>	string	The name of the boundary
Return Value	None		

Python Syntax	EditTouching (<NAME>)
Python Example	<pre>oModule.EditTouching(     [         "NAME:Touching3"     ])</pre>

VB Syntax	EditTouching (<NAME>)
VB Example	<pre>oModule.EditTouching Array ("NAME:Touching3")</pre>

## EditVectorPotential

**Use:** For Maxwell 2D Eddy Current, Magnetostatic, and Transient designs, edits a vector potential boundary condition.

UI Access	Double-click a vector potential boundary condition in the Project Manager to open the <b>Vector Potential Boundary</b> dialog box.												
Parameters	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;NAME&gt;</td><td>string</td><td>The name of the boundary</td></tr><tr><td>&lt;Edges&gt;</td><td>list</td><td>Array of edge IDs</td></tr><tr><td>&lt;Phase&gt;</td><td>string</td><td>Phase angle with unit (for 2D Eddy Current designs only)</td></tr></tbody></table>	Name	Type	Description	<NAME>	string	The name of the boundary	<Edges>	list	Array of edge IDs	<Phase>	string	Phase angle with unit (for 2D Eddy Current designs only)
Name	Type	Description											
<NAME>	string	The name of the boundary											
<Edges>	list	Array of edge IDs											
<Phase>	string	Phase angle with unit (for 2D Eddy Current designs only)											

	<code>&lt;Value&gt;</code>	string	Vector potential value in weber/m
	<code>&lt;CoordinateSystem&gt;</code>	string	Name of the reference coordinate system if the vector potential value is a spatial dependent function such as z+2
<b>Return Value</b>	None		

<b>Python Syntax</b>	<code>EditVectorPotential (&lt;NAME&gt;, &lt;Edges&gt;, &lt;Phase&gt;, &lt;Value&gt;, &lt;CoordinateSystem&gt;)</code>
<b>Python Example</b>	<pre>oModule.EditVectorPotential(     [         "NAME:VectorPotential2",         "Edges:=", [21],         "Phase:=", "60deg",         "Value:=", "z+2",         "CoordinateSystem:=", "Global"     ])</pre>

<b>VB Syntax</b>	<code>EditVectorPotential (&lt;NAME&gt;, &lt;Edges&gt;, &lt;Phase&gt;, &lt;Value&gt;, &lt;CoordinateSystem&gt;)</code>
<b>VB Example</b>	<pre>oModule.EditVectorPotential Array("NAME:VectorPotential3",     "Edges:=", Array(21), "Phase:=", "60deg", "Value:=", "z+2",     "CoordinateSystem:=", "Global")</pre>

## EditZeroTangentialHField

**Use:** Edits a zero tangential H-Field boundary.

**Command:** Double-click the boundary in the project tree to edit it.

**Syntax:** `EditZeroTangentialHField <BoundName>, <PerfectEArray>`

**Return Value:** None

## Script Commands for Creating and Modifying Excitations in Maxwell

Following are script commands for creating and modifying excitations that are recognized by the "BoundarySetup" module. In the following commands, all named data can be included/excluded as desired and may appear in any order.

- [AddTerminalsToWinding](#)
- [AssignCharge](#)
- [AssignCoilTerminal](#)
- [AssignCoilTerminalGroup](#)
- [AssignCurrent](#)
- [AssignCurrentDensity](#)
- [AssignCurrentDensityGroup](#)
- [AssignCurrentDensityTerminal](#)
- [AssignCurrentDensityTerminalGroup](#)
- [AssignCurrentGroup](#)
- [AssignFloating](#)
- [AssignSink](#)
- [AssignVoltage](#)
- [AssignVoltageAPhi](#)
- [AssignVolumeChargeDensity](#)
- [AssignVoltageDrop](#)
- [AssignVoltageDropGroup](#)

- [AssignVoltageGroup](#)
- [Assign Winding Group](#)
- [EditCharge](#)
- [EditCoilTerminal](#)
- [EditCurrent](#)
- [EditCurrentDensity](#)
- [EditCurrentDensityTerminal](#)
- [EditExternalCircuit](#)
- [EditFloating](#)
- [EditSink](#)
- [EditVoltage](#)
- [EditVoltageAPhi](#)
- [EditVolumeChargeDensity](#)
- [EditVoltageDrop](#)
- [EditWindingGroup](#)
- [SetCoreLoss](#)
- [SetEddyEffect](#)
- [SetMagnetizationCompute](#)
- [SetMinimumTimeStep](#)

## AddTerminalsToWinding

**Use:** Adds existing terminal(s) to the selected winding.

**Command:** Right-click a winding item in the project tree, and then select **Add Terminals**.

**Syntax:** AddTerminalsToWinding <AddTerminalsToWindingArray>

**Return Value:** None

**Parameters:** < AddTerminalsToWindingArray >

```
Array(" NAME:AddTerminalsToWinding", <BoundListArray>
      <BoundListArray> Array("NAME:BoundaryList",<CoilTerminalArray>)
```

*Example:*

```
oModule.AddTerminalsToWinding Array("NAME:AddTerminalsToWinding", _
Array("NAME:BoundaryList", Array("NAME:CoilTerminal1", _ 
"Point out of terminal:=", false, "Conductor number:=", _ 
"1", "Winding:=", "Winding1", "Faces:=", Array(12)), _ 
Array("NAME:CoilTerminal2", "Point out of terminal:=", _ 
false, "Conductor number:=", "1", "Winding:=", _ 
"Winding1", "Faces:=", Array(7))))
```

## AssignCharge

**Use:** Creates a charge excitation.

**Command:** Maxwell>Excitations>Assign>Charge

**Syntax:** AssignCharge <ChargeArray>

**Return Value:** None

**Parameters:** <ChargeArray>

```
Array("NAME:<BoundName>",
      "Value:=", <value>,
      "Objects:=", <AssignmentObjects>,
      "Faces:=", <AssignmentFaces>)
```

*Example:* oModule.AssignCharge Array("NAME:Charge1", "Value:=", \_ "1", "Faces:=", Array(11))

---

## AssignCoilGroup

**Use:** For Maxwell 2D designs, creates a group of coil excitations. The coil excitations can be assigned on 2D objects.

<b>UI Access</b>	Select the desired objects (coils) on the model or in the History tree. Then, select <b>Maxwell 2D&gt; Excitations&gt;Assign&gt;Coil</b> to open the <b>Coil Excitation</b> dialog where you can enter the base Name for the group of excitations being created, the number of conductors, and the polarity to be assigned to each of the new excitations.															
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;NAME&gt;</td> <td>string</td> <td>The name of the coil group</td> </tr> <tr> <td>&lt;Objects&gt;</td> <td>list</td> <td>List of objects to which the excitations are applied</td> </tr> <tr> <td>&lt;Conductor number&gt;</td> <td>int</td> <td>The number of conductors</td> </tr> <tr> <td>&lt;Polarity type&gt;</td> <td>string</td> <td>Positive or Negative</td> </tr> </tbody> </table>	Name	Type	Description	<NAME>	string	The name of the coil group	<Objects>	list	List of objects to which the excitations are applied	<Conductor number>	int	The number of conductors	<Polarity type>	string	Positive or Negative
Name	Type	Description														
<NAME>	string	The name of the coil group														
<Objects>	list	List of objects to which the excitations are applied														
<Conductor number>	int	The number of conductors														
<Polarity type>	string	Positive or Negative														
<b>Return Value</b>	None															

<b>Python Syntax</b>	AssignCoilGroup ( <i>CoilGroupArray</i> , <NAME>, <Objects>, <Conductor number>, <Polarity type>)
<b>Python Example</b>	<pre> oModule.AssignCoilGroup(["terminal_A_plus", "terminal_A_plus_1"], [{"NAME": "terminal_A_plus", "Objects:=", ["terminal_A_plus", "terminal_A_plus_1"], "Conductor number:=", "10", "PolarityType:=", Positive }])</pre>

<b>VB Syntax</b>	AssignCoilGroup <i>CoilGroupArray</i> , <NAME>, <Objects>, <Conductor number>, <Polarity type>
<b>VB Example</b>	<pre> OModule.AssignCoilGroup Array("terminal_A_plus", "terminal_A_plus_1"), Array(NAME:="terminal_A_plus", "Objects:=",</pre>

	Array("terminal_A_plus", "terminal_A_plus_1"), "Conductor number:=" , "10", "Polarity type:=", Positive)
--	-------------------------------------------------------------------------------------------------------------

## AssignCoilTerminal

**Use:** Assigns 2D terminals

**Command:** Maxwell>Excitations>Assign>Coil Terminal

**Syntax:** AssignCoilTerminal <TerminalArray>

**Parameters:** <TerminalArray>

```
Array ("NAME:AssignTerminals",
      Array ("Name:SourceList", Array ("Name:<SourceName>",
      "Excitation:=", <NetObject>, "Objects:=", <Assignment 2D>)),
      Array ("Name:SinkList", Array ("Name:<SinkName>",
      "Excitation:=", <NetObject>, "Objects:=", <Assignment 2D>),
      "Name:DeleteList", <Name Array>)
```

**Return Value:** None

**Example:** oModule.Assign2DTerminals Array ("NAME:AssignTerminals", Array ("NAME:SourceList", Array ("NAME:Source2", "Net:=", \_ "Box2", "Objects:=", Array ("Rectangle1"))), \_ Array ("NAME:SinkList", Array ("NAME:Sink1", "Net:=", \_ "Box1", "Objects:=", Array ("Rectangle2"))), \_ "DeleteList:=", "")  
oModule.AssignPerfectH Array ("NAME:PerfH1", "Faces:=", \_ Array (12))

## AssignCoilTerminalGroup

**Use:** For Maxwell 3D designs, creates a group of coil terminal excitations. The coil terminals can be assigned on the faces of 3D objects or directly on sheet objects.

<b>UI Access</b>	Select the desired objects (coils) on the model or in the History tree. Then, select <b>Maxwell 3D&gt; Excitations&gt;Assign&gt;Coil Terminal</b> to open the <b>Coil Terminal Excitation</b> dialog where you can enter the base Name for the group of excitations being created, the number of conductors, and use the Swap Direction button to set the current direction to be assigned to each of the new excitations.															
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;NAME&gt;</td> <td>string</td> <td>The name of the coil terminal group</td> </tr> <tr> <td>&lt;Objects&gt;</td> <td>list</td> <td>List of faces or sheet objects to which the excitations are applied</td> </tr> <tr> <td>&lt;Conductor number&gt;</td> <td>int</td> <td>The number of conductors</td> </tr> <tr> <td>&lt;Point out of terminal&gt;</td> <td>bool</td> <td>True sets the current direction out of the terminal</td> </tr> </table>	Name	Type	Description	<NAME>	string	The name of the coil terminal group	<Objects>	list	List of faces or sheet objects to which the excitations are applied	<Conductor number>	int	The number of conductors	<Point out of terminal>	bool	True sets the current direction out of the terminal
Name	Type	Description														
<NAME>	string	The name of the coil terminal group														
<Objects>	list	List of faces or sheet objects to which the excitations are applied														
<Conductor number>	int	The number of conductors														
<Point out of terminal>	bool	True sets the current direction out of the terminal														
<b>Return Value</b>	None															

<b>Python Syntax</b>	AssignCoilTerminalGroup ( <i>TerminalGroupArray</i> , <NAME>, <Objects>, <Conductor number>, <Point out of terminal>)
<b>Python Example</b>	<pre> oModule.AssignCoilTerminalGroup(["CoilTerminal_1", "CoilTerminal_2"], [     "NAME:CoilTerminal_1",     "Objects:=" , ["terminal_A_plus","terminal_A_plus_1"],     "Conductor number:=" , "10",     "Point out of terminal:=", False ]) </pre>

<b>VB Syntax</b>	AssignCoilTerminalGroup <i>TerminalGroupArray</i> , <NAME>, <Objects>, <Conductor number>, <Point out of terminal>
<b>VB Example</b>	<pre> OModule.AssignCoilTerminalGroup Array("CoilTerminal_1", "CoilTerminal_2"), </pre>

```
Array(NAME:CoilTerminal_1", "Objects:=",
      Array("terminal_A_plus","terminal_A_plus_1"),
      "Conductor number:=" , "10", "Point out of terminal:=", False)
```

## AssignCurrent

Creates a current source.

UI Access	Excitations>Assign>Current		
Parameters	Name <i>&lt;NAME&gt;</i>	Type string	Description The name of the excitation
	<i>&lt;Faces&gt;</i>	list	List of faces to which the excitation is applied
	<i>&lt;Current&gt;</i>	string	Current value with unit
	<i>&lt;IsSolid&gt;</i>	bool	True if type is Solid; False if Stranded
	<i>&lt;CurrentExcitationModel&gt;</i>	int	0 for Single Potential 1 for Double Potentials 2 for Double Potentials with Ground
	<i>&lt;Point out of terminal&gt;</i>	bool	True sets the current direction out of the terminal
Return Value	None		

Python Syntax	AssignCurrent_1 ( <i>&lt;NAME&gt;</i> , <i>&lt;Faces&gt;</i> , <i>&lt;Current&gt;</i> , <i>&lt;IsSolid&gt;</i> , <i>&lt;CurrentExcitationModel&gt;</i> , <i>&lt;Point out of terminal&gt;</i> )
Python Example	<pre>oModule.AssignCurrent (     [         "NAME:Current6",         "Faces:=" , [260],</pre>

```

    "Current:=" , "0mA",
    "IsSolid:=" , True,
    "CurrentExcitationModel:=", 0,
    "Point out of terminal:=", False
  ]
)

```

**VB Example**

```

oModule.AssignCurrent Array("NAME:Current6", "Faces:=", Array(952),
"Current:=", "0mA", "IsSolid:=", true, "CurrentExcitationModel:=", 0,
"Point out of terminal:=", false)

```

**AssignCurrentDensity**

**Use:** Creates a current density excitation.

**Command:** Maxwell>Excitations>Assign>Current Density

**Syntax:** AssignCurrentDensity <CurrentDensityArray>

**Return Value:** None

**Parameters:** <CurrentDensityArray>

```

Array("NAME:<BoundName>",
"CurrentDensityX:=", <value>,
"Phase:=", <value>, //for eddy current solution type
"CurrentDensityY:=", <value>,
"CurrentDensityZ:=", <bool>,
"CoordinateSystem Name:=", <string>,
"CoordinateSystem Type:=", <string>,
"Objects:=", <AssignmentObjects>)

```

**Example:** oModule.AssignCurrentDensity Array("NAME:CurrentDensity1", "CurrentDensityX:=", \_  
"1", "CurrentDensityY:=", "2", "CurrentDensityZ:=", \_ "3", "CoordinateSystem Name:=", "Global",  
\_ "CoordinateSystem Type:=", "Cartesian", "Objects:=", \_ Array("Box1"))

**Example:** oModule.AssignCurrentDensity Array("NAME:CurrentDensity1", "Phase:=", "12", \_ "Cur-  
rentDensityX:=", "1", "CurrentDensityY:=", \_  
"2", "CurrentDensityZ:=", "3", \_  
"CoordinateSystem Name:=", "Global",  
"CoordinateSystem Type:=", "Cartesian", "Objects:=", \_ Array("Box1"))

## AssignCurrentDensityGroup

**Use:** Creates a group of current density excitations.

**Command:** Maxwell>Excitations>Assign>Current Density

**Syntax:** AssignCurrentDensityGroup <BoundNameArray> <CurrentDensityArray>

**Return Value:** None

## AssignCurrentDensityTerminal

**Use:** Creates a current density terminal excitation.

**Command:** Maxwell>Excitations>Assign>Current Density Terminal

**Syntax:** AssignCurrentDensityTerminal <CurrentDensityTerminalArray>

**Return Value:** None

**Parameters:** <CurrentDensityTerminalArray>

```
Array("Name:<BoundName>",  
"Objects:=", <AssignmentObjects>,  
"Faces:=", <AssignmentFaces>)
```

**Example:** oModule.AssignCurrentDensityTerminal Array("NAME:CurrentDensityTerminal1", "Faces:=", Array(7))

## AssignCurrentDensityTerminalGroup

**Use:** Creates a group of current density terminal excitations.

**Command:** Maxwell>Excitations>Assign>Current Density Terminal

**Syntax:** AssignCurrentDensityTerminalGroup <BoundNameArray> <CurrentDensityTerminalArray>

**Return Value:** None

## AssignCurrentGroup

**Use:** Creates a group of current excitations.

**Command:** Maxwell>Excitations>Assign>Current

**Syntax:** AssignCurrentGroup <BoundNameArray> <CurrentArray>

**Return Value:** None

## AssignFloating

**Use:** Creates a floating excitation.

**Command:** Maxwell>Boundaries>Assign>Floating

**Syntax:** AssignFloating <FloatingArray>

**Return Value:** None

**Parameters:** <FloatingArray>

```
Array ("NAME:<BoundName>",
"Value:=", <value>,
"Objects:=", <AssignmentObjects>,
"Faces:=", <AssignmentFaces>)
```

**Example:** oModule.AssignFloating Array("NAME:Floating1", \_ "Value:=", "1", "Faces:=", Array(11))

## AssignSink (Maxwell)

**Use:** Creates a sink.

**Command:** Maxwell>Excitations>Assign>Sink

**Syntax:** AssignSink <SinkArray>

**Return Value:** None

**Parameters:** <SinkArray>

```
Array ("NAME:<SinkName>",
      "Faces:=", <AssignmentFaces>)
```

**Example:** oModule.AssignSink Array("NAME:Sink1", \_  
"Faces:=", Array(12))

## AssignVoltage (Maxwell)

**Use:** Creates a voltage excitation.

**Command:** Maxwell>Excitations>Assign>Voltage

**Syntax:** AssignVoltage <SourceArray>

**Return Value:** None

**Parameters:** <SourceArray>

```
Array ("NAME:<SourceName>",
      "Faces:=", <AssignmentFaces>)
```

**Example:** oModule.AssignVoltage Array("NAME:Source1", \_

```
"Faces:=", Array(12))
```

## AssignVoltageAPhi

Assigns a voltage excitation for 3D transient A-Phi solutions.

UI Access	Excitations>Assign>Voltage		
Parameters	Name	Type	Description
	<NAME>	string	The name of the voltage excitation
	<Objects>	list	List of objects to which the excitation is applied
	<VoltageAPhi>	string	Voltage value with unit
	<VoltageAPhilInitialCurrent>	string	Initial current value with unit
	<VoltageAPhiHasInitialCurrent>	bool	True if Initial Current is used.
	<VoltageAPhiExcitationModel>	int	0 for Single Potential 1 for Double Potentials 2 for Double Potentials with Ground
	<VoltageAPhi_Point_out_of_terminal>	bool	True sets the voltage direction out of the terminal
Return Value	None		

Python Syntax	AssignVoltageAPhi (<NAME>, <Objects>, <VoltageAPhi>, <VoltageAPhilInitialCurrent>, <VoltageAPhiHasInitialCurrent>, <VoltageAPhiExcitationModel>, <VoltageAPhi_Point_out_of_terminal>)
Python Example	<pre> oModule.AssignVoltageAPhi (     [         "NAME:Voltage4",         "Objects:="      , ["Cylinder1"],         "VoltageAPhi:=" , "0mV",     ] ) </pre>

```
"VoltageAPhiInitialCurrent:=", "0mA",
"VoltageAPhiHasInitialCurrent:=", False,
"VoltageAPhiExcitationModel:=", 0,
"VoltageAPhi_Point_out_of_terminal:=", False
])
```

**VB Example**

```
oModule.AssignVoltageAPhi Array("NAME:Voltage4", "Objects:=", Array("Cylinder1"),
"VoltageAPhi:=", "0mV", "VoltageAPhiInitialCurrent:=", "0mA",
"VoltageAPhiHasInitialCurrent:=", false, "VoltageAPhiExcitationModel:=", 0,
"VoltageAPhi_Point_out_of_terminal:=", false)
```

## AssignVoltageDrop

**Use:** Creates a voltage drop excitation.

**Command:** Maxwell>Excitations>Assign>Voltage Drop

**Syntax:** AssignVoltageDrop <VoltageDropArray>

**Return Value:** None

**Parameters:** <VoltageDropArray>

```
Array ("NAME:<BoundName>",
"Voltage Drop:=", <value>,
"Point out of terminal:=", <bool>,
"Objects:=", <AssignmentObjects>,
"Faces:=", <AssignmentFaces>)
```

**Example:** oModule.AssignVoltageDrop Array("NAME:VoltageDrop1", \_ "Voltage Drop:=", "1kV", "Point out of terminal:=", \_ true, "Faces:=", Array(7))

## AssignVoltageDropGroup

**Use:** Creates a group of voltage drop excitations.

**Command:** Maxwell>Excitations>Assign>Voltage Drop

**Syntax:** AssignVoltageDropGroup <BoundNameArray> <VoltageDropArray>

**Return Value:** None

## AssignVoltageGroup

**Use:** Creates a group of voltage excitations. The size of the bound name array must be identical to the size of assignment. This command is not supported for the **Electric** solution types.

**Command:** Maxwell>Excitations>Assign>Voltage

**Syntax:** AssignVoltageGroup <BoundNameArray> <VoltageArray>

**Return Value:** None

**Example:** oModule.AssignVoltageGroup Array("VoltageSrc2\_1", \_ "VoltageSrc2\_2"), Array ("NAME:VoltageSrc2\_1", \_ "Voltage:=", "0V", "Faces:=", Array(12, 11))

## AssignVolumeChargeDensity

**Use:** Creates a volume charge density excitation.

**Command:** Maxwell>Excitations>Assign>Volume Charge Density

**Syntax:** AssignVolumeChargeDensity <VolumeChargeDensityArray>

**Return Value:** None

**Parameters:** <VolumeChargeDensityArray>

    Array ("NAME:<BoundName>",

```
"Value:=", <value>,
"CoordinateSystem :=", <string>,
"Objects:=", <AssignmentObjects>)
```

**Example:** oModule.AssignVolumeChargeDensity Array("NAME:VolumeChargeDenstiyl", "Value:=", "1", \_  
"CoordinateSystem:=", "", "Objects:=", Array("Box1"))

**Example:** oModule.AssignVolumeChargeDensity Array("NAME:x", \_ "Value:=", "y", "CoordinateSystem:=", "Global", \_ "Objects:=", Array("Box2"))

## AssignWindingGroup

**Use:** Creates a winding group.

**Command:** Maxwell>Excitations>Add Winding

**Syntax:** AssignWindingGroup <WindingGroupArray>

**Return Value:** None

*Parameters:*

<WindingGroupArray>

```
Array("NAME:<BoundName>",
"Type:=", <WindingType>,
"IsSolid:=", <bool>, //true for solid, false for stranded
"Current:=", <value>,
"Resistance:=", <value>,
"Inductance:=", <value>,
"Voltage:=", <value>,
```

)

**Example:**

```
oModule.AssignWindingGroup Array("NAME:Winding1", _ "Type:=", "Voltage", "IsSolid:=", _ false, _  
"Current:=", "4A", "Resistance:=", "50Ohm", _  
"Inductance:=", "6mH", "Voltage:=", "7V")
```

**Example:** oModule.AssignWindingGroup Array("NAME:Winding2", \_ "Type:=", "Current", "IsSolid:=", \_  
false, "Current:=", \_ "1A", "Resistance:=", "0Ohm", "Inductance:=", "0mH", \_ "Voltage:=", "0V")

**Example:** oModule.AssignWindingGroup Array("NAME:Winding3", \_ "Type:=", "External", "IsSolid:=", \_  
true, "Current:=", \_ "1A", "Resistance:=", "00hm", "Inductance:=", "0mH", \_ "Voltage:=", "0V")

**EditCharge**

**Use:** Edits a charge excitation.

**Command:** Double-click the excitation in the project tree to edit it.

**Syntax:** EditCharge <BoundName> <ChargeArray>

**Return Value:** None

**EditCoilTerminal**

**Use:** Edits a coil terminal excitation.

**Command:** Double-click the excitation in the project tree to edit it.

**Syntax:** EditCoilTerminal <BoundName> <CoilTerminalArray>

**Return Value:** None

**EditCurrent**

Modifies a current source.

<b>UI Access</b>	placeholder		
<b>Parameters</b>	Name	Type	Description

	<code>&lt;NAME&gt;</code>	string	The name of the excitation
	<code>&lt;Current&gt;</code>	string	Current value with unit
	<code>&lt;IsSolid&gt;</code>	bool	True if type is Solid; False if Stranded
	<code>&lt;CurrentExcitationModel&gt;</code>	int	0 for Single Potential 1 for Double Potentials 2 for Double Potentials with Ground
	<code>&lt;Point out of terminal&gt;</code>	bool	True sets the current direction out of the terminal
<b>Return Value</b>	None		

<b>Python Syntax</b>	<code>EditCurrent_1 (&lt;NAME&gt;, &lt;Current&gt;, &lt;IsSolid&gt;, &lt;CurrentExcitationModel&gt;, &lt;Point out of terminal&gt;)</code>
<b>Python Example</b>	<pre>oModule.EditCurrent("Current7", [     "NAME:Current7",     "Current:=" , "0mA",     "IsSolid:=" , True,     "CurrentExcitationModel:=", 1,     "Point out of terminal:=", False ])</pre>

<b>VB Example</b>	<pre>oModule&gt;EditCurrent "Current7", Array("NAME:Current7", "Current:=", "0mA", "IsSolid:=", true, "CurrentExcitationModel:=", 1, "Point out of terminal:=", false)</pre>
-------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## EditCurrentDensity

**Use:** Edits a current density excitation.

**Command:** Double-click the excitation in the project tree to edit it.

**Syntax:** EditCurrentDensity <BoundName> <CurrentDensityArray>

**Return Value:** None

## EditCurrentDensityTerminal

**Use:** Edits a current density terminal excitation.

**Note:** Double-click the excitation in the project tree to edit it.

**Syntax:** EditCurrentDensityTerminal <BoundName> <CurrentDensityTerminalArray>

**Return Value:** None

## EditExternalCircuit

**Use:** Edits the external circuit for the winding. The SourceNameArray and SourceTypeArray parameters must be of the same size and be a one-to-one match.

**Command:** Maxwell>Excitations>External Circuit>Edit External Circuit

**Syntax:** EditExternalCircuit <fileName> <SourceNameArray> <SourceTypeArray>

**Return Value:** None

**Parameters:** <SourceNameArray> Array(<string>)  
<SourceTypeArray> Array(<srcTypeEnum>) <srcTypeEnum>  
1- Time dependent 2-Position dependent 3- Speed dependent

**Example:** oModule.EditExternalCircuit "C:\TestProjects\Tdslink\extnlckt00.ckt", Array("VC1", \_  
"VSA1\_0", "VSA1\_1", "VSA1\_2"), Array(1, 2, 1, 3)

## EditFloating

**Use:** Edits a floating excitation.

**Command:** Double-click the excitation in the project tree to edit it.

**Syntax:** EditFloating <BoundName> <FloatingArray>

**Return Value:** None

## EditSink

**Use:** Edits a sink excitation.

**Command:** Double-click the excitation in the project tree to edit it.

**Syntax:** EditSink <BoundName> <SinkArray>

**Return Value:** None

## EditVoltage

Modifies a voltage source.

UI Access	Double-click the excitation in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<BoundaryName>	String	Name of the voltage source to be edited.
	<VoltageArray>	Array	Structured array.  Array ("NAME:<BoundName>", "Voltage:=", <value>, <DirectionArray>)
	<DirectionArray>	Array	Structured array.

			Array ("NAME:Direction", "Start:=", <LineEndPoint>, "End:=", <LineEndPoint>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EditVoltage(<BoundaryName>, <VoltageArray>)
<b>Python Example</b>	<pre>oModule.EditVoltage("Voltage1",     ["NAME:Voltage1After",      "Voltage:=", "1000mV",      ["NAME:Direction",       "Start:=", [-0.4, -1.2, 0],       "End:=", [-1.4, -1.2, 0]      ] )</pre>

<b>VB Syntax</b>	EditVoltage <BoundaryName>, <VoltageArray>
<b>VB Example</b>	<pre>oModule.EditVoltage "Voltage1", _     Array("NAME:Voltage1After", _         "Voltage:=", "1000mV", _         Array("NAME:Direction", _</pre>

	<pre>"Start:=", Array(-0.4, -1.2, 0),_ "End:=", Array(-1.4, -1.2, 0)))</pre>
--	------------------------------------------------------------------------------

## EditVoltageAPhi

Edits a voltage excitation for 3D transient A-Phi solutions.

<b>UI Access</b>	Double-click the Voltage Excitations item in the Project tree to edit.		
<b>Parameters</b>	Name	Type	Description
	<NAME>	string	The name of the voltage excitation
	<VoltageAPhi>	string	Voltage value with unit
	<VoltageAPhiInitialCurrent>	string	Initial current value with unit
	<VoltageAPhiHasInitialCurrent>	bool	True if Initial Current is used.
	<VoltageAPhiExcitationModel>	int	0 for Single Potential 1 for Double Potentials 2 for Double Potentials with Ground
	<VoltageAPhi_Point_out_of_terminal>	bool	True sets the voltage direction out of the terminal
<b>Return Value</b>	None		

<b>Python Syntax</b>	EditVoltageAPhi (<NAME>, <VoltageAPhi>, <VoltageAPhiInitialCurrent>, <VoltageAPhiHasInitialCurrent>, <VoltageAPhiExcitationModel>, <VoltageAPhi_Point_out_of_terminal>)
<b>Python Example</b>	<pre>oModule.EditVoltageAPhi("Voltage1", [</pre>

```

"NAME:Voltage1",
"VoltageAPhi:=", "0mV",
"VoltageAPhiInitialCurrent:=", "0mA",
"VoltageAPhiHasInitialCurrent:=", True,
"VoltageAPhiExcitationModel:=", 1,
"VoltageAPhi_Point_out_of_terminal:=", True
])

```

<b>VB Example</b>	<code>oModule.EditVoltageAPhi "Voltage1", Array("NAME:Voltage1", "VoltageAPhi:=", "0mV", "VoltageAPhiInitialCurrent:=", "0mA", "VoltageAPhiHasInitialCurrent:=", false, "VoltageAPhiExcitationModel:=", 0, "VoltageAPhi_Point_out_of_terminal:=", false)</code>
-------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## EditVoltageDrop

**Use:** Edits a voltage drop excitation.

**Command:** Double-click the excitation in the project tree to edit it.

**Syntax:** `EditVoltageDrop <BoundName> <VoltageDropArray>`

**Return Value:** None

## EditVolumeChargeDensity

**Use:** Edits a volume charge density excitation.

**Command:** Double-click the excitation in the project tree to edit it.

**Syntax:** `EditVolumeChargeDensity <BoundName> <VolumeChargeDensityArray>`

**Return Value:** None

## EditWindingGroup

**Use:** Edits a winding group.

**Command:** Double-click the winding group item in the project tree to edit it.

**Syntax:** EditWindingGroup <BoundName> <WindingGroupArray>

**Return Value:** None

### **SetCoreLoss**

**Use:** Turns on the core loss effect. Please list all the objects that need to turn on the core loss effect. If an object is not in the list, then it is treated as core loss effect "off". This setting only takes effect if the object has the corresponding core loss definition in the material library.

**Command:** Maxwell>Excitation>Set Core Loss

**Syntax:** SetCoreLoss <ObjectNameArray>

**Return Value:** None

**Parameters:** < ObjectNameArray >

```
Array("<objectName>"), <bool>, //true to apply core loss effect on field, false means do not  
apply core loss effect. Default is false.
```

```
<objectName> string
```

**Example:** oModule.SetCoreLoss Array("Box1", "Box2"), true

### **SetEddyEffect**

**Use:** Sets the eddy effect for an excitation.

**Command:** Maxwell>Excitations>Set Eddy Effects

**Syntax:** SetEddyEffect <EddyEffectSettingArray>

**Return Value:** None

**Parameters:**

```
<EddyEffectSettingArray>
```

```
Array("NAME:Eddy Effect Setting", <EddyEffectVectorArray>)
```

```
<EddyEffectVectorArray>
    Array ("NAME:EddyEffectVector", Array<EddyEffectdataArray>)
<EddyEffectdataArray>
    Array (( "NAME:Data",
        "Object Name:=", <string>,
        "Eddy Effect:=", <bool>,
        "DisplacementCurrent:=", <bool>)
```

*Example:*

```
oModule.SetEddyEffect Array("NAME:Eddy Effect Setting", _
    Array("NAME:EddyEffectVector", Array("NAME:Data", _  
        "Object Name:=", "Box1", "Eddy Effect:=", true, _  
        "Displacement Current:=", true), Array("NAME:Data", _  
        "Object Name:=", "Box1_1", "Eddy Effect:=", true, _  
        "Displacement Current:=", false)))
```

**Example:** oModule.SetEddyEffect Array("NAME:Eddy Effect Setting", \_  
 Array("NAME:EddyEffectVector", Array("NAME:Data", \_  
 "Object Name:=", "Box1", "Eddy Effect:=", true), \_  
 Array("NAME:Data", "Object Name:=", "Box1\_1", \_  
 "Eddy Effect:=", false)))

## **SetMagnetizationCompute**

**Use:** Specifies Magnetization Computation Points settings.

**Command:** Maxwell > Excitations > Set Magnetization Computation

**Syntax:** SetMagnetizationCompute <MagnetizationComputationArray>

**Return Value:** None

**Parameters:** <MagnetizationComputationArray>

```
Array("NAME:<Magnetization>",
  "IsTarget:=", <bool>,
  "ForDemagComputation:=", <bool>,
  "Project:=", <string>,
  "Design:=", <string>,
  "Soln:=", <string>,
  Array("NAME:Params", )
  "ForceSourceToSolve:=", <bool>,
  "PreservePartnerSoln:=", <bool>,
  "PathRelativeTo:=", <string>,)
```

**Example:** Set oModule = oDesign.GetModule("BoundarySetup")  
oModule.SetMagnetizationCompute Array("NAME:Magnetization", "IsTarget:=", true, "ForDemagComputation:=", true, "Project:=", "test\_magnetization\_link.aedt", "Design:=", "mag\_hysteresis\_only\_source", "Soln:=", "Setup1 : Transient", Array("NAME:Params", "\$Temp:=", "20cel", "Cur:=", "0A"), "ForceSourceToSolve:=", false, "PreservePartnerSoln:=", false, "PathRelativeTo:=", "TargetProject")

### Other BoundarySetup Module Script Commands

Following are additional script commands that are recognized by the **BoundarySetup** module:

**Note:**

In the following commands, all named data can be included/excluded as desired and may appear in any order.

- [AssignWindingGroup](#)
- [EditWindingGroup](#)
- [AddTerminalsToWinding](#)
- [EditExternalCircuit](#)
- [SetCoreLoss](#)
- [SetEddyEffect](#)
- [SetMinimumTimeStep](#)

## SetMinimumTimeStep

**Use:** Sets the minimum time step for an external circuit excitation.

**Command:** Maxwell>Excitations>External Circuit>Set Minimum Time Step

**Syntax:** SetMinimumTimeStep <value>

**Return Value:** None

**Parameters:** None

**Example:** oModule.SetMinimumTimeStep "1ps"

## ShowWindow

**Use:** Sets up a Y connection among voltage windings. It will connect their negative voltage terminals to a common node of the circuit. Only one Y Connection is allowed per design.

**Syntax:** SetupYConnection <YConnectionArray>

**Return Value:** None

**Parameters:** <YConnectionArray>

Type: Array of strings

When the *YConnectionArray* is empty, it is used to delete the Y Connection.

Array(<YConnection>)

<YConnection>

Type: Array of strings

Array("Name:YConnection", "Windings:=", <WindingNames>)

<WindingNames>

Type: String

A list of winding names separated by commas

***VB Example:***

Add Y Connection:

```
oModule.SetupYConnection Array(Array("NAME:YConnection", "Windings:=", "Winding1,Winding2,Winding3,Winding4"))
```

Delete Y Connection:

```
oModule.SetupYConnection
```

Python Syntax	SetupYConnection(<YConnectionArray>)
Python Example	<p><u>Add Y Connection</u></p> <pre>oModule.SetupYConnection(     [         [             "NAME:YConnection",             "Windings:=" , "Winding1,Winding2,Winding3,Winding4"     ] )</pre>

```
    ]  
])
```

#### Delete Y Connection

```
oModule.SetupYConnection()
```

This page intentionally  
left blank.

# 14 - Mesh Operations Module Script Commands

Commands for mesh setup and operations should be executed by the "MeshSetup" module.

```
Set oModule = oDesign.GetModule("MeshSetup")
```

```
oModule.CommandName <args>
```

## Conventions Used in this Chapter

```
<OpName>
```

Type: <string>

Name of a mesh operation.

```
<AssignmentObjects>
```

**Type: Array of strings**

An array of object names.

```
<AssignmentFaces>
```

Type: Array of integers.

An array of face IDs. The ID of a face can be determined through the user interface using the **3DModeler>Measure>Area** command. The face ID is given in the **Measure Information** dialog box.

## The topics for this section include:

[General Commands Recognized by the Mesh Operations Module](#)

[Script Commands for Creating and Modifying Mesh Operations](#)

## General Commands Recognized by the Mesh Operations Module

General commands recognized by the Mesh Operations Module:

[DeleteOp](#)

[GetMeshOpAssignment](#)

[GetOperationName](#)

[RenameOp](#)

### **DeleteOp**

Deletes the specified mesh operations.

<b>UI Access</b>	<b>Delete</b> command in the <b>List</b> dialog box.
<b>Parameters</b>	Name <code>&lt;NameArray&gt;</code> Type Array Description Array of mesh operation names.
<b>Return Value</b>	None.

<b>Python Syntax</b>	<code>DeleteOp(&lt;NameArray&gt;)</code>
<b>Python Example</b>	<code>oModule.DeleteOp(["Length1", "SkinDepth1", "Length2"])</code>

<b>VB Syntax</b>	<code>DeleteOp &lt;NameArray&gt;</code>
<b>VB Example</b>	<code>oModule.DeleteOp Array("Length1", "SkinDepth1", _</code>

	"Length2")
--	------------

## GetOperationNames

Gets the names of mesh operations defined in a design.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;OperationType&gt;</td> <td>String</td> <td>Specified operation type.</td> </tr> </table>			Name	Type	Description	<OperationType>	String	Specified operation type.
Name	Type	Description							
<OperationType>	String	Specified operation type.							
<b>Return Value</b>	Array of strings containing mesh operation names.								

<b>Python Syntax</b>	GetOperationNames(<OperationType>)
<b>Python Example</b>	oModule.GetOperationNames ("Length Based")

<b>VB Syntax</b>	GetOperationNames <OperationType>
<b>VB Example</b>	oModule.GetOperationNames "Length Based"

## RenameOp

Renames a mesh operation.

<b>UI Access</b>	Right-click the mesh operation in the project tree, and then click <b>Rename</b> on the shortcut menu.											
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;OldName&gt;</td> <td>String</td> <td>Old name for the mesh operation.</td> </tr> <tr> <td>&lt;NewName&gt;</td> <td>String</td> <td>New name for the mesh operation.</td> </tr> </table>			Name	Type	Description	<OldName>	String	Old name for the mesh operation.	<NewName>	String	New name for the mesh operation.
Name	Type	Description										
<OldName>	String	Old name for the mesh operation.										
<NewName>	String	New name for the mesh operation.										

<b>Return Value</b>	None.
---------------------	-------

<b>Python Syntax</b>	RenameOp(<OldName>, <NewName>)
----------------------	--------------------------------

<b>Python Example</b>	<code>oModule.RenameOp ("SkinDepth1", "NewName")</code>
-----------------------	---------------------------------------------------------

<b>VB Syntax</b>	RenameOp <OldName>, <NewName>
------------------	-------------------------------

<b>VB Example</b>	<code>oModule.RenameOp "SkinDepth1", "NewName"</code>
-------------------	-------------------------------------------------------

## Script Commands for Creating and Modifying Mesh Operations

Script commands for creating and modifying mesh operations are as follows:

"AssignCylindricalGapOp" on the facing page

"AssignCylindricalGapOp" on the facing page

[AssignLengthOp](#)

[AssignModelResolutionOp](#)

"AssignSkinDepthLayerSetting" on page 14-17

[AssignSkinDepthOp](#)

[AssignTrueSurfOp](#)

"EditCylindricalGapOp" on page 14-22

"EditDensityControlOp" on page 14-24

[EditEdgeCutLayerOp](#)[EditLengthOp](#)[EditModelResolutionOp](#)[EditSkinDepthOp](#)[EditTrueSurfOp](#)

"InitialMeshSettings " on page 14-35

## AssignCylindricalGapOp

### 3D Designs

For 3D designs, assigns a cylindrical gap 3D mesh operation to enable use of Clone Mesh and associated Band Mapping Angle. For Maxwell 2D designs, [see below](#).

=====

Assigns a cylindrical gap 3D mesh operation to enable use of Clone Mesh and associated Band Mapping Angle.

UI Access	Maxwell3D > Mesh > Assign Mesh Operation > Cylindrical Gap Treatment		
Parameters	Name <code>&lt;CylindricalGapOpParams&gt;</code>	Type Array	Description Structured array.  <code>Array ("NAME:&lt;OpName&gt;","CloneMesh:=", &lt;bool&gt;,"BandMappingAngle:=", &lt;value&gt; [use if CloneMesh is "true"]"MovingSideLayers:=", &lt;integer&gt;,"StaticSideLayers:=", &lt;integer&gt;,"Objects:=", &lt;AssignmentObjects&gt;)</code>

<b>Return Value</b>	None.
---------------------	-------

<b>Python Syntax</b>	AssignCylindricalGapOp(<CylindricalGapOpParams>)
<b>Python Example</b>	<pre> oModule.AssignCylindricalGapOp (     "NAME:CylindricalGap1",     "CloneMesh:=", True,     "BandMappingAngle:=", 3deg,     "MovingSideLayers:=", 1,     "StaticSideLayers:=", 3,     "Objects:=", ["Box2"] ) </pre>

<b>VB Syntax</b>	AssignCylindricalGapOp <CylindricalGapOpParams>
<b>VB Example</b>	<pre> oModule.AssignCylindricalGapOp _ Array("NAME:CylindricalGap1", _       "CloneMesh:=", true, _       "BandMappingAngle:=", 3deg, _       "MovingSideLayers:=", 1, _ </pre>

```

"StaticSideLayers:=", 3, _
"Objects:=", Array("Box2"))

oModule.AssignCylindricalGapOp _
Array("NAME:CylindricalGap1", _
"CloneMesh:=", false, _
"Objects:=", Array("Box2"))

```

## Maxwell 2D Designs

For Maxwell 2D Designs, assigns a cylindrical gap 2D mesh operation.

UI Access	Maxwell 2D > Mesh > Assign Mesh Operation > Cylindrical Gap Treatment		
Parameters	Name <CylindricalGapOpParams>	Type Array	Description Structured array.  Array ("NAME:<OpName>", "UseBandMappingAngle:=", <bool>, "BandMappingAngle:=", <value> [use if UseBandMappingAngle is "true". Range is 0.0005deg through 3deg inclusive] "Objects:=", <AssignmentObjects>)
Return Value	None.		

Python Syntax	AssignCylindricalGapOp(<CylindricalGapOpParams>)
Python Example	oModule.AssignCylindricalGapOp

```
( [  
    "NAME:CylindricalGap1",  
    "UseBandMappingAngle:=" , True,  
    "BandMappingAngle:="      , "3deg",  
    "Objects:=", ["Circle2"]  
] )
```

<b>VB Syntax</b>	AssignCylindricalGapOp <CylindricalGapOpParams>
<b>VB Example</b>	<pre>oModule.AssignCylindricalGapOp Array("NAME:CylindricalGap1",     "UseBandMappingAngle:=" , True,     "BandMappingAngle:="      , "3deg",     "Objects:=", Array("Circle2"))</pre>

## AssignDensityControlOp

Assigns a density control mesh operation to specify refinement of clone mesh.

<b>UI Access</b>	<b>Maxwell3D &gt; Mesh &gt; Assign Mesh Operation &gt; Inside Selection &gt; Clone Mesh Density</b>		
<b>Parameters</b>	Name <DensityControl/Params>	Type Array	Description Structured array.  Array ("NAME:<OpName>",

			<pre>     "RefineInside:=", &lt;bool&gt;,     "Objects:=", &lt;AssignmentObjects&gt;,     "RestrictMaxElemLength:=", &lt;bool&gt;,     "MaxElemLength:=", &lt;value&gt;,     "RestrictLayersNum:=", &lt;bool&gt;,     "LayersNum:=", &lt;integer&gt;)  RefineInside  If true, Objects should be specified. Implies apply restrictions to tetrahedra inside the object.  If false, Faces and/or Objects can be spe- cified. Implies apply restrictions to triangles on the surface of the face or object.  RestrictMaxElemLength  If true, MaxElemLength should be specified.  RestrictLayersNum  If true, LayersNum should be specified. </pre>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	AssignDensityControlOp(<DensityControlParams>)
<b>Python Example</b>	<pre> oModule.AssignDensityControlOp(     [ </pre>

```
"NAME:Clone Mesh Density3",
"RefineInside:="      , True,
"Objects:="          , ["Inner_arm"],
"RestrictMaxElemLength:=", True,
"MaxElemLength:="     , "0.015mm",
"RestrictLayersNum:=" , True,
"LayersNum:="         , "15"
])
```

VB Syntax	AssignDensityControlOp <DensityControlParams>
<b>VB Example</b>	<pre>oModule.AssignDensityControlOp _ Array("NAME:Clone Mesh Density3", _ "RefineInside:=", true, _ "Objects:=", Array("Inner_arm"), _ "RestrictMaxElemLength:=", true, _ "MaxElemLength:=", "0.015mm", _ "RestrictLayersNum:=", true, _ "LayersNum:=", "15")</pre>

## AssignEdgeCutLayerOp

Assigns the edge cut mesh operation to specify refinement of layer mesh.

UI Access	Maxwell 3D > Mesh > Assign Mesh Operation > Inside Selection > Edge Cut Based		
Parameters	Name <i>&lt;EdgeCutLayerOpParams&gt;</i>	Type Array	Description Structured array.  Array ("NAME:<OpName>"," "Objects:=", <AssignmentObjects>, "Layer Thickness:=", <value>)
Return Value	None.		

Python Syntax	AssignEdgeCutLayerOp( <i>&lt;EdgeCutLayerOpParams&gt;</i> )
Python Example	<pre>oModule.AssignEdgeCutLayerOp(     [         "NAME:EdgeCut1",         "Objects:="           , ["Rotor"],         "Layer Thickness:="   , "0.55mm"     ]) </pre>

VB Syntax	AssignEdgeCutLayerOp < <i>EdgeCutLayerOpParams</i> >
VB Example	<pre>oModule.AssignEdgeCutLayerOp _</pre>

```
Array("NAME:EdgeCut1", _
      "Objects:=", Array("Rotor"), _
      "Layer Thickness:=", "0.55mm")
```

## AssignLengthOp

Assigns length-based operations to the selection.

UI Access	<b>Maxwell 3D &gt; Mesh &gt; Assign Mesh Operation &gt; On Selection   Inside Selection &gt; Length Based</b>								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;LengthOpParams&gt;</code></td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array("NAME:&lt;OpName&gt;",       "RefineInside:=", &lt;bool&gt;,       "Objects:=", &lt;AssignmentObjects&gt;,       "Faces:=", &lt;AssignmentFaces&gt;,       "Edges:=", &lt;AssignmentEdges&gt;,       "RestrictElem:=", &lt;bool&gt;       "NumMaxElem:=", &lt;integer&gt;       "RestrictLength:=", &lt;bool&gt;       "MaxLength:=", &lt;value&gt;       "ApplyToInitialMesh:=", &lt;bool&gt;)  RefineInside</pre> <p>If true, Objects should be specified. Implies apply restrictions to tet-</p> </td> </tr> </tbody> </table>	Name	Type	Description	<code>&lt;LengthOpParams&gt;</code>	Array	<p>Structured array.</p> <pre>Array("NAME:&lt;OpName&gt;",       "RefineInside:=", &lt;bool&gt;,       "Objects:=", &lt;AssignmentObjects&gt;,       "Faces:=", &lt;AssignmentFaces&gt;,       "Edges:=", &lt;AssignmentEdges&gt;,       "RestrictElem:=", &lt;bool&gt;       "NumMaxElem:=", &lt;integer&gt;       "RestrictLength:=", &lt;bool&gt;       "MaxLength:=", &lt;value&gt;       "ApplyToInitialMesh:=", &lt;bool&gt;)  RefineInside</pre> <p>If true, Objects should be specified. Implies apply restrictions to tet-</p>		
Name	Type	Description							
<code>&lt;LengthOpParams&gt;</code>	Array	<p>Structured array.</p> <pre>Array("NAME:&lt;OpName&gt;",       "RefineInside:=", &lt;bool&gt;,       "Objects:=", &lt;AssignmentObjects&gt;,       "Faces:=", &lt;AssignmentFaces&gt;,       "Edges:=", &lt;AssignmentEdges&gt;,       "RestrictElem:=", &lt;bool&gt;       "NumMaxElem:=", &lt;integer&gt;       "RestrictLength:=", &lt;bool&gt;       "MaxLength:=", &lt;value&gt;       "ApplyToInitialMesh:=", &lt;bool&gt;)  RefineInside</pre> <p>If true, Objects should be specified. Implies apply restrictions to tet-</p>							

			<p>rahedra inside the object.</p> <p>If false, Edges, Faces and/or Objects can be specified. Implies apply restrictions to triangles on the surface of the face or object.</p> <p><b>RestrictElem</b></p> <p><i>If true</i>, NumMaxElem should be specified.</p> <p><b>RestrictLength</b></p> <p><i>If true</i>, MaxLength should be specified.</p>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	AssignLengthOp(<LengthOpParams>)
<b>Python Example</b>	<pre> oModule.AssignLengthOp (     "NAME:Length1",     "RefineInside:=", True,     "Objects:=", ["Box1"],     "RestrictElem:=", True,     "NumMaxElem:=", 1000,     "RestrictLength:=", True,     "MaxLength:=", "1mm"     "ApplyToInitialMesh:=", True ) </pre>

```
oModule.AssignLengthOp  
([  
    "NAME:Length1",  
    "RefineInside:=", False,  
    "Edges:=", [58,73],  
    "RestrictElem:=", True,  
    "NumMaxElem:=", 1000,  
    "RestrictLength:=", True,  
    "MaxLength:=", "1mm"  
    "ApplyToInitialMesh:=", True  
])
```

VB Syntax	AssignLengthOp <LengthOpParams>
<b>VB Example</b>	<pre>oModule.AssignLengthOp Array("NAME:Length1", _     "RefineInside:=", true,     "Objects:=", Array("Box1"),     "RestrictElem:=", true,     "NumMaxElem:=", 1000,</pre>

```

    "RestrictLength:=", true,
    "MaxLength:=", "1mm")
    "ApplyToInitialMesh:=", True

oModule.AssignLengthOp Array ("NAME:Length1", _
    "RefineInside:=", false,
    "Edges:=", Array(58,73),
    "RestrictElem:=", true,
    "NumMaxElem:=", 1000,
    "RestrictLength:=", true,
    "MaxLength:=", "1mm")
    "ApplyToInitialMesh:=", True

```

## AssignModelResolutionOp

Assigns a model resolution name, value and unit for mesh operations, or specify to UseAutoFeaturelength. If UseAutoFeature length is true, the Defeature length is not used.

UI Access	Maxwell 3D > Mesh > Assign Mesh Operation > Model Resolution		
Parameters	Name <i>&lt;ModelResParams&gt;</i>	Type Array	Description Structured array. Array ("NAME:<ModelResName>", "Objects:=", <array of object names>, "UseAutoLength:=", <boolean>, "DefeatureLength:=", <double>)

			"DefeatureLength:=", "<value><units>")
<b>Return Value</b>	None.		

<b>Python Syntax</b>	AssignModelResolutionOp(<ModelResParams>)
<b>Python Example</b>	<pre> oModule.AssignModelResolutionOp (     "NAME:ModelResolution1",     "Objects:=", [ "waveguide" ],     "UseAutoLength:=", True,     "DefeatureLength:=", "71.5891053163818mil" ) </pre>

<b>VB Syntax</b>	AssignModelResolutionOp <ModelResParams>
<b>VB Example</b>	<pre> oModule.AssignModelResolutionOp _     Array("NAME:ModelResolution1", _         "Objects:=", Array( "waveguide"), _         "UseAutoLength:=", true, _         "DefeatureLength:=", "71.5891053163818mil") </pre>

## AssignSkinDepthLayerSetting

**Use:** Assigns a skin-depth layer settings to the selected edges.

**Command:** Click Mesh > Assign Mesh Operation > On Selection > Skin Depth Based

**Syntax:** AssignSkinDepthLayerSetting Array("NAME:<LayerSettingName>", "Enabled:=", <SettingEnabled>, "Edges:=", Array(<EdgeIDArray>), "SkinDepth:=", "<SkinDepthValue>", "NumLayers:=", "<LayerNumber>")

**Return Value:** None

**Parameters:** <LayerSettingName>

The name of the skin depth layer setting.

<SettingEnabled>

Boolean: "Enabled:=", true

<EdgeIDArray>

Array of edge IDs. "Edges:=", Array(10, 12, 13)

<SkinDepthValue>

Skin depth value with unit. "SkinDepth:=", "1mm"

<LayerNumber>

Integer, the number of layers. "NumLayers:=", "2"

**Example:** oModule.AssignSkinDepthLayerSetting Array("NAME:SkinDepthLayer1",  
"Enabled:=", true  
"Edges:=", Array(10, 12, 13),  
"SkinDepth:=", "1mm",  
"NumLayers:=", 2)

## AssignSkinDepthOp

Assigns a skin-depth based operations to the selection.

UI Access	Maxwell 3D > Mesh > Assign Mesh Operation > On Selection > Skin Depth Based		
Parameters	Name <code>&lt;SkinDepthOpParams&gt;</code>	Type Array	Description Structured array.  <code>Array ("NAME:&lt;OpName&gt;,"       "Faces:=", &lt;AssignmentFaces&gt;,       "RestrictElem:=", &lt;bool&gt;,       "NumMaxElem:=", &lt;int&gt;,       "SkinDepth:=", &lt;value&gt;,       "SurfTriMaxLength:=", &lt;value&gt;,       "NumLayers:=", &lt;int&gt;)  RestrictElem  <i>If true, NumMaxElem should be specified.</i></code>
Return Value	None.		

Python Syntax	<code>AssignSkinDepthOp(&lt;SkinDepthOpParams&gt;)</code>
Python Example	<code>oModule.AssignSkinDepthOp ([</code>

```

    "NAME:SkinDepth1",
    "Faces:=", [7],
    "RestrictElem:=", True,
    "NumMaxElem:=", 1000,
    "SkinDepth:=", "1mm",
    "SurfTriMaxLength:=", "1mm",
    "NumLayers:=", 2
)

```

<b>VB Syntax</b>	AssignSkinDepthOp < <i>SkinDepthOpParams</i> >
<b>VB Example</b>	<pre> oModule.AssignSkinDepthOp Array("NAME:SkinDepth1", _     "Faces:=", Array(7), _     "RestrictElem:=", true, _     "NumMaxElem:=", 1000, _     "SkinDepth:=", "1mm", _     "SurfTriMaxLength:=", "1mm", _     "NumLayers:=", 2) </pre>

## AssignTrueSurfOp

Assigns a true surface-based mesh operation on the selection.

**UI Access**

**Maxwell 3D > Mesh > Assign Mesh Operation > Surface Approximation**

	Name	Type	Description
<b>Parameters</b>	<code>&lt;TrueSurfOpParams&gt;</code>	Array	<p>Structured array.</p> <pre>         Array ("NAME:&lt;OpName&gt;",                "Faces:=", &lt;AssignmentFaces&gt;,                "SurfDevChoice:=", &lt;RadioOption&gt;,                "SurfDev:=", &lt;value&gt;,                "NormalDevChoice:=", &lt;RadioOption&gt;,                "NormalDev:=", &lt;value&gt;,                "AspectRatioChoice:=", &lt;RadioOption&gt;,                "AspectRatio:=", &lt;double&gt;)          &lt;RadioOption&gt;     </pre> <p><i>Type: &lt;int&gt;</i></p> <ul style="list-style-type: none"> <li>0: Ignore</li> <li>1: Use defaults</li> <li>2: Specify the value</li> </ul>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	AssignTrueSurfOp(<TrueSurfOpParams>)
----------------------	--------------------------------------

<b>Python Example</b>	<pre>oModule.AssignTrueSurfOp (     "NAME:TrueSurf1",     "Faces:=", [9],     "SurfDevChoice:=", 2,     "SurfDev:=", "0.04123105626mm",     "NormalDevChoice:=", 2,     "NormalDev:=", "15deg",     "AspectRatioChoice:=", 1 )</pre>
-----------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	<b>AssignTrueSurfOp &lt;TrueSurfOpParams&gt;</b>
<b>VB Example</b>	<pre>oModule.AssignTrueSurfOp Array("NAME:TrueSurf1",     "Faces:=", Array(9), _     "SurfDevChoice:=", 2, _     "SurfDev:=", "0.04123105626mm", _     "NormalDevChoice:=", 2, _     "NormalDev:=", "15deg", _     "AspectRatioChoice:=", 1)</pre>

## EditCylindricalGapOp

Modifies an existing cylindrical gap 3D mesh operation to enable/disable use of Clone Mesh and associated Band Mapping Angle.

Modifies an existing cylindrical gap 2D mesh operation to enable/disable use of Band Mapping Angle.

<b>UI Access</b>	Double-click the operation in the project tree to modify its settings.		
<b>Parameters</b>	Name <code>&lt;OpName&gt;</code>	Type String	Description Name of the operation to be edited.
	<code>&lt;CylindricalGapOpParams&gt;</code>	Array	Structured array for 3D mesh.  <code>Array ("NAME:&lt;OpName&gt;","CloneMesh:=", &lt;bool&gt;,"BandMappingAngle:=", &lt;value&gt; <i>[use if CloneMesh is true]</i>,"MovingSideLayers:=", &lt;integer&gt;,"StaticSideLayers:=", &lt;integer&gt;)</code>  Structured array for 2D mesh.  <code>Array ("NAME:&lt;OpName&gt;","UseBandMappingAngle:=", &lt;bool&gt;,"BandMappingAngle:=", &lt;real&gt;) <i>[range is 0.0005deg through 3deg inclusive]</i></code>
<b>Return Value</b>	None		

<b>Python Syntax</b>	EditCylindricalGapOp(<OpName>, <CylindricalGapOpParams>)
<b>Python Example</b>	<pre> oModule.EditCylindricalGapOp("CylindricalGap1",     ["NAME:CylindricalGap1",      "CloneMesh:=", True,      "BandMappingAngle:=", 3deg,      "MovingSideLayers:=", 1,      "StaticSideLayers:=", 3] ) </pre> <p><b>Maxwell 2D Example</b></p> <pre> oModule.EditCylindricalGapOp("CylindricalGap1",     ["NAME:CylindricalGap1",      "UseBandMappingAngle:=", True,      "BandMappingAngle:=", 3deg,     ] ) </pre>

<b>VB Syntax</b>	EditCylindricalGapOp <OpName>, <CylindricalGapOpParams>
<b>VB Example</b>	<pre> oModule.EditCylindricalGapOp "CylindricalGap1",     Array("NAME:CylindricalGap1",         "CloneMesh:=", true,         "BandMappingAngle:=", 3deg,         "MovingSideLayers:=", 1,     ) </pre>

```
"StaticSideLayers:=", 3)
```

### Maxwell 2D Example

```
oModule.EditCylindricalGapOp "CylindricalGap1", _  
    Array ("NAME:CylindricalGap1",  
        "UseBandMappingAngle:=", true,  
        "BandMappingAngle:=", 3deg)
```

## EditDensityControlOp

Edits an existing density control operation. This cannot be used to modify assignments. Instead, the mesh operation should be deleted and a new one created.

UI Access	Double-click the operation in the project tree to modify its settings.		
	Name <i>&lt;OpName&gt;</i>	Type String	Description Name of the operation to be edited.
Parameters	<i>&lt;DensityControlParams&gt;</i>	Array	Structured array.  Array ("NAME:<OpName>"," "RefineInside:=", <bool>, "RestrictMaxElemLength:=", <bool>, "MaxElemLength:=", <value>, "RestrictLayersNum:=", <bool>, "LayersNum:=", <integer>)  RefineInside

---

			<p>If true, Objects should be specified. Implies apply restrictions to tetrahedra inside the object.</p> <p>If false, Faces and/or Objects can be specified. Implies apply restrictions to triangles on the surface of the face or object.</p> <p>RestrictMaxElemLength</p> <p>If true, MaxElemLength should be specified.</p> <p>RestrictLayersNum</p> <p>If true, LayersNum should be specified.</p>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EditDensityControlOp(<OpName>, <DensityControlOpParams>)
<b>Python Example</b>	<pre>oModule.EditDensityControlOp("Clone Mesh Density1", [     "NAME:Clone Mesh Density1",     "RefineInside:=", True,     "RestrictMaxElemLength:=", True,     "MaxElemLength:=", "0.02mm",     "RestrictLayersNum:=", True,     "LayersNum:=", "15" ])</pre>

<b>VB Syntax</b>	EditDensityControlOp <OpName>, <DensityControlOpParams>
<b>VB Example</b>	<pre>oModule.EditDensityControlOp "Clone Mesh Density1", _     Array("NAME:Clone Mesh Density1", _         "RefineInside:=", true, _         "RestrictMaxElemLength:=", true, _         "MaxElemLength:=", "0.015mm", _         "RestrictLayersNum:=", true, _         "LayersNum:=", "15")</pre>

## EditEdgeCutLayerOp

Edits an existing edge cut operation. This cannot be used to modify assignments. Instead, the mesh operation should be deleted and a new one created.

<b>UI Access</b>	Double-click the operation in the project tree to modify its settings.		
<b>Parameters</b>	Name <OpName> <EdgeCutLayerOpParams>	Type String Array	Description Name of the operation to be edited. Structured array.  Array ("NAME:<OpName>", "Layer Thickness:=", <value>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EditEdgeCutLayerOp(<OpName>, <EdgeCutLayerOpParams>)
<b>Python Example</b>	<pre> oModule.EditEdgeCutLayerOp ("EdgeCut1", [     "NAME:EdgeCut1",     "Layer Thickness:=",      "0.66mm" ]) </pre>

<b>VB Syntax</b>	EditEdgeCutLayerOp <OpName>, <EdgeCutLayerOpParams>
<b>VB Example</b>	<pre> oModule.EditEdgeCutLayerOp "EdgeCut1", _ Array("NAME:EdgeCut1", _ "Layer Thickness:=", "0.66mm") </pre>

## EditLengthOp

Edits an existing length-based operation. This cannot be used to modify assignments. Instead, the mesh operation should be deleted and a new one created.

<b>UI Access</b>	Double-click the operation in the project tree to modify its settings.		
<b>Parameters</b>	Name <OpName>	Type String	Description Name of the operation to be edited.

	<LengthOpParams>	Array	Structured array.  Array ("NAME:<OpName>", "RefineInside:=", <bool>, "Objects:=", <AssignmentObjects>,
--	------------------	-------	--------------------------------------------------------------------------------------------------------------------

		<pre>"Faces:=", &lt;AssignmentFaces&gt;, "Edges:=", &lt;AssignmentEdges&gt;, "RestrictElem:=", &lt;bool&gt;, "NumMaxElem:=", &lt;integer&gt;, "RestrictLength:=", &lt;bool&gt;, "MaxLength:=", &lt;value&gt; "ApplyToInitialMesh:=", &lt;bool&gt;)  RefineInside      If true, Objects should be specified. Implies apply restrictions to tetrahedra inside the object.      If false, Edges, Faces and/or Objects can be specified. Implies apply restrictions to triangles on the surface of the face or object.  RestrictElem      If true, NumMaxElem should be specified.  RestrictLength      If true, MaxLength should be specified.</pre>
<b>Return Value</b>	None.	

<b>Python Syntax</b>	EditLengthOp(<OpName>, <LengthOpParams>)
<b>Python Example</b>	oModule>EditLengthOpK("Length1",

```
[  

  "NAME:Length1",  

  "RefineInside:=", True,  

  "Objects:=", ["Circle2", "Rectangle1"],  

  "RestrictElem:=", False,  

  "NumMaxElem:=", 1000,  

  "RestrictLength:=", True,  

  "MaxLength:=", "2mm",  

  "ApplyToInitialMesh:=", False  

])
```

<b>VB Syntax</b>	EditLengthOp <OpName>, <LengthOpParams>
<b>VB Example</b>	<pre>oModule.EditLengthOpK "Length1", Array("NAME:Length1", _    "RefineInside:=", false,    "Objects:=", Array("Box1"),    "RestrictElem:=", false,    "NumMaxElem:=", 1000,    "RestrictLength:=", true,    "MaxLength:=", "2mm"    "ApplyToInitialMesh:=", False)</pre>

## EditModelResolutionOp

Assigns a model resolution name, value and unit for mesh operations. If UseAutoLength is true, the Defeature length is not used.

<b>UI Access</b>	Double-click the operation in the project tree to modify its settings.		
<b>Parameters</b>	Name	Type	Description
	<OpName>	String	Name of the operation to be edited.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EditModelResolutionOp(<OpName>, <ModelResParams>)
<b>Python Example</b>	<pre>oModule.EditModelResolutionOp ("ModelResolution1", [     "NAME:ModelResolution1",     "UseAutoLength:=", False,     "DefeatureLength:=", "71.5891053163818mil" ])</pre>

<b>VB Syntax</b>	EditModelResolutionOp <OpName>, <ModelResParams>
<b>VB Example</b>	<pre>oModule.EditModelResolutionOp "ModelResolution1", _ Array ("NAME:ModelResolution1", "UseAutoLength:=", false, _ "DefeatureLength:=", "71.5891053163818mil")</pre>

## EditSkinDepthOp

Modifies an existing skin-depth based mesh operation. Assignments cannot be changed using this command. To change the assignment, you must delete operation and create it using a new assignment.

<b>UI Access</b>	Double-click the operation in the project tree to modify its settings.									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;OpName&gt;</td> <td>String</td> <td>Name of the operation to be edited.</td> </tr> <tr> <td>&lt;SkinDepthOpParams&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array ("NAME:&lt;OpName&gt;",       "RestrictElem:=", &lt;bool&gt;,       "NumMaxElem:=", &lt;int&gt;,       "SkinDepth:=", &lt;value&gt;,       "SurfTriMaxLength:=", &lt;value&gt;,       "NumLayers:=", &lt;int&gt;) RestrictElem</pre> <p>If true, NumMaxElem should be specified.</p> </td> </tr> </tbody> </table>	Name	Type	Description	<OpName>	String	Name of the operation to be edited.	<SkinDepthOpParams>	Array	<p>Structured array.</p> <pre>Array ("NAME:&lt;OpName&gt;",       "RestrictElem:=", &lt;bool&gt;,       "NumMaxElem:=", &lt;int&gt;,       "SkinDepth:=", &lt;value&gt;,       "SurfTriMaxLength:=", &lt;value&gt;,       "NumLayers:=", &lt;int&gt;) RestrictElem</pre> <p>If true, NumMaxElem should be specified.</p>
Name	Type	Description								
<OpName>	String	Name of the operation to be edited.								
<SkinDepthOpParams>	Array	<p>Structured array.</p> <pre>Array ("NAME:&lt;OpName&gt;",       "RestrictElem:=", &lt;bool&gt;,       "NumMaxElem:=", &lt;int&gt;,       "SkinDepth:=", &lt;value&gt;,       "SurfTriMaxLength:=", &lt;value&gt;,       "NumLayers:=", &lt;int&gt;) RestrictElem</pre> <p>If true, NumMaxElem should be specified.</p>								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	EditSkinDepthOp(<OpName>, <SkinDepthOpParams>)
<b>Python Example</b>	<pre>oModule.EditSkinDepthOp("SkinDepth1", [      "NAME:SkinD",     "RestrictElem:=", False,     "SkinDepth:=", "2mm",     "SurfTriMaxLength:=", "1mm",     "NumLayers:=", 2 ])</pre>

<b>VB Syntax</b>	EditSkinDepthOp <OpName>, <SkinDepthOpParams>
<b>VB Example</b>	<pre>oModule.EditSkinDepthOp "SkinDepth1", Array("NAME:SkinD",_       "RestrictElem:=", false, _       "SkinDepth:=", "2mm", _       "SurfTriMaxLength:=", "1mm", _       "NumLayers:=", 2)</pre>

## EditTrueSurfOp

Modifies an existing true surface approximation-based mesh operation. Assignments cannot be changed using this command. To change the assignment, delete this operation and create it using a new assignment.

<b>UI Access</b>	Double-click the operation in the project tree to modify its settings.									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;OpName&gt;</td> <td>String</td> <td>Name of the operation to be edited.</td> </tr> <tr> <td>&lt;TrueSurfOpParams&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array ("NAME:&lt;OpName&gt;",       "SurfDevChoice:=", &lt;RadioOption&gt;,       "SurfDev:=", &lt;value&gt;,       "NormalDevChoice:=", &lt;RadioOption&gt;,       "NormalDev:=", &lt;value&gt;,       "AspectRatioChoice:=", &lt;RadioOption&gt;,       "AspectRatio:=", &lt;double&gt;)  &lt;RadioOption&gt;   Type: &lt;int&gt;   0: Ignore   1: Use defaults   2: Specify the value</pre> </td> </tr> </tbody> </table>	Name	Type	Description	<OpName>	String	Name of the operation to be edited.	<TrueSurfOpParams>	Array	<p>Structured array.</p> <pre>Array ("NAME:&lt;OpName&gt;",       "SurfDevChoice:=", &lt;RadioOption&gt;,       "SurfDev:=", &lt;value&gt;,       "NormalDevChoice:=", &lt;RadioOption&gt;,       "NormalDev:=", &lt;value&gt;,       "AspectRatioChoice:=", &lt;RadioOption&gt;,       "AspectRatio:=", &lt;double&gt;)  &lt;RadioOption&gt;   Type: &lt;int&gt;   0: Ignore   1: Use defaults   2: Specify the value</pre>
Name	Type	Description								
<OpName>	String	Name of the operation to be edited.								
<TrueSurfOpParams>	Array	<p>Structured array.</p> <pre>Array ("NAME:&lt;OpName&gt;",       "SurfDevChoice:=", &lt;RadioOption&gt;,       "SurfDev:=", &lt;value&gt;,       "NormalDevChoice:=", &lt;RadioOption&gt;,       "NormalDev:=", &lt;value&gt;,       "AspectRatioChoice:=", &lt;RadioOption&gt;,       "AspectRatio:=", &lt;double&gt;)  &lt;RadioOption&gt;   Type: &lt;int&gt;   0: Ignore   1: Use defaults   2: Specify the value</pre>								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	EditTrueSurfOp(<OpName>, <TrueSurfOpParams>)
<b>Python Example</b>	<pre>oModule.EditTrueSurfOp ("TrueSurf2", [      "NAME:trusurf",     "SurfDevChoice:=", 2,     "SurfDev:=", "0.03mm",     "NormalDevChoice:=", 1,     "AspectRatioChoice:=", 2,     "AspectRatio:=", 10 ])</pre>

<b>VB Syntax</b>	EditTrueSurfOp <OpName>, <TrueSurfOpParams>
<b>VB Example</b>	<pre>oModule.EditTrueSurfOp "TrueSurf2", Array("NAME:trusurf", _       "SurfDevChoice:=", 2, _       "SurfDev:=", "0.03mm", _       "NormalDevChoice:=", 1, _       "AspectRatioChoice:=", 2, _</pre>

	"AspectRatio:=", 10)
--	----------------------

## InitialMeshSettings

Assigns a true surface-based mesh operation to the selection.

UI Access	Mesh > Initial Mesh Settings								
Parameters	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;<i>InitialMeshSettingsParams</i>&gt;</td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array ("NAME:&lt;MeshSettings&gt;",       "CurvedSurfaceApproxChoice:=", &lt;"UseSlider"       or "Manual Settings"&gt;       "SliderMeshSettings:=", &lt;integer 1 through       9&gt;,       "UseLegacyFaceterForTauVolumeMesh:=", &lt;boolean&gt;,       "DynamicSurfaceResolution:=", &lt;boolean&gt;,       "UseFlexMeshingForTAUvolumeMesh:=", &lt;boolean&gt;,       "UseAlternativeMeshMethodsAsFallBack:=", &lt;boolean&gt;,       "AllowPhiForLayeredGeometry:=", &lt;boolean&gt;)</pre> </td> </tr> </table>	Name	Type	Description	< <i>InitialMeshSettingsParams</i> >	Array	<p>Structured array.</p> <pre>Array ("NAME:&lt;MeshSettings&gt;",       "CurvedSurfaceApproxChoice:=", &lt;"UseSlider"       or "Manual Settings"&gt;       "SliderMeshSettings:=", &lt;integer 1 through       9&gt;,       "UseLegacyFaceterForTauVolumeMesh:=", &lt;boolean&gt;,       "DynamicSurfaceResolution:=", &lt;boolean&gt;,       "UseFlexMeshingForTAUvolumeMesh:=", &lt;boolean&gt;,       "UseAlternativeMeshMethodsAsFallBack:=", &lt;boolean&gt;,       "AllowPhiForLayeredGeometry:=", &lt;boolean&gt;)</pre>		
Name	Type	Description							
< <i>InitialMeshSettingsParams</i> >	Array	<p>Structured array.</p> <pre>Array ("NAME:&lt;MeshSettings&gt;",       "CurvedSurfaceApproxChoice:=", &lt;"UseSlider"       or "Manual Settings"&gt;       "SliderMeshSettings:=", &lt;integer 1 through       9&gt;,       "UseLegacyFaceterForTauVolumeMesh:=", &lt;boolean&gt;,       "DynamicSurfaceResolution:=", &lt;boolean&gt;,       "UseFlexMeshingForTAUvolumeMesh:=", &lt;boolean&gt;,       "UseAlternativeMeshMethodsAsFallBack:=", &lt;boolean&gt;,       "AllowPhiForLayeredGeometry:=", &lt;boolean&gt;)</pre>							
Return Value	None.								

Python Syntax	InitialMeshSettings(< <i>InitialMeshSettingsParams</i> >)
---------------	-----------------------------------------------------------

**Python Example**

```
oModule.InitialMeshSettings  
([  
    "NAME:MeshSettings",  
    [  
        "NAME:GlobalSurfApproximation",  
        "CurvedSurfaceApproxChoice:=", "UseSlider",  
        "SliderMeshSettings:=" , 5  
    ],  
    [  
        "NAME:GlobalCurvilinear",  
        "Apply:=", False  
    ],  
    [  
        "NAME:GlobalModelRes",  
        "UseAutoLength:=", True  
    ],  
    [  
        "NAME:GlobalLengthMeshSetup",  
        "RefineInside:=", True,  
        "ID:=", -1,  
    ]
```

```

    "Type:=", "LengthBased",
    "RestrictElem:=", False,
    "NumMaxElem:=", "1000",
    "RestrictLength:=", True,
    "MaxLength:=", "45mm",
    "ApplyToInitialMesh:=", True,
    "IsGlobal:=", True
),
"MeshMethod:=", "Auto",
"UseLegacyFaceterForTAUVolumeMesh:=", False,
"DynamicSurfaceResolution:=", False,
"UseFlexMeshingForTAUvolumeMesh:=", False,
"UseAlternativeMeshMethodsAsFallBack:=", False,
"AllowPhiForLayeredGeometry:=", True
])
)

```

<b>VB Syntax</b>	InitialMeshSettings < <i>InitialMeshSettingsParams</i> >
<b>VB Example</b>	<pre> oModule.InitialMeshSettings Array("NAME:MeshSettings", Array("NAME:GlobalSurfApproximation", "CurvedSurfaceApproxChoice:=", "UseSlider", "SliderMeshSettings:=", 5), </pre>

```
        Array("NAME:GlobalCurvilinear", "Apply:=", false),
        Array("NAME:GlobalModelRes", "UseAutoLength:=", true), "MeshMethod:=", "Auto",
        "UseLegacyFaceterForTauVolumeMesh:=", false,
        "DynamicSurfaceResolution:=", false,
        "UseFlexMeshingForTAUvolumeMesh:=", true,
        "UseAlternativeMeshMethodsAsFallBack:=", false,
        "AllowPhiForLayeredGeometry:=", true)

oModule.InitialMeshSettings Array("NAME:MeshSettings",
Array("NAME:GlobalSurfApproximation",
"CurvedSurfaceApproxChoice:",
    "ManualSettings",
    "SurfDevChoice:=", 2,
    "SurfDev:=", "0.01mm",
    "NormalDevChoice:=", 2,
    "NormalDev:=", "22.5deg",
    "AspectRatioChoice:=", 2, "AspectRatio:=", "10"),
Array("NAME:GlobalCurvilinear", "Apply:=", false),
Array("NAME:GlobalModelRes", "UseAutoLength:=", true),
Array("GlobalLengthMeshSetup", "RefineInside:=", True, "ID:=", -1,
"Type:=", "LengthBased", "RestrictElem:=", False, "NumMaxElem:=", "1000",
"RestrictLength:=", True, "MaxLength:=", "45mm",
```

```
"ApplyToInitialMesh:=", True, "IsGlobal:=", True)  
"MeshMethod:=", "Auto",  
"UseLegacyFaceterForTauVolumeMesh:=", false,  
"DynamicSurfaceResolution:=", false,  
"UseFlexMeshingForTAUvolumeMesh:=", true,  
"UseAlternativeMeshMethodsAsFallBack:=", false,  
"AllowPhiForLayeredGeometry:=", true)
```

This page intentionally  
left blank.

# 15 - Analysis Setup Module Script Commands

Maxwell analysis setup commands should be executed by the Analysis module, referred to in Maxwell scripts as the "AnalysisSetup" module.

```
Set oModule = oDesign.GetModule("AnalysisSetup")
```

[ClearLinkedData](#)

[CopySetup](#)

[DeleteSetups](#)

[EditSetup](#)

[ExportCircuit](#)

[ExportIcepak](#)

[ExportSolnData](#)

[GetSetupCount](#)

[GetSetups](#)

[GetSweepCount](#)

[GetSweeps](#)

[PasteSetup](#)

[PasteSweep](#)

[RenameSetup](#)

[ResetAllToTimeZero](#)

[ResetSetupToTimeZero](#)

[RevertAllToInitial](#)

[RevertAllToInitialTemperature](#)

[RevertSetupToInitial](#)

[RevertSetupToInitialTemperature](#)

## ClearLinkedData (Module)

Clear the linked data of all the solution setups. Similar to the ClearLinkedData command for the design level.

**Right-click menu of individual setups under the Analysis item.**

<b>UI Access</b>	<b>Project Manager &gt; Design name &gt; Analysis &gt; right-click Setup name &gt; Clear Linked Data</b>		
<b>Parameters</b>	Name <SetupNameArray>	Type Array	Description Specify the name of the setups whose linked data are to be cleaned
<b>Return Value</b>	None		

<b>Python Syntax</b>	ClearLinkedData(<SetupNameArray>)
<b>Python Example</b>	oModule.ClearLinkedData(["setup1"])

<b>VB Syntax</b>	ClearLinkedData <SetupNameArray>
<b>VB Example</b>	oModule.ClearLinkedData Array("setup1")

## CopySetup

Copy the specified Optimetrics setup.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SetupName&gt;</td> <td>String</td> <td>Name of the setup.</td> </tr> </tbody> </table>	Name	Type	Description	<SetupName>	String	Name of the setup.
Name	Type	Description					
<SetupName>	String	Name of the setup.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	CopySetup (<SetupName>)
<b>Python Example</b>	<code>oModule.CopySetup ("OptimizationSetup1")</code>

<b>VB Syntax</b>	CopySetup <SetupName>
<b>VB Example</b>	<code>oModule.CopySetup "OptimizationSetup1"</code>

## DeleteSetups

Deletes one or more solution setups, which are specified by an array of solution setup names.

<b>UI Access</b>	Right-click a solution setup in the project tree and then click <b>Delete</b> on the shortcut menu, or delete selected solution setups in the <b>List</b> dialog box.						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SetupArray&gt;</td> <td>Array</td> <td>Array of solution setup names.</td> </tr> </tbody> </table>	Name	Type	Description	<SetupArray>	Array	Array of solution setup names.
Name	Type	Description					
<SetupArray>	Array	Array of solution setup names.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	DeleteSetups (<SetupArray>)
----------------------	-----------------------------

**Python Example**

```
oModule.DeleteSetups ( ["Setup1", "Setup2"] )
```

**VB Syntax**

```
DeleteSetups <SetupArray>
```

**VB Example**

```
oModule.DeleteSetups Array("Setup1", "Setup2")
```

## EditSetup [Maxwell]

**Use:** Modifies an existing solution setup.

**Command:** Double-click a solution setup in the project tree to modify its settings.

**Syntax:** EditSetup <SetupName>, <AttributesArray>

**Return Value:** None

*Parameters:*

<SetupName>

Type: <string>

Name of the solution setup being edited.

<AttributesArray>

Array("NAME:<NewSetupName>", <NamedParameters>)

NamedParameters depend upon the solution type of the design. See [InsertSetup](#) for additional details and examples.

**Example:** 3D EddyCurrent EditSetup:

```
oModule.EditSetup "Setup1",
Array("NAME:Setup1", "Enabled:=", true,
Array("NAME:MeshLink", "ImportMesh:=", false),
```

```

"MaximumPasses:=", 10,
"MinimumPasses:=", 2,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
"SolveFieldOnly:=", false,
"PercentError:=", 1,
"SolveMatrixAtLast:=", true,
"UseNonLinearIterNum:=", false,
"UseCacheFor:=", Array("Pass"),
"UseIterativeSolver:=", false,
"RelativeResidual:=", 1E-05,
"NonLinearResidual:=", 0.0001,
"SmoothBHCurve:=", true,
"Frequency:=", "60Hz",

```

**Note:** For Eddy Current designs, Frequency also supports use of variables and expressions.

```

"HasSweepSetup:=", true,
Array("NAME:SweepRanges",
Array("NAME:Subrange",
"RangeType:=", "LinearStep",
"RangeStart:=", "10Hz"
"RangeEnd:=", "1000Hz",
"RangeStep:=", "10Hz"),
Array("NAME:Subrange",
"RangeType:=", "SinglePoints",
"RangeStart:=", "1000Hz",
"RangeEnd:=", "1000Hz",
"SaveSingleField:=", false)),
"SaveAllFields:=", true,
"UseHighOrderShapeFunc:=", false,
"UseMuLink:=", false)

```

**Example:** 3D ElectricTransient EditSetup:

```
oModule.EditSetup "Setup1",
Array("NAME:Setup1",
"Tolerance:=", 0.005,
Array("NAME:Data",
"SaveField:=", true,
"Stop:=", "100s",
"InitialStep:=", "0.01s",
"MaxStep:=", "0.5s"),
"Initial Voltage:=", "0V",
"NumberOfOutputVars:=", 0)
```

**Example:** 3D Transient EditSetup:

```
oModule.EditSetup "Setup1",
Array("NAME:Setup1", "Enabled:=", true,
Array("NAME:MeshLink", "ImportMesh:=", false),
"NonlinearSolverResidual:=", "0.005",
"ScalarPotential:=", "Second Order",
"SmoothBHCurve:=", false,
"StopTime:=", "10000000ns",
"TimeStep:=", "2000000ns",
"OutputError:=", false,
"OutputPerObjectCoreLoss:=", true,
"OutputPerObjectSolidLoss:=", false,
"UseControlProgram:=", false,
"ControlProgramName:=", " ",
"ControlProgramArg:=", " ",
"CallCtrlProgAfterLastStep:=", false,
"FastReachSteadyState:=", false,
"AutoDetectSteadyState:=", false,
"IsGeneralTransient:=", true,
"IsHalfPeriodicTransient:=", false,
"SaveFieldsType:=", "None",
"UseNonLinearIterNum:=", false,
```

```
"CacheSaveKind:=", "Count",
"NumberSolveSteps:=", 1,
"RangeStart:=", "0s",
"RangeEnd:=", "0.1s")
```

**Example:** 3D ACConduction EditSetup:

```
oModule.EditSetup "Setup1",
Array("NAME:Setup1", "Enabled:=", true,
Array("NAME:MeshLink", "ImportMesh:=", false),
"MaximumPasses:=", 10,
"MinimumPasses:=", 2,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
"SolveFieldOnly:=", false,
"PercentError:=", 1,
"SolveMatrixAtLast:=", true,
"UseNonLinearIterNum:=", true,
"MinIterNum:=", 1,
"MaxIterNum:=", 100,
"CacheSaveKind:=", "Delta",
"ConstantDelta:=", "0s",
"UseIterativeSolver:=", false,
"RelativeResidual:=", 1E-06,
"NonLinearResidual:=", 0.001,
"Frequency:=", "60Hz",
"HasSweepSetup:=", false)
```

**Example:** 2D DCConduction EditSetup:

```
oModule.EditSetup "Setup1",
Array("NAME:Setup1",
"MaximumPasses:=", 20,
"MinimumPasses:=", 2,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
```

```
"SolveFieldOnly:=", false,
"PercentError:=", 1,
"SolveMatrixAtLast:=", true,
"UseOutputVariable:=", false,
"PreAdaptMesh:=", false)
```

**Example:** 2D Transient EditSetup:

```
Set oModule = oDesign.GetModule("AnalysisSetup")
oModule.EditSetup "Setup1",
Array("NAME:Setup1",
"Enabled:=", true,
"NonlinearSolverResidual:=", "0.0001",
"TimeIntegrationMethod:=", "BackwardEuler",
"SmoothBHCurve:=", true,
"StopTime:=", "0.2s",
"TimeStep:=", "0.0002s",
"OutputError:=", false,
"OutputPerObjectCoreLoss:=", true,
"OutputPerObjectSolidLoss:=", false,
"UseControlProgram:=", false,
"ControlProgramName:=", " ",
"ControlProgramArg:=", " ",
"CallCtrlProgAfterLastStep:=", false,
```

```
"FastReachSteadyState:=", true,
"AutoDetectSteadyState:=", true,
"FrequencyOfAddedVoltageSource:=", "50Hz",
"StopCriterion:=", 0.005,
"IsGeneralTransient:=", false,
"IsHalfPeriodicTransient:=", true,
"HasSweepSetup:=", true,
"SaveFieldsType:=" , "Custom",
"Sweep Ranges:="
[
  "SweepSetupType:=" , "LinearStep",
  "StartValue:=" , "0",
  "StopValue:=" , "200ms",
  "StepSize:=" , "25ms"
]
```

**Example:** Transient EditSetup:

```
oModule.EditSetup("Setup1",
[
  "NAME:Setup1",
  "Enabled:=" , True,
  "NonlinearSolverResidual:=", "0.9",
  "ScalarPotential:=" , "Second Order",
  "TimeIntegrationMethod:=", 0,
  "SmoothBHCurve:=" , True,
  "StopTime:=" , "200ms",
```

```
"TimeStep:=" , "10ms",
"OutputError:=" , False,
"UseControlProgram:=" , False,
"ControlProgramName:=" , " ",
"ControlProgramArg:=" , " ",
"CallCtrlProgAfterLastStep:=" , False,
"FastReachSteadyState:=" , False,
"AutoDetectSteadyState:=" , False,
"IsGeneralTransient:=" , True,
"IsHalfPeriodicTransient:=" , False,
"HasSweepSetup:=" , True,
"SaveFieldsType:=" , "Every N Steps",
"N Steps:=" , "$MyStepCount"
])
```

## ExportCircuit

*Use:* Export equivalent circuit data.

*Command:* Right-click a setup in the project tree or the **Analysis** folder and choose **Export Circuit**.

*Syntax:* ExportCircuit <Solution>, <Variation>, <FileName>, <ExportSettings>, <modelName>, <Freq>

*Return Value:* none

*Parameters:* <Solution>

<SetupName>:<SolutionName>

<SetupName>

Type: <string>

Name of the setup where circuit is being exported

<SolutionName>

Type: <String>  
Name of the solution.

<Variation>  
Type: <string>  
The variation where circuit is being exported

<FileName>  
Type: <string>  
The name of the file where circuit is being exported

<ModelName>  
Type: <String>  
Model name or name of the sub circuit (Optional). If not specified then <FileName> is considered as model name.

<Freq>  
Type: <double>  
Sweep frequency in hz.

<ExportSettings>  
Array("NAME:CircuitData", "MatrixName:=", \_  
<ReduceMatrix>, "NumberOfCells:=", <NumCell>, "UserHasChangedSettings:=", <User-  
ChangedSettings>, "IncludeCap:=", <IncludeCap>, "IncludeCond:=", <IncludeCond>, Array  
("NAME:CouplingLimits", <CouplingLimitsArray>, "IncludeDCR:=", <IncludeDCR>, "IncludeDCL:=",  
<IncludeDCL>, "IncludeACR:=", <IncludeACR>, "IncludeACL:=", <IncludeACL>, "ADDResistance:=",  
<AddResistance>)

**Parameters:**

```
<ReduceMatrix>
  Type: <string>
    One of the reduced matrix setup or "Original"

<NumCell>
  Type: <string>
    Number of cells in export. Can be a variable.

<UserChangedSettings>
  Type: <bool>
    Whether user changed settings or use default settings.

<IncludeCap>
  Type: <bool>
    Flag indicates whether to export Capacitance matrix.

<IncludeCond>
  Type: <bool>
    Flag indicates whether to export Conductance matrix.

<IncludeDCR>
  Type: <bool>
    Flag indicates whether to export DC resistance matrix.

<IncludeDCL>
  Type: <bool>
```

```
Flag indicates whether to export DC Inductance matrix.  
<IncludeACR>  
Type: <bool>  
Flag indicates whether to export AC resistance matrix.  
<IncludeACL>  
Type: <bool>  
Flag indicates whether to export AC inductance matrix.  
<AddResistance>  
Type: <bool>  
Adds the DC and AC resistance.
```

**Note:**

You cannot export both AC and DC matrices unless **AddResistance** is selected.

```
<CouplingLimitsArray>  
Array("NAME:CouplingLimits", "CouplingLimitType:=", <CouplingLimitType>, _  
<CouplingLimitsParameters>, 0.01, "CondFraction:=", 0.01)  
<CouplingLimitType> = "None"  
  
Argument not needed  
<CouplingLimitType> = "ByFraction"  
<CouplingLimitsParameters>  
"CapFraction:=", <Fraction>, "IndFraction:=", <Fraction>,
```

"ResFraction:=", <Fraction>, "CondFraction:=", <Fraction>,

**Parameters:**

<Fraction>

Type: <double>

Fraction of the self term

```
<CouplingLimitType> = "ByValue"  
  
<CouplingLimitsParameters>  
  
"CapLimit:=", <Limit>, "IndLimit:=", <Limit>, "ResLimit:=", <Limit>,  
"CondLimit:=", <Limit>,
```

**Parameters:**

<Limit>

Type: <string>

Value of the limit.

*VB Example:*

```
oModule.ExportCircuit  
"Setup1 : LastAdaptive", "", "C:/Project/Q3D/FourNets.cir", Array("NAME:CircuitData", _  
"MatrixName:=", "Original", "NumberOfCells:=", "1", "UserHasChangedSettings:=", true, _  
"IncludeCap:=", true, "IncludeCond:=", true, Array("NAME:CouplingLimits", "CouplingLimitType:=",  
"By Fraction", "CapFraction:=", 0.01, "IndFraction:=", 0.01, "ResFraction:=", 0.01, _  
"CondFraction:=", 0.01), "IncludeDCR:=", false, "IncludeDCL:=", false, "IncludeACR:=", false, _  
"IncludeACL:=", false, "ADDResistance:=", true), "", 200000000000
```

## ExportIcepak

**Use:** Exports Icepak data to the specified output directory.

**Command:** Right-click a setup under **Analysis** in the project tree, and select **Create Icepak Design**. From the dialog box, select an appropriate solution setup and variation, enter the path to the output directory, and click the **OK** button.

**Syntax:** ExportIcepak <SetupName>, <Variation>, <Output Directory>

**Return Value:** None

**Parameters:** <SetupName>

Type: string

Setup name.

<Variation>

Type: string

Variation values.

<Output Directory>

Type: string

Path to the output directory in which the Icepak design files will be saved.

**Example:** oModule.ExportIcepak "Setup1", "air\_gap='1mm' diameter='120mm", "D:\Work\Icepak\"

## ExportSolnData

**Use:** Exports solution data to a file.

**Command:** Right-click a parameter setup under **Parameters** in the project tree, and select **ViewSolution**. From the dialog box, select an appropriate solution type, and click the **Export Solution** button.

**Syntax:** ExportSolnData <SetupName>, <SolutionName>, <IsPostProcessed>, <Variation>, <ExportFileName>,

**Return Value:** None

**Parameters:** <SetupName>

Type: string

Setup name.

<SolutionName>

Type: string

Solution setup name.

<IsPostProcessed>

Type: bool

Is post processed (true/false).

<Variation>

Type: string

Variation values.

<ExportFileName>

Type: string

Export file name.

**Example:** oModule.ExportSolnData "Setup1", "Matrix1", false, "", \_

"C:/Maxwell/Projects/Export/OutputSolution.txt"

## GetSetupCount

Gets the number of analysis setups in a design.

---

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Integer containing number of setups.

<b>Python Syntax</b>	<code>GetSetupCount ()</code>
<b>Python Example</b>	<code>oModule.GetSetupCount ()</code>

<b>VB Syntax</b>	<code>GetSetupCount</code>
<b>VB Example</b>	<code>oModule.GetSetupCount</code>

## GetSetups

Gets the names of analysis setups in a design.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array of analysis setup names

<b>Python Syntax</b>	<code>GetSetups ()</code>
<b>Python Example</b>	<code>oModule.GetSetups ()</code>

<b>VB Syntax</b>	GetSetups
<b>VB Example</b>	<code>oModule.GetSetups</code>

## GetSweepCount

Gets the number of sweeps in a specified analysis setup.

<b>UI Access</b>	N/A						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td><code>&lt;SetupName&gt;</code></td><td>String</td><td>Name of specified setup.</td></tr></table>	Name	Type	Description	<code>&lt;SetupName&gt;</code>	String	Name of specified setup.
Name	Type	Description					
<code>&lt;SetupName&gt;</code>	String	Name of specified setup.					
<b>Return Value</b>	Integer containing number of sweeps for the named setup.						

<b>Python Syntax</b>	<code>GetSweepCount(&lt;SetupName&gt;)</code>
<b>Python Example</b>	<code>oModule.GetSweepCount("Setup1")</code>

<b>VB Syntax</b>	<code>GetSweepCount &lt;SetupName&gt;</code>
<b>VB Example</b>	<code>oModule.GetSweepCount "Setup1"</code>

## GetSweeps

Gets the names of all sweeps in a given analysis setup.

<b>UI Access</b>	N/A
------------------	-----

<b>Parameters</b>	Name <code>&lt;SetupName&gt;</code>	Type String	Description Name of specified setup.
<b>Return Value</b>	Array of sweep names.		

<b>Python Syntax</b>	<code>GetSweeps (&lt;SetupName&gt;)</code>
<b>Python Example</b>	<code>oModule.GetSweeps ("Setup1")</code>

<b>VB Syntax</b>	<code>GetSweeps &lt;SetupName&gt;</code>
<b>VB Example</b>	<code>oModule.GetSweeps "Setup1"</code>

## InsertSetup [Maxwell]

**Use:** Adds a new solution setup.

**Command:** Maxwell3D or Maxwell2D>Analysis Setup>Add Solution Setup

**Syntax:** InsertSetup <SetupType>, <AttributesArray>

**Return Value:** None

*Parameters:*

<SetupType>

Type: <string>

Possible values for 3D designs are: "[Magnetostatic](#)", "[EddyCurrent](#)", "[Transient](#)", "[Electrostatic](#)", "[DCConduction](#)", "[ElectroDCConduction](#)", "[ElectricTransient](#)", and "[ACConduction](#)".

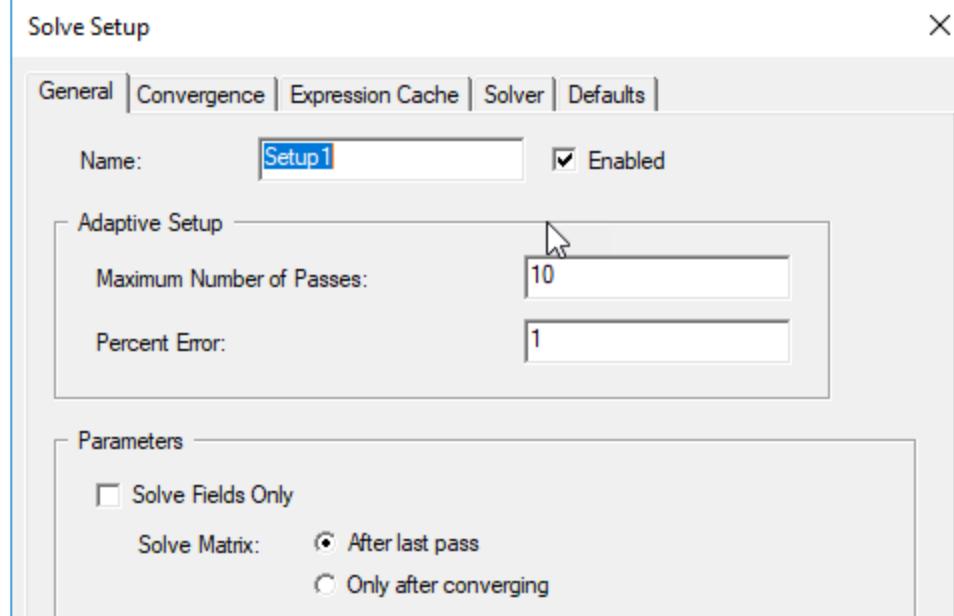
Possible values for 2D designs are: "[Magnetostatic](#)", "[EddyCurrent](#)", "[Transient](#)", "[Electrostatic](#)", "[ACConduction](#)", "[DCConduction](#)".

```
<AttributesArray>
```

```
    Array("NAME:<SetupName>", <NamedParameters>)
```

```
<NamedParameters>
```

The named parameters vary according to the setup type and settings selected for a setup. Please see the typical examples below for each solution type. The parameters correspond to fields in the Setup dialog box. For example, here is a default dialog for a Maxwell 3D Maganetostatic design.



Some of the attributes Array correspond to these fields on the General tab of the Setup.

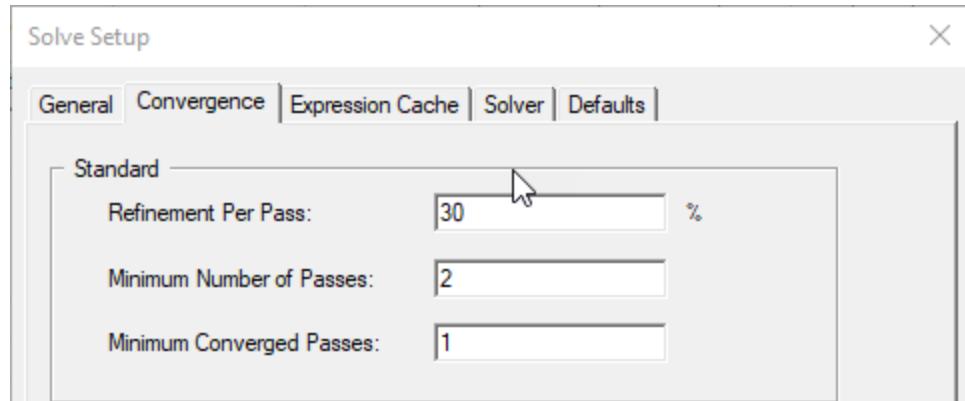
```
oModule.InsertSetup "Magnetostatic", Array("NAME:Setup1",
```

```
"MaximumPasses:=", 10,
```

```
...
```

```
"SolveMatrixAtLast:=", true,
```

Other parameters correspond to fields in the Convergence tab of the Setup.



```
"MinimumPasses:=", 2,  
"MinimumConvergedPasses:=", 1,  
"PercentRefinement:=", 30,
```

Still other parameters correspond to fields in the Solver tab of the Setup.

```
"NonLinearResidual:=", 0.001,  
UseNonLinearIterNum:=", True,  
"MinIterNum:=", 5,  
"MaxIterNum:=", 10,
```

Each different example below corresponds to the fields for each different setup.

*Example:*

3D Magnetostatic solve setup:

```
oModule.InsertSetup "Magnetostatic", Array("NAME:Setup1",  
"MaximumPasses:=", 10,  
"MinimumPasses:=", 2,
```

```
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
"SolveFieldOnly:=", false,
"PercentError:=", 1,
"SolveMatrixAtLast:=", true,
"UseOutputVariable:=", false,
"PreAdaptMesh:=", false,
"UseNonLinearIterNum:=", True,
"MinIterNum:=", 5,
"MaxIterNum:=", 10,
"NonLinearResidual:=", 0.001,
"MuNonLinearBH:=", true,
"ComputeHc:=", false,
"HcNonLinearBH:=", true,
"UserOverrideMu:=", true,
"UserExePath:=", "E:\views\projects\ucp.exe",
"UserExeArg:=", "-3D")
```

*Example:*

3D EddyCurrent solve setup:

```
oModule.InsertSetup "EddyCurrent", Array("NAME:Setup1",
"Enabled:=", true,
"MaximumPasses:=", 10,
"MinimumPasses:=", 2,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
"SolveFieldOnly:=", false,
"PercentError:=", 1,
"SolveMatrixAtLast:=", true,
"UseNonLinearIterNum:=", True,
"MinIterNum:=", 5,
"MaxIterNum:=", 10,
"UseIterativeSolver:=", true,
```

```
"RelativeResidual:=", 0.00013,  
"ComputeForceDensity:=", false,  
"ComputePowerLoss:=", false,  
"ThermalFeedback:=", true,  
"Frequency:=", "60Hz",
```

**Note:** For Eddy Current designs, Frequency also supports use of variables and expressions.

```
"HasSweepSetup:=", true,  
Array("NAME:SweepRanges", Array("NAME:Subrange", "RangeType:=", "LinearStep", "RangeStart:=",  
"10Hz", "RangeEnd:=", "1000Hz", "RangeStep:=", "10Hz"),  
Array("NAME:Subrange", "RangeType:=", "SinglePoints", "RangeStart:=", "1000Hz", "RangeEnd:=",  
"1000Hz", "SaveSingleField:=", false)), "SaveAllFields:=", true,  
"UseHighOrderShapeFunc:=", false)
```

*Example:*

3D Transient solve setup:

```
oModule.InsertSetup "Transient",  
Array("NAME:Setup1", "Enabled:=", true,  
"NonlinearSolverResidual:=", 0.005,  
"ScalarPotential:=", "First Order",  
"TimeIntegrationMethod:=", -1,  
Array("NAME:PrevSoln", "Project:=", "This Project*",  
"Product:=", "Maxwell",  
"Design:=", "NL_V_ED1_Link",  
"Soln:=", "Setup1 : LastAdaptive",  
Array("NAME:Params"), "ForceSourceToSolve:=", false,  
"PreservePartnerSoln:=", false,  
"PathRelativeTo:=", "TargetProject"),  
"StopTime:=", "10000000ns",
```

```
"TimeStep:=", "2000000ns",
"OutputError:=", true,
"OutputPerObjectCoreLoss:=", true,
"OutputPerObjectSolidLoss:=", false,
"UseControlProgram:=", false,
"ControlProgramName:=", "",
"ControlProgramArg:=", " ",
"CallCtrlProgAfterLastStep:=", false,
"FastReachSteadyState:=", true,
"FrequencyOfAddedVoltageSource:=", "60Hz",
"AutoDetectSteadyState:=", false,
"StopCriterion:=", "0.005",
"IsGeneralTransient:=", true,
"IsHalfPeriodicTransient:=", false,
"SaveFieldsType:=", "Every N Steps", "N Steps:=", "10", "Steps From:=", "0s", "Steps To:=",
"10000000ns",
"UseNonLinearIterNum:=" , True,
"MinIterNum:=" , 5,
"MaxIterNum:=" , 10,
"CacheSaveKind:=", "Count", "NumberSolveSteps:=", 1, "RangeStart:=", "0s", "RangeEnd:=",
"0.1s")
```

*Example:*

3D Electrostatic solve setup:

```
oModule.InsertSetup "Electrostatic", Array("NAME:Setup1",
"MaximumPasses:=", 10,
"MinimumPasses:=", 2,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
"SolveFieldOnly:=", false,
"PercentError:=", 1,
"SolveMatrixAtLast:=", true,
"UseOutputVariable:=", false,
"PreAdaptMesh:=", false)
```

*Example:*

3D DCConduction solve setup:

```
oModule.InsertSetup "DCConduction", Array("NAME:Setup2",
"MaximumPasses:=", 10,
"MinimumPasses:=", 2,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
"SolveFieldOnly:=", false,
"PercentError:=", 1,
"SolveMatrixAtLast:=", true,
"UseOutputVariable:=", false,
"PreAdaptMesh:=", false)
```

*Example:*

3D ElectroDCConduction solve setup:

```
oModule.InsertSetup "ElectroDCConduction", Array("NAME:Setup3",
"MaximumPasses:=", 10,
"MinimumPasses:=", 2,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
"SolveFieldOnly:=", false,
"PercentError:=", 1,
"SolveMatrixAtLast:=", true,
"UseOutputVariable:=", false,
"PreAdaptMesh:=", false)
```

*Example:*

3D ElectricTransient solve setup:

```
oModule.InsertSetup "ElectricTransient", Array("NAME:Setup1",
"_tolerance:=", 0.005,
Array("NAME:Data",
"SaveField:=", true,
"Stop:=", "100s",
"InitialStep:=", "0.01s",
"MaxStep:=", "5s"),
"Initial Voltage:=", "0V",
"NumberOfOutputVars:=", 0)
```

*VB Example:*

3D ACCondution solve setup:

```
oModule.InsertSetup "ACConduction", Array("NAME:Setup2", "Enabled:=", true,
Array("NAME:MeshLink", "ImportMesh:=", false),
"MaximumPasses:=", 10,
"MinimumPasses:=", 2,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
"SolveFieldOnly:=", false,
"PercentError:=", 1,
"SolveMatrixAtLast:=", true,
"UseNonLinearIterNum:=", true,
"MinIterNum:=", 1,
"MaxIterNum:=", 100,
"UseIterativeSolver:=", false,
"RelativeResidual:=", 1E-06,
"NonLinearResidual:=", 0.001,
"Frequency:=", "60Hz",
"HasSweepSetup:=", false)
```

*Python Example:*

```
3D ACConduction solve setup:  
oModule.InsertSetup("ACConduction",  
[  
    "NAME:Setup2",  
    "Enabled:=", True,  
    [  
        "NAME:MeshLink",  
        "ImportMesh:=", False  
    ],  
    "MaximumPasses:=", 10,  
    "MinimumPasses:=", 2,  
    "MinimumConvergedPasses:=", 1,  
    "PercentRefinement:=", 30,  
    "SolveFieldOnly:=", False,  
    "PercentError:=", 1,  
    "SolveMatrixAtLast:=", True,  
    "UseNonLinearIterNum:=", True,  
    "MinIterNum:=", 1,  
    "MaxIterNum:=", 100,  
    "UseIterativeSolver:=", False,
```

```
"RelativeResidual:=" , 1E-06,  
"NonLinearResidual:=" , 0.001,  
"Frequency:=" , "60Hz",  
"HasSweepSetup:=" , False  
])
```

*Example:*

2D Magnetostatic solve setup:

```
oModule.InsertSetup "Magnetostatic", Array("NAME:Setup1",  
"MaximumPasses:=", 10,  
"MinimumPasses:=", 2,  
"MinimumConvergedPasses:=", 1,  
"PercentRefinement:=", 30,  
"SolveFieldOnly:=", false,  
"PercentError:=", 1,  
"SolveMatrixAtLast:=", true,  
"UseNonLinearIterNum:=" , True,  
"MinIterNum:=" , 5,  
"MaxIterNum:=" , 10,  
"UseOutputVariable:=", false,  
"PreAdaptMesh:=", false,  
"NonLinearResidual:=", 0.0001,  
"MuNonLinearBH:=", true,  
"ComputeHc:=", false,  
"HcNonLinearBH:=", true  
"UserOverrideMu:=", true,  
"UserExePath:=" , "E:\views\projects\ucp.exe",  
"UserExeArg:=" , "-3D")
```

*Example:*

2D EddyCurrent solve setup:

```
oModule.InsertSetup "EddyCurrent", Array("NAME:Setup1",
"MaximumPasses:=", 10,
"MinimumPasses:=", 2,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
"SolveFieldOnly:=", false,
"PercentError:=", 1,
"SolveMatrixAtLast:=", true,
"UseNonLinearIterNum:=" , True,
"MinIterNum:=" , 5,
"MaxIterNum:=" , 10,
"UseOutputVariable:=", false,
"PreAdaptMesh:=", false,
"Frequency:=", "60Hz",
```

**Note:** For Eddy Current designs, Frequency also supports use of variables and expressions.

```
"NonLinearResidual:=", 0.0001,
"HasSweepSetup:=", false)
```

**Example:** 2D Transient solve setup:

```
oModule.InsertSetup "Transient",
Array("NAME:Setup1",
"Enabled:=", true,
"NonlinearSolverResidual:=", "0.0001",
"TimeIntegrationMethod:=", "RungeKutta",
"StopTime:=", "0.02s",
"TimeStep:=", "0.0005s",
"OutputError:=", false,
"OutputPerObjectCoreLoss:=", true,
```

```
"OutputPerObjectSolidLoss:=", false,
"UseControlProgram:=", false,
"ControlProgramName:=", "",
"ControlProgramArg:=", " ",
"CallCtrlProgAfterLastStep:=", false,
"FastReachSteadyState:=", false,
"FrequencyOfAddedVoltageSource:=", "60Hz",
"AutoDetectSteadyState:=", false,
"StopCriterion:=", "0.005",
"IsGeneralTransient:=", true,
"IsAutoSubDivision:=", true,
"HasSweepSetup:=", true,
"SweepSetupType:=", "LinearStep",
"StartValue:=", "0s",
"StopValue:=", "0.1s",
"StepSize:=", "0.005s",
"UseNonLinearIterNum:=" , True,
"MinIterNum:=" , 5,
"MaxIterNum:=" , 10,
"UseAdaptiveTimeStep:=", false,
"InitialTimeStep:=", "0.002s",
"MinTimeStep:=", "0.001s",
"MaxTimeStep:=", "0.003s",
"TimeStepErrTolerance:=", 0.0001)
```

**Example:** 2D Electrostatic solve setup:

```
oModule.InsertSetup "Electrostatic", Array("NAME:Setup1",
"MaximumPasses:=", 10,
"MinimumPasses:=", 2,
"MinimumConvergedPasses:=", 1,
"PercentRefinement:=", 30,
"SolveFieldOnly:=", false,
"PercentError:=", 1,
"SolveMatrixAtLast:=", true,
```

```
"UseOutputVariable:=", false,  
"PreAdaptMesh:=", false)
```

**Example:** 2D ACConduction solve setup:

```
oModule.InsertSetup "ACConduction", Array("NAME:Setup1",  
"MaximumPasses:=", 10,  
"MinimumPasses:=", 2,  
"MinimumConvergedPasses:=", 1,  
"PercentRefinement:=", 30,  
"SolveFieldOnly:=", false,  
"PercentError:=", 1,  
"SolveMatrixAtLast:=", true,  
"UseOutputVariable:=", false,  
"PreAdaptMesh:=", false,  
"Frequency:=", "60Hz",  
"HasSweepSetup:=", false)
```

**Example:** 2D DCConduction solve setup:

```
oModule.InsertSetup "DCConduction", Array("NAME:Setup1",  
"MaximumPasses:=", 10,  
"MinimumPasses:=", 2,  
"MinimumConvergedPasses:=", 1,  
"PercentRefinement:=", 30,  
"SolveFieldOnly:=", false,  
"PercentError:=", 1,  
"SolveMatrixAtLast:=", true,  
"UseOutputVariable:=", false,  
"PreAdaptMesh:=", false)
```

## PasteSetup

*Use:* Paste a solve setup.

*Syntax:* PasteSetup

*Return Value:* None

*VB Example:* oModule.PasteSetup

## PasteSweep

Pastes a copied sweep.

<b>UI Access</b>	Right-click on a setup, select <b>Paste</b>						
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;SetupName&gt;</td> <td>String</td> <td>Name of setup where the copied sweep is pasted to.</td> </tr> </table>	Name	Type	Description	<SetupName>	String	Name of setup where the copied sweep is pasted to.
Name	Type	Description					
<SetupName>	String	Name of setup where the copied sweep is pasted to.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	PasteSweep (<SetupName>)
<b>Python Example</b>	oModule.PasteSweep ("Setup1")

<b>VB Syntax</b>	PasteSweep <SetupName>
<b>VB Example</b>	oModule.PasteSweep "Setup1"

## RenameSetup

Renames an existing solution setup.

<b>UI Access</b>	Right-click a solution setup in the Project Manager and then click <b>Rename</b> on the shortcut menu.		
<b>Parameters</b>	Name	Type	Description
	<OldSetupName>	String	Name of the solution setup being renamed.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	RenameSetup (<OldSetupName>, <NewSetupName>)
<b>Python Example</b>	oModule.RenameSetup ("Setup1", "MySetup")

<b>VB Syntax</b>	RenameSetup <OldSetupName>, <NewSetupName>
<b>VB Example</b>	oModule.RenameSetup "Setup1", "MySetup"

## ResetAllToTimeZero

Forces the next solve to start from time 0 for all setups. Applies only to the Transient solution type.

**Command:** Right-click **Analysis** in the project tree, and select **Revert to Time Zero**.

**Syntax:**ResetAllToTimeZero

**Return Value:** None

Parameters: None

**Example:** oModule.ResetAllToTimeZero

## ResetSetupToTimeZero

**Use:** Forces the next solve to start from time 0 for a given setup. Applies only to the Transient solution type.

**Command:** Right-click a setup in the project tree, and select **Revert to Time Zero**.

**Syntax:** ResetSetupToTimeZero <SetupName>

**Return Value:** None

**Parameters:** <SetupName>

Type: string

Setup name

**Example:** oModule.ResetSetupToTimeZero "Setup1"

## RevertAllToInitial

Marks the current mesh for all solution setups as invalid. This will force the next simulation to begin with the initial mesh.

<b>UI Access</b>	Maxwell > Analysis Setup > Revert to Initial Mesh.
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	RevertAllToInitial ()
<b>Python Example</b>	oModule.RevertAllToInitial ()

<b>VB Syntax</b>	RevertAllToInitial
<b>VB Example</b>	<code>oModule.RevertAllToInitial</code>

## RevertAllToInitialTemperature

Reverts all setups to their initial temperature. Used when coupling with thermal designs.

<b>UI Access</b>	<b>Maxwell &gt; Set Object Temperature &gt; Enable Feedback.</b> Then in the Project Manager, right-click <b>Analysis &gt; Revert to Initial Temperature</b> . Repeat for all setups.
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	RevertAllToInitialTemperature()
<b>Python Example</b>	<code>oModule.RevertAllToInitialTemperature()</code>

<b>VB Syntax</b>	RevertAllToInitialTemperature
<b>VB Example</b>	<code>oModule.RevertAllToInitialTemperature</code>

## RevertSetupToInitial

Marks the current mesh for a solution setup as invalid. This will force the next simulation to begin with the initial mesh.

<b>UI Access</b>	Right-click a solution setup in the Project Manager and then click <b>Rename</b> on the shortcut menu.		
<b>Parameters</b>	Name <code>&lt;SetupName&gt;</code>	Type String	Description Name of specified setup.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>RevertSetupToInitial (&lt;SetupName&gt;)</code>
<b>Python Example</b>	<code>oModule.RevertSetupToInitial ("Setup1")</code>

<b>VB Syntax</b>	<code>RevertSetupToInitial &lt;SetupName&gt;</code>
<b>VB Example</b>	<code>oModule.RevertSetupToInitial "Setup1"</code>

## RevertSetupToInitialTemperature

Reverts a specified setup to its initial temperature. Used when coupling with thermal designs.

<b>UI Access</b>	<b>Maxwell &gt; Set Object Temperature &gt; Enable Feedback.</b> Then in the Project Manager, right-click <b>Analysis</b> > <b>Revert to Initial Temperature</b> .		
<b>Parameters</b>	Name <code>&lt;setupName&gt;</code>	Type String	Description Name of solution setup.
<b>Return Value</b>	None		

<b>Python Syntax</b>	<code>RevertSetupToInitialTemperature(&lt;setupName&gt;)</code>
----------------------	-----------------------------------------------------------------

**Python Example**

```
oModule.RevertSetupToInitialTemperature('Setup1')
```

**VB Syntax**

```
RevertSetupToInitialTemperature <setupName>
```

**VB Example**

```
oModule.RevertSetupToInitialTemperature "Setup1"
```

# 16 - Optimetrics Module Script Commands

Optimetrics script commands should be executed by the "Optimetrics" module.

```
Set oModule = oDesign.GetModule("Optimetrics")
oModule.CommandName <args>
```

## Conventions Used in this Chapter

<VarName>

Type: <string>

Name of a variable.

<VarValue>

Type: <string>

Value with unit (i.e., <value>, but cannot be an expression).

<StartV>

Type: <VarValue>

The starting value of a variable.

<StopV>

Type: <VarValue>

The stopping value of a variable.

<MinV>

Type: <VarValue>

The minimum value of a variable.

<MaxV>

Type: <VarValue>

The maximum value of a variable.

<IncludeVar>

Type: <bool>

Specifies whether the variable is included in the analysis.

<StartingPoint>

```
Array("NAME:StartingPoint", "<VarName>:=",
      <VarValue>, .... "<VarName>:=", <VarValue>)
```

<SaveField>

Type: <bool>

Specifies whether HFSS will remove the non-nominal field solution.

<MaxIter>

Type: <int>

Maximum iteration allowed in an analysis.

<PriorSetup>

Type: <string>

The name of the embedded parametric setup.

<Precede>

Type: <bool>

If true, the embedded parametric setup will be solved before the analysis begins.

If false, the embedded parametric setup will be solved during each iteration of the analysis.

<Constraint>

```
Array("NAME:LCS",
  "lc:=", Array("<VarName>:=",
<Coeff>, ...<VarName>:=", <Coeff>, "rel:=", <Cond>, "rhs:=", <Rhs>), ...
  "lc:=", Array("<VarName>:=", <Coeff>, ..."
```

```
<VarName>:=", <Coeff>, "rel:=", <Cond>, "rhs:=",
<Rhs>))
```

<Coeff>

Type: <double>

Coefficient for a variable in the linear constraint.

<Cond>

Type: <string>

Inequality condition.

<Rhs>

Type: <double>

Inequality value.

```
<OptiGoalSpec>

    "Solution:=", <Soln>, "Calculation:=", <Calc>,
    "Context:=", <Geometry>

    Array("NAME:Ranges",
        "Range:", Array("Var:=",
            <VarName>, "Type:=", <RangeType>, "Start:=",
            <StartV>, "Stop:=", <StopV>), ...
        "Range:", Array("Var:=", <VarName>, "Type:=",
            <RangeType>, "Start:=", <StartV>, "Stop:=",
            <StopV>))
    <Soln>

        Type: <string>
        Name of the solution.

<Calc>

        Type: <string>
        An expression that is composed of a basic solution quantity and an
        output variable.

<ContextName>

        Type: <string>
        Name of context needed in the evaluation of <Calc>.
```

<Geometry>

Type: <string>

Name of geometry needed in the evaluation of <Calc>.

<RangeType>

Type: <string>

if "r", start and stop values specify a range for the variable.

if "s", start values specify the single value for the variable.

[EditSetup](#)

[EditSetup \[Optimization\]](#)

[EditSetup \[Sensitivity\]](#)

[EditSetup \[Statistical\]](#)

[GetPropNames \[Optimetrics\]](#)

[GetPropValue \[Optimetrics\]](#)

[GetSetupNames \[Optimetrics\]](#)

[GetSetupNamesByType \[Optimetrics\]](#)

[InsertSetup \[Parametric\]](#)

[InsertSetup \[Optimization\]](#)

[InsertSetup \[Sensitivity\]](#)

[InsertSetup \[Statistical\]](#)

[PasteSetup \[Optimetrics\]](#)

[RenameSetup \[Optimetrics\]](#)

[SetPropValue \[Optimetrics\]](#)

[SolveSetup \[Optimetrics\]](#)

**The topics for this section include:**

[General Commands Recognized by the Optimetrics Module](#)

[Parametric Script Commands](#)

[Optimization Script Commands](#)

[Sensitivity Script Commands](#)

[Statistical Script Commands](#)

## General Commands Recognized by the Optimetrics Module

Following are general script commands recognized by the **Optimetrics** module:

[CopySetup](#)

[DistributedAnalyzeSetup](#)

[EditSetup](#)

[ExportDXConfigFile](#)

[ExportOptimetricsProfile](#)

[ExportOptimetricsResult](#)

[ExportParametricResults](#)

[GetOptimetricResult](#)

[GetPropNames \[Optimetrics\]](#)

[GetPropValue \[Optimetrics\]](#)

[GetSetupNames \[Optimetrics\]](#)

[GetSetupNamesByType \[Optimetrics\]](#)

[ImportSetup](#)

[PasteSetup \[Optimetrics\]](#)

[RenameSetup \[Optimetrics\]](#)

[SetPropValue \[Optimetrics\]](#)

[SolveSetup \[Optimetrics\]](#)

[SolveAllSetup](#)

## **CopySetup**

Copy the specified Optimetrics setup.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;SetupName&gt;</td><td>String</td><td>Name of the setup.</td></tr></table>	Name	Type	Description	<SetupName>	String	Name of the setup.
Name	Type	Description					
<SetupName>	String	Name of the setup.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	CopySetup (<SetupName>)
<b>Python Example</b>	<pre>oModule.CopySetup ("OptimizationSetup1")</pre>

<b>VB Syntax</b>	CopySetup <SetupName>
------------------	-----------------------

<b>VB Example</b>	<code>oModule.CopySetup "OptimizationSetup1"</code>
-------------------	-----------------------------------------------------

## DeleteSetups [Optimetrics]

Deletes the specified Optimetrics setups.

<b>UI Access</b>	Right-click the setup in the project tree, and then click <b>Delete</b> on the shortcut menu						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;NameArray&gt;</td><td>Array of Strings</td><td>An Array of Setup Names</td></tr></table>	Name	Type	Description	<NameArray>	Array of Strings	An Array of Setup Names
Name	Type	Description					
<NameArray>	Array of Strings	An Array of Setup Names					
<b>Return Value</b>	None						

<b>Python Syntax</b>	<code>DeleteSetups (&lt;NameArray&gt;)</code>
<b>Python Example</b>	<code>oModule.DeleteSetups ( ["OptimizationSetup1"] )</code>

<b>VB Syntax</b>	<code>DeleteSetups &lt;NameArray&gt;</code>
<b>VB Example</b>	<code>oModule.DeleteSetups Array("OptimizationSetup1")</code>

## DistributedAnalyzeSetup

Distributes all variable value instances within a parametric sweep to different machines already specified from within the user interface

<b>UI Access</b>	Right-click the parametric setup name in the project tree and select Distribute Analysis.
------------------	-------------------------------------------------------------------------------------------

<b>Parameters</b>	Name <code>&lt;ParametricSetupName&gt;</code>	Type String	Description Name of the Setup
<b>Return Value</b>	None		

<b>Python Syntax</b>	<code>DistributedAnalyzeSetup (&lt;ParametricSetupName&gt;)</code>
<b>Python Example</b>	<code>oModule.DistributedAnalyzeSetup ("ParametricSetup1")</code>

<b>VB Syntax</b>	<code>DistributedAnalyzeSetup &lt;ParametricSetupName&gt;</code>
<b>VB Example</b>	<code>oModule.DistributedAnalyzeSetup "ParametricSetup1"</code>

## ExportDXConfigFile

Create an xml file with the setup information for Design Xplorer

<b>UI Access</b>	Right click on the Design Xplorer setup in the project tree and choose <b>Export External Connector Addin Configuration...</b>						
<b>Parameters</b>	<table border="1"> <tr> <td>Name <code>&lt;SetupName&gt;</code></td> <td>Type String</td> <td>Description Must be one of existing DesignExplorer setup names</td> </tr> <tr> <td><code>&lt;FileName&gt;</code></td> <td>String</td> <td>Must be a valid file path and name</td> </tr> </table>	Name <code>&lt;SetupName&gt;</code>	Type String	Description Must be one of existing DesignExplorer setup names	<code>&lt;FileName&gt;</code>	String	Must be a valid file path and name
Name <code>&lt;SetupName&gt;</code>	Type String	Description Must be one of existing DesignExplorer setup names					
<code>&lt;FileName&gt;</code>	String	Must be a valid file path and name					
<b>Return Value</b>	None						

<b>Python Syntax</b>	<code>ExportDXConfigFile (&lt;SetupName&gt;, &lt;FileName&gt;)</code>
----------------------	-----------------------------------------------------------------------

**Python Example**

```
oModule.ExportDXConfigFile ("DesignXplorerSetup1",
    "c:/exportdir/DXSetup1.xml")
```

**VB Syntax**

```
ExportDXConfigFile <SetupName>, <FileName>
```

**VB Example**

```
oModule.ExportDXConfigFile ("DesignXplorerSetup1",
    "c:/exportdir/DXSetup1.xml")
```

## ExportOptimetricsProfile

Export Optimetrics profile data

<b>UI Access</b>	Right click on the Optimetrics setup in the project tree and choose <b>View Analysis Result...</b> On the <b>Post Analysis Display</b> dialog box, click the <b>Profile</b> tab and click on the <b>Export</b> button.												
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;SetupName&gt;</td><td>String</td><td>Must be one of the existing Parametric, Optimization, Sensitivity, Statistical or DesignXplorer setup names</td></tr><tr><td>&lt;FileName&gt;</td><td>String</td><td>Must be a valid file path and name with extension of csv, tab, dat, or txt</td></tr><tr><td>[profileNum]</td><td>String</td><td>Must be a numeric string. Optional: defaulted to last profile number. It should be a zero indexed profile number.</td></tr></tbody></table>	Name	Type	Description	<SetupName>	String	Must be one of the existing Parametric, Optimization, Sensitivity, Statistical or DesignXplorer setup names	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt	[profileNum]	String	Must be a numeric string. Optional: defaulted to last profile number. It should be a zero indexed profile number.
Name	Type	Description											
<SetupName>	String	Must be one of the existing Parametric, Optimization, Sensitivity, Statistical or DesignXplorer setup names											
<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt											
[profileNum]	String	Must be a numeric string. Optional: defaulted to last profile number. It should be a zero indexed profile number.											
<b>Return Value</b>	None												

**Python Syntax**

```
ExportOptimetricsProfile (<SetupName>, <FileName>, [profileNum])
```

<b>Python Example</b>	<pre>oModule.ExportOptimetricsProfile ("StatisticalSetup1", "c:/exportdir/test.csv")</pre>
-----------------------	------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	ExportOptimetricsProfile <SetupName>, <FileName>, [profileNum]
<b>VB Example</b>	<pre>oModule.ExportOptimetricsProfile "StatisticalSetup1", "c:/exportdir/test.csv"</pre>

## ExportOptimetricsResult

Export an existing Optimization, Sensitivity, Statistical or DesignXplorer result. (Does not export Parametric results.)

<b>UI Access</b>	Right click on the desired Optimetrics setup in the project tree and choose <b>View Analysis Result...</b> On the <b>Post Analysis Display</b> dialog box, click the <b>Result</b> tab, then select <b>Table</b> view, and click on the <b>Export</b> button												
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SetupName&gt;</td> <td>String</td> <td>Must be one of the existing Optimization, Sensitivity, Statistical, or DesignXplorer setup names</td> </tr> <tr> <td>&lt;FileName&gt;</td> <td>String</td> <td>Must be a valid file path and name with extension of csv, tab, dat, or txt..</td> </tr> <tr> <td>[useFullOutputName]</td> <td>Boolean</td> <td>Optional: defaulted to false. If set to true values will be printed with units. This parameter is ignored for Optimization and Statistical results.</td> </tr> </tbody> </table>	Name	Type	Description	<SetupName>	String	Must be one of the existing Optimization, Sensitivity, Statistical, or DesignXplorer setup names	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt..	[useFullOutputName]	Boolean	Optional: defaulted to false. If set to true values will be printed with units. This parameter is ignored for Optimization and Statistical results.
Name	Type	Description											
<SetupName>	String	Must be one of the existing Optimization, Sensitivity, Statistical, or DesignXplorer setup names											
<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat, or txt..											
[useFullOutputName]	Boolean	Optional: defaulted to false. If set to true values will be printed with units. This parameter is ignored for Optimization and Statistical results.											
<b>Return Value</b>	None												

<b>Python Syntax</b>	ExportOptimetricsResult (<SetupName>, <FileName>, [useFullOutputName])
<b>Python Example</b>	<pre>oModule.ExportOptimetricsResult ("StatisticalSetup1", "c:/exportdir/test.csv", false)</pre>

<b>VB Syntax</b>	ExportOptimetricsResult <SetupName>, <FileName>, [useFullOutputName]
<b>VB Example</b>	<pre>oModule.ExportOptimetricsResult "StatisticalSetup1", "c:/exportdir/test.csv", false</pre>

## ExportParametricResults

Export existing Parametric results.

<b>UI Access</b>	Right click on the desired Parametric setup in the project tree and choose <b>View Analysis Result...</b> On the <b>Post Analysis Display</b> dialog box, click the <b>Result</b> tab, then select <b>Table</b> view, and click on the <b>Export</b> button.												
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;SetupName&gt;</td><td>String</td><td>Must be one of the existing Parametric setup names</td></tr><tr><td>&lt;FileName&gt;</td><td>String</td><td>Must be a valid file path and name with extension of csv, tab, dat or txt</td></tr><tr><td>&lt;bOutputUnits&gt;</td><td>Boolean</td><td>If set to true, values will be printed with units</td></tr></tbody></table>	Name	Type	Description	<SetupName>	String	Must be one of the existing Parametric setup names	<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat or txt	<bOutputUnits>	Boolean	If set to true, values will be printed with units
Name	Type	Description											
<SetupName>	String	Must be one of the existing Parametric setup names											
<FileName>	String	Must be a valid file path and name with extension of csv, tab, dat or txt											
<bOutputUnits>	Boolean	If set to true, values will be printed with units											
<b>Return Value</b>	None												

<b>Python Syntax</b>	ExportParametricResults (<SetupName>, <FileName>, <bOutputUnits>)
<b>Python Example</b>	<pre>oModule.ExportParametricResults ( "ParametricSetup1", "c:/exportdir/test.csv", False)</pre>

<b>VB Syntax</b>	ExportParametricResults <SetupName>, <FileName>, <bOutputUnits>
------------------	-----------------------------------------------------------------

<b>VB Example</b>	<pre> oModule.ExportParametricResults "ParametricSetup1", "c:/exportdir/test.csv", false </pre>
-------------------	-------------------------------------------------------------------------------------------------

## GetSetupNames [Optimetrics]

Gets a list of Optimetrics setup names

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>None</td> <td></td> <td></td> </tr> </tbody> </table>	Name	Type	Description	None		
Name	Type	Description					
None							
<b>Return Value</b>	IAnsoftCollectionObj – a collection of Optimetrics setup names						

<b>Python Syntax</b>	GetSetupNames()
<b>Python Example</b>	<pre> oModule = oDesign.GetModule("Optimetrics") setupNames = oModule.GetSetupNames() </pre>

<b>VB Syntax</b>	GetSetupNames()
<b>VB Example</b>	<pre> Set oModule_opt = oDesign.GetModule("Optimetrics") Set opt_setup_list = oModule.GetSetupNames() numsetups = setupnames.Count </pre>

## GetSetupNamesByType [Optimetrics]

Gets a list of Optimetrics setup names by type.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;Optimetrics type&gt;</td> <td>String</td> <td>Examples: parametric, optimization, statistical, sensitivity</td> </tr> </tbody> </table>	Name	Type	Description	<Optimetrics type>	String	Examples: parametric, optimization, statistical, sensitivity
Name	Type	Description					
<Optimetrics type>	String	Examples: parametric, optimization, statistical, sensitivity					
<b>Return Value</b>	Array of Optimetrics setup names of the given type.						

<b>Python Syntax</b>	GetSetupNamesByType (<Optimetrics type>)
<b>Python Example</b>	<pre>for name in oModule.GetSetupNamesByType("optimization")     AddInfoMessage(str(name))</pre>

<b>VB Syntax</b>	GetSetupNamesByType <Optimetrics type>
<b>VB Example</b>	<pre>For each name in oModule.GetSetupNamesByType("optimization")     MsgBox name Next</pre>

## ImportSetup

Import an Optimetric setup from a file.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SetupTypeName&gt;</td> <td>String</td> <td>Must be one of "OptiParametric", "OptiOptimization", "OptiSensitivity", "OptiS-</td> </tr> </tbody> </table>	Name	Type	Description	<SetupTypeName>	String	Must be one of "OptiParametric", "OptiOptimization", "OptiSensitivity", "OptiS-
Name	Type	Description					
<SetupTypeName>	String	Must be one of "OptiParametric", "OptiOptimization", "OptiSensitivity", "OptiS-					

		tatistical", or "OptiDesignExplorer".
	<SetupInfo>	<p>Array("NAME:&lt;SetupName&gt;", "FilePath")</p> <p>&lt;SetupName&gt;</p> <p>Type: &lt;string&gt;</p> <p>Name of the setup.</p> <p>&lt;FilePath&gt;</p> <p>Type : &lt;string: file path&gt;</p> <p>Must be a valid file path and name.</p>
<b>Return Value</b>	None	

<b>Python Syntax</b>	ImportSetup (<SetupTypeName>, <SetupInfo>)
<b>Python Example</b>	<pre>oModule.ImportSetup ("OptiStatistical", ["NAME:StatisticalSetup1", "c:/importdir/mySetupInfoFile"])</pre>

<b>VB Syntax</b>	ImportSetup <SetupTypeName>, <SetupInfo>
<b>VB Example</b>	<pre>oModule.ImportSetup "OptiStatistical", Array("NAME:StatisticalSetup1", "c:/importdir/mySetupInfoFile")</pre>

## PasteSetup [Optimetrics]

Pastes the specified Optimetrics setup.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;SetupName&gt;</td><td>String</td><td>Name of the Setup</td></tr></tbody></table>	Name	Type	Description	<SetupName>	String	Name of the Setup
Name	Type	Description					
<SetupName>	String	Name of the Setup					
<b>Return Value</b>	None						

<b>Python Syntax</b>	PasteSetup (<SetupName>)
<b>Python Example</b>	<code>oModule.PasteSetup ("OptimizationSetup1")</code>

<b>VB Syntax</b>	PasteSetup <SetupName>
<b>VB Example</b>	<code>oModule.PasteSetup "OptimizationSetup1"</code>

## RenameSetup [Optimetrics]

Renames the specified Optimetrics setup.

<b>UI Access</b>	Right-click the setup in the project tree, and then click <b>Rename</b> on the shortcut menu.									
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;OldName&gt;</td><td>String</td><td>The name that needs to be replaced</td></tr><tr><td>&lt;NewName&gt;</td><td>String</td><td>Replacement name</td></tr></tbody></table>	Name	Type	Description	<OldName>	String	The name that needs to be replaced	<NewName>	String	Replacement name
Name	Type	Description								
<OldName>	String	The name that needs to be replaced								
<NewName>	String	Replacement name								

<b>Return Value</b>	None
---------------------	------

<b>Python Syntax</b>	RenameSetup (<OldName> <NewName>)
<b>Python Example</b>	<code>oModule.RenameSetup ("OptimizationSetup1" "MyOptimization")</code>

<b>VB Syntax</b>	RenameSetup <OldName> <NewName>
<b>VB Example</b>	<code>oModule.RenameSetup "OptimizationSetup1" "MyOptimization"</code>

## SolveSetup [Optimetrics]

Solves the specified Optimetrics setup.

<b>UI Access</b>	Right-click the setup in the project tree, and then click <b>Analyze</b> on the shortcut menu.						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SetupName&gt;</td> <td>String</td> <td>Name of the setup to be solved</td> </tr> </tbody> </table>	Name	Type	Description	<SetupName>	String	Name of the setup to be solved
Name	Type	Description					
<SetupName>	String	Name of the setup to be solved					
<b>Return Value</b>	None						

<b>Python Syntax</b>	SolveSetup (<SetupName>)
<b>Python Example</b>	<code>oModule.SolveSetup ("OptimizationSetup1")</code>

<b>VB Syntax</b>	SolveSetup <SetupName>
<b>VB Example</b>	oModule.SolveSetup "OptimizationSetup1"

## SolveAllSetup

Solves all Optimetrics setups

<b>UI Access</b>	Right-click on <b>Optimetrics</b> in Project Manager and select <b>Analyze&gt;All</b> from context menu						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>None</td><td></td><td></td></tr></table>	Name	Type	Description	None		
Name	Type	Description					
None							
<b>Return Value</b>	None						

<b>Python Syntax</b>	SolveAllSetup()
<b>Python Example</b>	oModule.SolveAllSetup ()

<b>VB Syntax</b>	SolveAllSetup
<b>VB Example</b>	oModule.SolveAllSetup

## Parametric Script Commands

[EditSetup \[Parametric\]](#)

[ExportParametricSetupTable](#)

[GenerateVariationData \[Parametric\]](#)[InsertSetup \[Parametric\]](#)

## EditSetup [Parametric]

Modifies an existing parametric setup

<b>UI Access</b>	Right-click the setup in the project tree, and then click <b>Properties</b> on the shortcut menu.											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SetupName&gt;</td> <td>String</td> <td>Name of the Setup</td> </tr> <tr> <td></td> <td></td> <td></td> </tr> </tbody> </table>			Name	Type	Description	<SetupName>	String	Name of the Setup			
Name	Type	Description										
<SetupName>	String	Name of the Setup										
<b>Return Value</b>	None											

<b>Python Syntax</b>	EditSetup (<SetupName>, <ParametricParams>)
<b>Python Example</b>	See <a href="#">EditSetup [Optimization]</a>

<b>VB Syntax</b>	EditSetup <SetupName>, <ParametricParams>
<b>VB Example</b>	See <a href="#">EditSetup [Optimization]</a>

## ExportParametricSetupTable

Exports the parametric setup table as a CSV file.

<b>UI Access</b>	Double-click parametric setup. Select <b>Table</b> tab. Click <b>Export</b> .		
<b>Parameters</b>	Name	Type	Description

	<SetupName>	String	Name of the setup.
	<filePath>	String	Full path for file export.
<b>Return Value</b>	None		

<b>Python Syntax</b>	ExportParametricSetupTable (<SetupName>, <filePath>)
<b>Python Example</b>	oModule.ExportParametricSetupTable('ParametricSetup1', 'E:/Files/ParametricSetup1_Table.csv')

<b>VB Syntax</b>	ExportParametricSetupTable <SetupName>, <filePath>
<b>VB Example</b>	obj.ExportParametricSetupTable "ParametricSetup1", "E:/Files/ParametricSetup1_Table.csv"

## GenerateVariationData [Parametric]

Generate variation data before parametric solve for CAD integrated project

*Command:* Right click on the parametric setup in the project tree and choose "Generate Variation Data"

*Syntax:* GenerateVariationData <SetupName>

*Return Value:* None

*Parameters:* <SetupName>

Name of the setup.

*VB Example:*

```
oModule.GenerateVariationData "ParametricSetup1"
```

## InsertSetup [Parametric]

Inserts a new parametric setup.

UI Access	Right-click the <b>Optimetrics</b> folder in the project tree, and then click <b>Add&gt; Parametric</b> on the shortcut menu.		
Parameters	Name	Type	Description
	<Parametric Params>	Array	Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "Sim. Setups:=", <SimSetups>, <SweepDefs>, <SweepOps>, Array("NAME:Goals", Array("NAME:Goal", <OptiGoalSpec>), ... Array("NAME:Goal", <OptiGoalSpec>))
	<SetupName>	String	Name of the parametric setup.
	<SimSetups>	Array of Strings	An array of Twin Builder solution setup names.
	<SweepDefs>	Array	Array("NAME:Sweeps", Array("NAME:SweepDefinition", "Variable:=", <VarName>, "Data:=", <SweepData>, "Synchronize:=", <SyncNum>), ... Array("NAME:SweepDefinition", "Variable:=", <VarName>, "Data:=", <SweepData>,

		"Synchronize:=", <SyncNum>))
	<SweepData>	String "<SweepType>, <StartV>, <StopV>, <StepV>"
	<SweepType>	String The type of sweep data.
	<SyncNum>	Integer SweepDatas with the same value are synchronized.
	<SweepOps>	Array("NAME:Sweep Operations", "<OpType>:=, Array(<VarValue>, ..., <VarValue>), ... <OpType>:=, Array(<VarValue>, ..., <VarValue>))
	<OpType>	String The sweep operation type.
<b>Return Value</b>	None	

<b>Python Syntax</b>	InsertSetup ("OptiParametric", <ParametricParams>)
<b>Python Example</b>	<pre> oModule.InsertSetup (     "OptiParametric",     [ "NAME:ParametricSetup1", _          "SaveFields:=", true, _          [ "NAME:StartingPoint" ], _          "Sim. Setups:=", [ "Setup1" ], _          [ "NAME:Sweeps", _              [ "NAME:SweepDefinition", _                  "Variable:=", "\$width", _  </pre>

```
"Data:=", "LIN 12mm 17mm 2.5mm", _  
"OffsetF1:=", false, _  
"Synchronize:=", 0],  
["NAME:SweepDefinition", _  
"Variable:=", "$length", _  
"Data:=", "LIN 8mm 12mm 2mm", _  
"OffsetF1:=", false, _  
"Synchronize:=", 0]],  
["NAME:Sweep Operations"], _  
["NAME:Goals", _  
["NAME:Goal", _  
"Solution:=", "Setup1 : LastAdaptive", _  
"Calculation:=", "returnloss", _  
"Context:=", "", _  
["NAME:Ranges", _  
"Range:=", ["Var:=", "Freq", "Type:=", "s", _  
"Start:=", "8GHz", "Stop:=", "8GHz"]], _  
["NAME:Goal", _  
"Solution:=", "Setup1 : LastAdaptive", _  
"Calculation:=", "reflect", _
```

```

"Context:=", "", _
[ "NAME:Ranges", _
"Range:=", [ "Var:=", "Freq", "Type:=", "s", _
"Start:=", "8GHz", "Stop:=", "8GHz"]]]])

```

<b>VB Syntax</b>	InsertSetup "OptiParametric", <ParametricParams>
<b>VB Example</b>	<pre> oModule.InsertSetup "OptiParametric", Array("NAME:ParametricSetup1", _ "SaveFields:=", true, _ Array("NAME:StartingPoint"), _ "Sim. Setups:=", Array("Setup1"),_ Array("NAME:Sweeps", _ Array("NAME:SweepDefinition", _ "Variable:=", "\$width", _ &gt;Data:=", "LIN 12mm 17mm 2.5mm", _ "OffsetF1:=", false, _ "Synchronize:=", 0), Array("NAME:SweepDefinition", _ </pre>

```
"Variable:=", "$length", _  
"Data:=", "LIN 8mm 12mm 2mm", _  
"OffsetFl:=", false, _  
"Synchronize:=", 0)),  
Array("NAME:Sweep Operations"), _  
Array("NAME:Goals", _  
Array("NAME:Goal", _  
"Solution:=", "Setup1 : LastAdaptive", _  
"Calculation:=", "returnloss", _  
"Context:=", "", _  
Array("NAME:Ranges", _  
"Range:=", Array("Var:=", "Freq", "Type:=", "s", _  
"Start:=", "8GHz", "Stop:=", "8GHz"))), _  
Array("NAME:Goal", _  
"Solution:=", "Setup1 : LastAdaptive", _  
"Calculation:=", "reflect", _  
"Context:=", "", _  
Array("NAME:Ranges", _  
"Range:=", Array("Var:=", "Freq", "Type:=", "s", _  
"Start:=", "8GHz", "Stop:=", "8GHz")))))
```

## Optimization Script Commands

[EditSetup \[Optimization\]](#)

[InsertSetup \[Optimization\]](#)

### EditSetup [Optimization]

Modifies an existing optimization setup.

<b>UI Access</b>	Right-click the setup in the project tree, and then click Properties on the shortcut menu									
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;SetupName&gt;</td><td>String</td><td>Name of the Setup</td></tr><tr><td></td><td></td><td></td></tr></table>	Name	Type	Description	<SetupName>	String	Name of the Setup			
Name	Type	Description								
<SetupName>	String	Name of the Setup								
<b>Return Value</b>	None									

<b>Python Syntax</b>	EditSetup (<SetupName>, <OptimizationParams>)
<b>Python Example</b>	<pre>oModule.EditSetup("OptimizationSetup1", [     "NAME:OptimizationSetup1",     "UseFastCalculationUpdateAlgo:=", False,     "FastCalcOptCtrlledByUser:=", False,     "IsEnabled:=", True,     "SaveSolutions:=", False,</pre>

```
[  
  "NAME:StartingPoint"  
],  
  "Optimizer:=", "Quasi Newton",  
  [  
    "NAME:AnalysisStopOptions",  
    "StopForNumIteration:=", True,  
    "StopForElapsTime:=", False,  
    "StopForSlowImprovement:=", False,  
    "StopForGrdTolerance:=", False,  
    "MaxNumIteration:=", 1000,  
    "MaxSolTimeInSec:=", 3600,  
    "RelGradientTolerance:=", 0,  
    "MinNumIteration:=", 10  
],  
  "CostFuncNormType:=", "L2",  
  "PriorPSetup:=", "",  
  "PreSolvePSetup:=", True,  
  [  

```

```
[  
  "NAME:LCS"  
],  
[  
  "NAME:Goals",  
  [  
    "NAME:Goal",  
    "ReportType:=", "Standard",  
    "Solution:=", "TR",  
    [  
      "NAME:SimValueContext",  
      "SimValueContext:=", [1,0,2,0,False,False,-1,1,0,1,1,"",0,0  
    ]  
    "Calculation:=", "acosh(Time)",  
    "Name:=", "Time",  
    [  
      "NAME:Ranges",  
      "Range:=", ["Var:=", "Time", "Type:=", "a"]  
    ]  
  ]  
]
```

```
],
"Condition:=", "==",
[
"NAME:GoalValue",
"GoalValueType:=", "Independent",
"Format:=", "Real/Imag",
"bG:=", ["v:=", "[1;]"]
],
"Weight:=", "[1;]"
]
],
"Acceptable_Cost:=" , 0,
"Noise:=", 0.0001,
"UpdateDesign:=", False,
"UpdateIteration:=", 5,
"KeepReportAxis:=", True,
"UpdateDesignWhenDone:=", True
])
```

**VB Syntax**

EditSetup &lt;SetupName&gt;, &lt;OptimizationParams&gt;

**VB Example**

```
oModule>EditSetup "OptimizationSetup1",
Array("NAME:OptimizationSetup1", "UseFastCalculationUpdateAlgo:=", _
      false, "FastCalcOptCtrlledByUser:=", false,
".IsEnabled:=", true, "SaveSolutions:=", _
      false, Array("NAME:StartingPoint"), "Optimizer:=",
"Quasi Newton", Array("NAME:AnalysisStopOptions", "StopForNumIteration:=", _
      true, "StopForElapsTime:=", false,
"StopForSlowImprovement:=", false, "StopForGrdTolerance:=", _
      false, "MaxNumIteration:=", 1000, "MaxSoltTimeInSec:=",
3600, "RelGradientTolerance:=", _
      0, "MinNumIteration:=", 10), "CostFuncNormType:=",
"L2", "PriorPSetup:=", "", "PreSolvePSetup:=", _
      true, Array("NAME:Variables"), Array("NAME:LCS"),
Array("NAME:Goals", Array("NAME:Goal", "ReportType:=", _
      "Standard", "Solution:=", "TR", Array("NAME:SimValueContext",
"SimValueContext:=", Array( _
      1, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0)),
"Calculation:=", "Time", "Name:=", _
      "Time", Array("NAME:Ranges", "Range:=",
Array("Var:=", "Time", "Type:=", "a"))), "Condition:=", _
```

```

"==", Array("NAME:GoalValue", "GoalValueType:=",
"Independent", "Format:=", _
"Real/Imag", "bG:=", Array("v:=", "[1; ]")),
"Weight:=", "[1; ]"), "Acceptable_Cost:=", _
0, "Noise:=", 0.0001, "UpdateDesign:=", false,
"UpdateIteration:=", 5, "KeepReportAxis:=", _
true, "UpdateDesignWhenDone:=", true)

```

## InsertSetup [Optimization]

*Use:* Inserts a new optimization setup.

UI Access	Right-click the <b>Optimetrics</b> folder in the project tree, and then click <b>Add&gt;Optimization</b> on the shortcut menu.		
Parameters	Name <OptimizationParams>	Type Array	Description Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "Optimizer:=", <Optimizer>, "MaxIterations:=", <MaxIter>, "PriorPSetup:=", <PriorSetup>, "PreSolvePSetup:=", <Preceed>, <OptimizationVars>, <Constraint>, Array("NAME:Goals", Array("NAME:Goal", <OptiGoalSpec>, <OptimizationGoalSpec>), ... Array("NAME:Goal", <OptiGoalSpec>, <OptimizationGoalSpec>)),

		"Acceptable_Cost:=", <AcceptableCost>, "Noise:=", <Noise>, "UpdateDesignWhenDone:=", <UpdateDesign>
<OptimizationVars>	Array	Array ("NAME:Variables", "VarName:=", Array ("i:=", <IncludeVar>, "Min:=", <MinV>, "Max:=", <MaxV>, "MinStep:=", <MinStepV>, "MaxStep:=", <MaxStepV>), ..... "VarName:=", Array ("i:=", <IncludeVar>, "Min:=", <MinV>, "Max:=", <MaxV>, "MinStep:=", <MinStepV>, "MaxStep:=", <MaxStepV>) )
<MinStepV>	VarValue	The minimum step of the variable.
<MaxStepV>	VarValue	The maximum step of the variable.
<AcceptableCost>	Double	The acceptable cost value for the optimizer to stop.
<Noise>	Double	The noise of the design.
<UpdateDesign>	Boolean	Specifies whether or not to apply the optimal variation to the design after the optimization is done.
<OptimizationGoalSpec>	Array	"Condition:=", <OptimizationCond>, Array ("NAME:GoalValue", "GoalValeType:=", <GoalValueType>, "Format:=", <GoalValueFormat>, "bG:=", Array ("v:=", <GoalValue>)), "Weight:=", <Weight>)
<OptimizationCond>	String	Either "<=", "==" or ">="
<GoalValueType>	String	Either "Independent" or "Dependent"
<GoalValueFormat>	String	Either "Real/Imag" or "Mag/Ang".
<GoalValue>	String	Value in string. Value can be a real number, complex number, or expression.

<b>Return Value</b>	None
---------------------	------

**Command:** Right-click the **Optimetrics** folder in the project tree, and then click **Add>Optimization** on the shortcut menu.

**Syntax:** InsertSetup "OptiOptimization", <OptimizationParams>

**Return Value:** None

**Parameters:** <OptimizationParams>

```

Array("NAME:<SetupName>", "SaveFields:=",
<SaveField>, <StartingPoint>, "Optimizer:=",
<Optimizer>,
"MaxIterations:=", <MaxIter>, "PriorPSetup:=",
<PriorSetup>, "PreSolvePSetup:=", <Preceed>,
<OptimizationVars>, <Constraint>,
Array("NAME:Goals", Array("NAME:Goal",
<OptiGoalSpec>, <OptimizationGoalSpec>), ...
      Array("NAME:Goal", <OptiGoalSpec>,
<OptimizationGoalSpec>)),
"Acceptable_Cost:=", <AcceptableCost>, "Noise:=",
<Noise>, "UpdateDesignWhenDone:=", <UpdateDesign>

<OptimizationVars>
Array("NAME:Variables", "VarName:=", Array("i:=",
<IncludeVar>, "Min:=", <MinV>, "Max:=", <MaxV>,

```

```
"MinStep:=", <MinStepV>, "MaxStep:=", <MaxStepV>),  
..... "VarName:=", Array("i:=", <IncludeVar>, "Min:=", <MinV>, "Max:=", <MaxV>,  
"MinStep:=", <MinStepV>, "MaxStep:=", <MaxStepV>) )
```

<MinStepV>

Type : <VarValue>

The minimum step of the variable.

<MaxStepV>

Type: <VarValue>

The maximum step of the variable.

<AcceptableCost>

Type: <double>

The acceptable cost value for the optimizer to stop.

<Noise>

Type: <double>

The noise of the design.

```
<UpdateDesign>
```

Type: <bool>

Specifies whether or not to apply the optimal variation to the design after the optimization is done.

```
<OptimizationGoalSpec>
```

```
"Condition:=", <OptimizationCond>,
Array("NAME:GoalValue", "GoalValueType:=",
<GoalValueType>,
"Format:=", <GoalValueFormat>, "bG:=",
Array("v:=", <GoalValue>)), "Weight:=", <Weight>)
```

```
<OptimizationCond>
```

Type: <string>

Either "<=", "==", or ">="

```
<GoalValueType>
```

Type: <string>

Either "Independent" or "Dependent"

```
<GoalValueFormat>
```

Type:<string>

Either "Real/Imag" or "Mag/Ang".

<GoalValue>

Type: <string>

Value in string. Value can be a real number, complex number, or expression.

*VB Example:*

```
oModule.InsertSetup "OptiOptimization",  
    Array("NAME:OptimizationSetup1", _  
        "SaveFields:=", false, _  
        Array("NAME:StartingPoint", "$length:=", "8mm", _  
            "$width:=", "14.5mm"), _  
        "Optimizer:=", "Quasi Newton", _  
        "MaxIterations:=", 100, _  
        "PriorPSetup:=", "ParametricSetup1", _  
        "PreSolvePSetup:=", true, _  
        Array("NAME:Variables", _  
            "$length:=", Array("i:=", true, "Min:=", "6mm", _  
                "Max:=", "18mm", _  
                "MinStep:=", "0.001mm", "MaxStep:=", _  
                "1.2mm"), _
```

```
"$width:=", Array("i:=", true, "Min:=", _  
"6.5mm", "Max:=", "19.5mm", _  
"MinStep:=", "0.001mm", "MaxStep:=", _  
"1.3mm)), _  
    Array ("NAME:LCS"), _  
    Array ("NAME:Goals", _  
Array("NAME:Goal", _  
"Solution:=", "Setup1 : LastAdaptive", _  
"Calculation:=", "reflect", _  
"Context:=", "", _  
Array("NAME:Ranges", _  
"Range:=", Array("Var:=", "Freq", _  
"Type:=", "s", _  
"Start:=", "8GHz", "Stop:=", "8GHz")), _  
"Condition:=", "<=", _  
Array("NAME:GoalValue", _  
"GoalValueType:=", "Independent", _  
"Format:=", "Real/Imag", _  
"bG:=", Array("v:=", "[0.0001])), _  
"Weight:=", "[1"])),  
    "Acceptable_Cost:=", 0.0002, _
```

```
"Noise:=", 0.0001, _  
"UpdateDesign:=", true, _  
"UpdateIteration:=", 5, _  
"KeepReportAxis:=", true, _  
"UpdateDesignWhenDone:=", true)
```

Python Syntax	InsertSetup ("OptiOptimization", <OptimizationParams>)
Python Example	<pre>oModule.InsertSetup(     "OptiOptimization", _     [ "NAME:OptimizationSetup1", _         "SaveFields:=", false, _         [ "NAME:StartingPoint", "\$length:=", "8mm", _             "\$width:=", "14.5mm"], _         "Optimizer:=", "Quasi Newton", _         "MaxIterations:=", 100, _         "PriorPSetup:=", "ParametricSetup1", _         "PreSolvePSetup:=", true, _         [ "NAME:Variables", _             "\$length:=", [ "i:=", true, "Min:=", "6mm", _</pre>

```
"Max:=", "18mm", _
"MinStep:=", "0.001mm", "MaxStep:=", _
"1.2mm"], _
"$width:=", [{"i:=", true, "Min:=", _
"6.5mm", "Max:=", "19.5mm", _
"MinStep:=", "0.001mm", "MaxStep:=", _
"1.3mm"}], _
["NAME:LCS"], _
["NAME:Goals"], _
["NAME:Goal", _
"Solution:=", "Setup1 : LastAdaptive", _
"Calculation:=", "reflect", _
"Context:=", "", _
["NAME:Ranges"], _
"Range:=", [{"Var:=", "Freq", _
>Type:=", "s", _
"Start:=", "8GHz", "Stop:=", "8GHz"}], _
"Condition:=", "<=", _
["NAME:GoalValue"], _
"GoalValueType:=", "Independent", _
```

```
"Format:=", "Real/Imag", _  
"bG:=", ["v:=", "[0.0001]"]], _  
"Weight:=", "[1]]],  
"Acceptable_Cost:=", 0.0002, _  
"Noise:=", 0.0001, _  
"UpdateDesign:=", true, _  
"UpdateIteration:=", 5, _  
"KeepReportAxis:=", true, _  
"UpdateDesignWhenDone:=", true)
```

VB Syntax	InsertSetup "OptiOptimization", <OptimizationParams>
VB Example	<pre>oModule.InsertSetup "OptiOptimization",_ Array("NAME:OptimizationSetup1", _ "SaveFields:=", false, _ Array("NAME:StartingPoint", "\$length:=", "8mm", _ "\$width:=", "14.5mm"), _ "Optimizer:=", "Quasi Newton", _ "MaxIterations:=", 100, _</pre>

```
"PriorPSetup:=", "ParametricSetup1", _  
"PreSolvePSetup:=", true, _  
Array("NAME:Variables", _  
"$length:=", Array("i:=", true, "Min:=", "6mm", _  
"Max:=", "18mm", _  
"MinStep:=", "0.001mm", "MaxStep:=", _  
"1.2mm"), _  
"$width:=", Array("i:=", true, "Min:=", _  
"6.5mm", "Max:=", "19.5mm", _  
"MinStep:=", "0.001mm", "MaxStep:=", _  
"1.3mm")), _  
Array("NAME:LCS"), _  
Array("NAME:Goals"), _  
Array("NAME:Goal", _  
"Solution:=", "Setup1 : LastAdaptive", _  
"Calculation:=", "reflect", _  
"Context:=", "", _  
Array("NAME:Ranges", _  
"Range:=", Array("Var:=", "Freq", _  
"Type:=", "s", _
```

```
"Start:=", "8GHz", "Stop:=", "8GHz")) , _  
"Condition:=", "<=", _  
Array("NAME:GoalValue", _  
"GoalValueType:=", "Independent", _  
"Format:=", "Real/Imag", _  
"bG:=", Array("v:=", "[0.0001]")), _  
"Weight:=", "[1]"),  
"Acceptable_Cost:=", 0.0002, _  
"Noise:=", 0.0001, _  
"UpdateDesign:=", true, _  
"UpdateIteration:=", 5, _  
"KeepReportAxis:=", true, _  
"UpdateDesignWhenDone:=", true)
```

## Sensitivity Script Commands

[EditSetup \[Sensitivity\]](#)

[InsertSetup \[Sensitivity\]](#)

### **EditSetup [Sensitivity]**

Modifies an existing sensitivity setup.

<b>UI Access</b>	Right-click the setup in the project tree, and then click Properties on the shortcut menu									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;SetupName&gt;</td> <td>String</td> <td>Name of the Setup</td> </tr> <tr> <td></td> <td></td> <td></td> </tr> </tbody> </table>	Name	Type	Description	<SetupName>	String	Name of the Setup			
Name	Type	Description								
<SetupName>	String	Name of the Setup								
<b>Return Value</b>	None									

<b>Python Syntax</b>	EditSetup (<SetupName>, <SensitivityParams>)
<b>Python Example</b>	<pre> oModule.EditSetup("OptimizationSetup1", [     "NAME:OptimizationSetup1",     "UseFastCalculationUpdateAlgo:=", False,     "FastCalcOptCtrlledByUser:=", False,     "IsEnabled:=", True,     "SaveSolutions:=", False,     [         "NAME:StartingPoint"     ],     "Optimizer:=", "Quasi Newton",     [         "NAME:AnalysisStopOptions",     ] ]) </pre>

```
"StopForNumIteration:=", True,  
"StopForElapsTime:=", False,  
"StopForSlowImprovement:=", False,  
"StopForGrdTolerance:=" , False,  
"MaxNumIteration:=", 1001,  
"MaxSolTimeInSec:=", 3600,  
"RelGradientTolerance:=", 0,  
"MinNumIteration:=", 10  
,  
"CostFuncNormType:=", "L2",  
"PriorPSetup:=", "",  
"PreSolvePSetup:=", True,  
[  
"NAME:Variables"  
,  
[  
"NAME:LCS"  
,  
[
```

```
"NAME:Goals",
[
  "NAME:Goal",
    "ReportType:=", "Standard",
    "Solution:=", "TR3",
    [
      "NAME:SimValueContext",
        "SimValueContext:=", [1,0,2,0,False,False,-1,1,0,1,1,"",0,0]
      ],
      "Calculation:=", "mag(DIFF1.VAL)",
      "Name:=", "DIFF1.VAL",
      [
        "NAME:Ranges",
          "Range:=", [
            "Var:=", "Time", "Type:=", "a"]
          ],
        "Condition:=", "==",
        [
          "NAME:GoalValue",
            "GoalValueType:=", "Independent",
            "Format:=", "Real/Imag",
```

```
"bG:=", ["v:=", "[1;]"]  
],  
"Weight:=", "[1;]"  
]  
],  
"Acceptable_Cost:=", 0,  
"Noise:=", 0.0001,  
"UpdateDesign:=", False,  
"UpdateIteration:=", 5,  
"KeepReportAxis:=", True,  
"UpdateDesignWhenDone:=", True  
])
```

<b>VB Syntax</b>	EditSetup <SetupName>, <SensitivityParams>
<b>VB Example</b>	<pre>oModule&gt;EditSetup("OptimizationSetup1",     Array(         "NAME:OptimizationSetup1",         "UseFastCalculationUpdateAlgo:=", False,         "FastCalcOptCtrlledByUser:=", False,</pre>

```
".IsEnabled:=", True,  
"SaveSolutions:=", False,  
Array(  
    "NAME:StartingPoint"  
)  
,"Optimizer:=", "Quasi Newton",  
Array(  
    "NAME:AnalysisStopOptions",  
    "StopForNumIteration:=", True,  
    "StopForElapsTime:=", False,  
    "StopForSlowImprovement:=", False,  
    "StopForGrdTolerance:=", False,  
    "MaxNumIteration:=", 1001,  
    "MaxSolTimeInSec:=", 3600,  
    "RelGradientTolerance:=", 0,  
    "MinNumIteration:=", 10  
)  
,"CostFuncNormType:=", "L2",  
"PriorPSetup:=", "",  
"PreSolvePSetup:=", True,  
Array(
```

```
"NAME:Variables"
),
Array(
"NAME:LCS"
),
Array(
"NAME:Goals",
Array(
"NAME:Goal",
"ReportType:=", "Standard",
"Solution:=", "TR3",
Array(
"NAME:SimValueContext",
"SimValueContext:=",
Array(1,0,2,0,False,False,-1,1,0,1,1,"",0,0)
),
"Calculation:=", "mag(DIFF1.VAL)",
"Name:=", "DIFF1.VAL",
Array(
```

```
"NAME:Ranges",
"Range:=", Array("Var:=", "Time", "Type:=", "a")
),
"Condition:=", "==",
Array(
"NAME:GoalValue",
"GoalValueType:=", "Independent",
"Format:=", "Real/Imag",
"bG:=", Array("v:=", "Array(1;]")
),
"Weight:=", "Array(1;]"
)
),
"Acceptable_Cost:=", 0,
"Noise:=", 0.0001,
"UpdateDesign:=", False,
"UpdateIteration:=", 5,
"KeepReportAxis:=", True,
"UpdateDesignWhenDone:=", True
))
```

## InsertSetup [Sensitivity]

Inserts a new sensitivity setup.

UI Access	Right-click <b>Optimetrics</b> in the project tree, and then click <b>Add&gt;Sensitivity</b> on the shortcut menu.		
<b>Parameters</b>	<b>Name</b>	Type	Description
	<SensitivityParams>	Array	<pre>Array("NAME:&lt;SetupName&gt;", "SaveFields:=",       &lt;SaveField&gt;, &lt;StartingPoint&gt;, "MaxIterations:=",       &lt;MaxIter&gt;, "PriorPSetup:=", &lt;PriorSetup&gt;,       "PreSolvePSetup:=", &lt;Preceed&gt;, &lt;SensitivityVars&gt;,       &lt;Constraint&gt;,       Array("NAME:Goals", Array("NAME:Goal",       &lt;OptiGoalSpec&gt;), ..., Array("NAME:Goal",       &lt;OptiGoalSpec&gt;)), "Primary Goal:=". &lt;PrimaryGoalID&gt;,       "PrimaryError:=", &lt;PrimaryError&gt;)</pre>
	<b>&lt;SensitivityVars&gt;</b>	Array	<pre>Array("NAME:Variables",       "VarName:=", Array("i:=", &lt;IncludeVar&gt;,       "Min:=", &lt;MinV&gt;, "Max:=", &lt;MaxV&gt;,       "IDisp:=", &lt;InitialDisp&gt;),       "VarName:=", Array("i:=", &lt;IncludeVar&gt;,       "Min:=", &lt;MinV&gt;, "Max:=", &lt;MaxV&gt;,       "IDisp:=", &lt;InitialDisp&gt;))</pre>

	<InitialDisp>	VarValue	Index of the Primary goal. Index starts from zero.
	<PrimaryError>	Double	Error associated with the Primary goal.
<b>Return Value</b>	None		

<b>Python Syntax</b>	InsertSetup ("OptiSensitivity", <SensitivityParams>)
<b>Python Example</b>	<pre> oModule.InsertSetup (     "OptiSensitivity", _     ["NAME:SensitivitySetup1", _      "SaveFields:=", true, _      ["NAME:StartingPoint"], _      "MaxIterations:=", 20, _      "PriorPSetup:=", "", _      "PreSolvePSetup:=", true, _      ["NAME:Variables"], _      ["NAME:LCS"], _      "NAME:Goals", _      ["NAME:Goal", _       "Solution:=", "Setup1 : LastAdaptive", _       "Calculation:=", "returnloss", _       "Context:=", "", _</pre>

```
[ "NAME:Ranges",_
  "Range:=", [ "Var:=", "Freq", "_"
  Type:=", "s",_
  "Start:=", "8GHz", "Stop:=", "8GHz"]]],_
  [ "NAME:Goal",_
  "Solution:=", "Setup1 : LastAdaptive",_
  "Calculation:=", "reflect",_
  "Context:=", "",_
  [ "NAME:Ranges",_
  "Range:=", [ "Var:=", "Freq", _
  "Type:=", "s",_
  "Start:=", "8GHz", "Stop:=", "8GHz"]]],_
  "Primary Goal:=", 1,_
  "PrimaryError:=", 0.001])
```

<b>VB Syntax</b>	InsertSetup "OptiSensitivity", <SensitivityParams>
<b>VB Example</b>	<pre>oModule.InsertSetup "OptiSensitivity", _ Array("NAME:SensitivitySetup1", _</pre>

```
"SaveFields:=", true, _  
    Array("NAME:StartingPoint"), _  
    "MaxIterations:=", 20, _  
    "PriorPSetup:=", "", _  
    "PreSolvePSetup:=", true, _  
    Array("NAME:Variables"), _  
    Array("NAME:LCS"), _  
    Array("NAME:Goals"), _  
    Array("NAME:Goal", _  
        "Solution:=", "Setup1 : LastAdaptive", _  
        "Calculation:=", "returnloss", _  
        "Context:=", "", _  
        Array("NAME:Ranges", _  
            "Range:=", Array("Var:=", "Freq", " _  
                Type:=", "s", _  
                "Start:=", "8GHz", "Stop:=", "8GHz"))), _  
        Array("NAME:Goal", _  
            "Solution:=", "Setup1 : LastAdaptive", _  
            "Calculation:=", "reflect", _  
            "Context:=", "", _
```

```
Array("NAME:Ranges",_
      "Range:=", Array("Var:=", "Freq",_
      "Type:=", "s",_
      "Start:=", "8GHz", "Stop:=", "8GHz"))),_
      "Primary Goal:=", 1,_
      "PrimaryError:=", 0.001)
```

## Statistical Script Commands

[EditSetup \[Statistical\]](#)

[InsertSetup Statistical](#)

### EditSetup [Statistical]

Modifies an existing statistical setup.

<b>UI Access</b>	Right-click the setup in the project tree, and clickProperties on the shortcut menu.	
<b>Parameters</b>	Name <SetupName>	Type String
<b>Return Value</b>	None	

**Python Syntax** EditSetup (<SetupName>, <StatisticalParams>)

**Python Example** See [EditSetup \[Optimization\]](#)

<b>VB Syntax</b>	EditSetup <SetupName>, <StatisticalParams>
<b>VB Example</b>	See EditSetup [Optimization]

*Example:*

```

Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule

Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject("optiguides")
Set oDesign = oProject.SetActiveDesign("HFSSModel1")
Set oModule = oDesign.GetModule("Optimetrics")

oModule.EditSetup "StatisticalSetup1", Array("NAME:StatisticalSetup1",
Array("NAME:ProdOptiSetupData",
"SaveFields:=", true, "CopyMesh:=", false),
Array("NAME:StartingPoint", "$length:=", "7.824547736mm",
"$width:=", "14.8570192mm"),

```

```
"MaxIterations:=", 50,
"PriorPSetup:=", "",

Array("NAME:Variables",
"$length:", Array("i:=", true,
"int:", false,
"Dist:=", "Uniform",
"Tol:=", "10%", "StdD:=", "0.2mm", "Min:=", "-3",
"Max:=", "3", "Shape:=", "1", "Scale:=", "0.04mm",
"Location:=", "0.4mm",
"Dataset:=", "", "LatinHypercube:=", "true", "VarMin:=", "0.2mm", "VarMax:=", "0.6mm", "Prob:=",
"0.01",
"Mean:=", "0.4mm"),

"$width:", Array("i:=", true,
"int:", false,
"Dist:=", "Gaussian",
"Tol:=", "10%",
"StdD:=", "0.2mm",
"Min:=", "-3", "Max:=", "3",
"Shape:=", "1",
"Scale:=", "0.04mm",
"Location:=", "0.4mm",
"Dataset:=", "",
"LatinHypercube:=", "true",
"VarMin:=", "0.2mm", "VarMax:=", "0.6mm",
"Prob:=", "0.02",
"Mean:=", "0.4mm")),

Array("NAME:Goals", Array("NAME:Goal",
"ReportType:=", "Modal Solution Data",
"Solution:=", "Setup1 : PortOnly",
Array("NAME:SimValueContext", "Domain:=", "Sweep"),
```

```

"Calculation:=", "returnloss",
"Name:=", "returnloss",

Array("NAME:Ranges",
"Range:=", Array("Var:=", "Freq",
>Type:=", "s",
"Start:=", "8.2GHz", "Stop:=", "0")),

Array("NAME:Goal",
"ReportType:=", "Modal Solution Data",
"Solution:=", "Setup1 : PortOnly",
Array("NAME:SimValueContext",
"Domain:=", "Sweep"),
"Calculation:=", "reflect",
>Name:=", "reflect",
Array("NAME:Ranges",
"Range:=", Array("Var:=", "Freq",
>Type:=", "s",
"Start:=", "8.2GHz", "Stop:=", "0")))
)

```

## InsertSetup [Statistical]

Inserts a new statistical setup.

<b>UI Access</b>	Right-click <b>Optimetrics</b> in the project tree, and then click <b>Add&gt;Statistical</b> on the shortcut menu.		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	<StatisticalParams>	Array	Array("NAME:<SetupName>", "SaveFields:=", <SaveField>, <StartingPoint>, "MaxIterations:=", <MaxIter>, "PriorPSetup:=", <PriorSetup>, "PreSolvePSetup:=", <Preceed>, <StatisticalVars>, Array("NAME:Goals", Array("NAME:Goal",

		<OptiGoalSpec>), ..., Array("NAME:Goal", <OptiGoalSpec>)))
	<StatisticalVars>	Array Array ("NAME:Variables", "VarName:=", Array("i:=", <IncludeVar>, "Dist:=", <DistType>, "Tol:=", <Tolerance>, "StdD:=", <StdD>, "Min:=", <MinCutoff>, "Max:=", <MaxCutoff>, ... "VarName:=", Array("i:=", <IncludeVar>, "Dist:=", <DistType>, "Tol:=", <Tolerance>, "StdD:=", <StdD>, "Min:=", <MinCutoff>, "Max:=", <MaxCutoff>))
	<DistType>	String Distrbution can be "Gaussian" or "Uniform".
	<Tolerance>	VarValue The tolerance for the variable when distribution is Uniform
	<StdD>	VarValue The standard deviation for the variable when distribution is Gaussian.
	<MinCutoff>	Double The minimum cut-off for the variable when distribution is Gaussian.
	<MaxCutoff>	Double The maximum cut-off for the variable when distribution is Gaussian.
<b>Return Value</b>	None	

<b>Python Syntax</b>	InsertSetup ("OptiStatistical", <StatisticalParams>)
<b>Python Example</b>	oModule.InsertSetup( "OptiStatistical", _

```
[ "NAME:StatisticalSetup1", _  
  "SaveFields:=", true, _  
  [ "NAME:StartingPoint"],_  
  "MaxIterations:=", 50,_  
  "PriorPSetup:=", "", _  
  [ "NAME:Variables"], _  
  [ "NAME:Goals", _  
    [ "NAME:Goal", _  
      "Solution:=", "Setup1 : LastAdaptive", _  
      "Calculation:=", "returnloss", _  
      "Context:=", "", _  
      [ "NAME:Ranges", _  
        "Range:=", [ "Var:=", "Freq", _  
          "Type:=", "s", _  
          "Start:=", "8GHz", "Stop:=", "8GHz"]]],_  
      [ "NAME:Goal", _  
        "Solution:=", "Setup1 : LastAdaptive", _  
        "Calculation:=", "reflect", _  
        "Context:=", "", _  
        [ "NAME:Ranges", _
```

```
"Range:=", [ "Var:=", "Freq", "Type:=", _
"s", "Start:=", "8GHz", "Stop:=", "8GHz" ] ] )
```

VB Syntax	InsertSetup "OptiStatistical", <StatisticalParams>
	<pre>oModule.InsertSetup     "OptiStatistical", _     Array("NAME:StatisticalSetup1", _         "SaveFields:=", true, _         Array("NAME:StartingPoint"), _         "MaxIterations:=", 50, _         "PriorPSetup:=", "", _         Array("NAME:Variables"), _         Array("NAME:Goals"), _         Array("NAME:Goal"), _         "Solution:=", "Setup1 : LastAdaptive", _         "Calculation:=", "returnloss", _         "Context:=", "", _         Array("NAME:Ranges", _             "Range:=", Array("Var:=", "Freq", _</pre>
<b>VB Example</b>	

```
"Type:=", "s",_
"Start:=", "8GHz", "Stop:=", "8GHz"))),_
Array("NAME:Goal",_
"Solution:=", "Setup1 : LastAdaptive",_
"Calculation:=", "reflect",_
"Context:=", "", _
Array("NAME:Ranges",_
"Range:=", Array("Var:=", "Freq", "Type:=",_
"s", "Start:=", "8GHz", "Stop:=", "8GHz"))))
```

*Example:*

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
```

```
Set oProject = oDesktop.SetActiveProject("OptimTee")
Set oDesign = oProject.SetActiveDesign("TeeModel")

oDesign.ChangeProperty Array("NAME:AllTabs",
Array("NAME:LocalVariableTab",
Array("NAME:PropServers", "LocalVariables"),
Array("NAME:ChangedProps",
Array("NAME:offset",
Array("NAME:Statistical", "Included:=", true)))))

Set oModule = oDesign.GetModule("Optimetrics")

oModule.InsertSetup "OptiStatistical", Array("NAME:StatisticalSetup1",
Array("NAME:ProdOptiSetupData",
"SaveFields:=", false, "CopyMesh:=", false),
Array("NAME:StartingPoint", "offset:=", "0in"),
"MaxIterations:=", 50, "PriorPSetup:=", "",
Array("NAME:Variables",
"offset:=", Array("i:=", true,
"int:=", false,
"Dist:=", "Gaussian",
"Tol:=", "10%",
"StdD:=", ".5in",
"Min:=", "-3",
"Max:=", "3",
"Shape:=", "1",
"Scale:=", "0in",
"Location:=", "0in",
"Dataset:=", ""),
"LatinHypercube:=", "true",
"VarMin:=", "-lin",
"VarMax:=", "lin",
```

```
"Prob:=", "0.01",
"Mean:=", "0in")),

Array("NAME:Goals", Array("NAME:Goal",
"ReportType:=", "Modal Solution Data",
"Solutions:=", "Setup1 : LastAdaptive", Array("NAME:SimValueContext"),
"Calculation:=", "Power11",
"Name:=", "Power11",
Array("NAME:Ranges",
"Range:=", Array("Var:=", "Freq", "Type:=", "d",
"DiscreteValues:=", "10GHz"))))
```

### For Q3D Extractor and Circuit the command details are as follows:

Inserts a new statistical setup.

*Command:* Right-click **Optimetrics** in the project tree, and then click **Add>Statistical** on the shortcut menu.

*Syntax:* InsertSetup "OptiStatistical", <StatisticalParams>

*Return Value:* None

*Parameters:* <StatisticalParams>

```
Array("NAME:<SetupName>", "SaveFields:=",
<SaveField>, <StartingPoint>, "MaxIterations:=",
<MaxIter>, "PriorPSetup:=", <PriorSetup>,
"PreSolvePSetup:=", <Preceed>, <StatisticalVars>,
Array("NAME:Goals", Array("NAME:Goal",
<OptiGoalSpec>), ..., Array("NAME:Goal",
<OptiGoalSpec>)) ) ,
```

```
<StatisticalVars>
    Array("NAME:Variables",
        "VarName:=", Array("i:=", <IncludeVar>, "Dist:=",
<DistType>, "Tol:=", <Tolerance>,
        "StdD:=", <StdD>, "Min:=", <MinCutoff>, "Max:=",
        <MaxCutoff>, ...
        "VarName:=", Array("i:=", <IncludeVar>, "Dist:=",
<DistType>, "Tol:=", <Tolerance>, "StdD:=",
        <StdD>, "Min:=", <MinCutoff>, "Max:=",
        <MaxCutoff>))
```

**Parameters:**

```
<DistType>
    Type : <string>
```

Distrbution can be "Gaussian" or "Uniform".

```
<Tolerance>
```

```
    Type: <VarValue>
```

The tolerance for the variable when distribution is Uniform.

```
<StdD>
```

```
    Type: <VarValue>
```

The standard deviation for the variable when distribution is Gaussian.

```
<MinCutoff>
  Type: <double>
    The minimum cut-off for the variable when distribution is Gaussian.

<MaxCutoff>
  Type: <double>
    The maximum cut-off for the variable when distribution is Gaussian.

Example: oModule.InsertSetup "OptiStatistical", _
  Array("NAME:StatisticalSetup1", _
    "SaveFields:=", true, _
    Array("NAME:StartingPoint"), _
    "MaxIterations:=", 50, _
    "PriorPSetup:=", "", _
    Array("NAME:Variables"), _
    Array("NAME:Goals", _
      Array("NAME:Goal", _
        "Solution:=", "Setup1 : LastAdaptive", _
        "Calculation:=", "returnloss", _
        "Context:=", "", _
        Array("NAME:Ranges", _
          "Range:=", Array("Var:=", "Freq", _
            "Type:=", "s",_
```

```
"Start:=", "8GHz", "Stop:=", "8GHz"))),_
Array("NAME:Goal",_
"Solution:=", "Setup1 : LastAdaptive",_
"Calculation:=", "reflect",_
"Context:=", "", _
Array("NAME:Ranges",_
"Range:=", Array("Var:=", "Freq", "Type:=", _
"s", "Start:=", "8GHz", "Stop:=", "8GHz"))))
```

# 17 - Solutions Module Script Commands

Solutions commands should be executed by the "Solutions" module.

```
Set oModule = oDesign.GetModule("Solutions")
```

```
oModule.CommandName <args>
```

[DeleteSolutionVariation](#)

[ExportNetworkData](#)

GetAvailableVariations

[GetValidISolutionList](#)

[TDROnReport](#)

## DeleteSolutionVariation

Deletes all solution data for specific solutions and design variations. This is obsolete and is supported only for backward compatibility. You should use DeleteFullVariation. See also [DeleteVariation](#).

<b>UI Access</b>	Right-click on <b>Results</b> , select <b>Browse Solutions...</b> , click <b>Delete</b> button in the dialog.		
<b>Parameters</b>	Name <i>&lt;SoluParams&gt;</i>	Type Array	Description Structured array.  Array (<DataSpecifierArray>, ...)
	<i>&lt;DataSpecifierArray&gt;</i>	Array	Structured array.  Array (<DesignVariationKey>, <SetupName>, <SolnName>)
<b>Return Value</b>	None.		

<b>Python Syntax</b>	DeleteSolutionVariation(<SoluParam>)
<b>Python Example</b>	<pre>oModule.DeleteSolutionVariation([     ["width='2in'", "Setup1", "Adaptive_1"],     ["width='2in'", "Setup1", "Sweep1"] ])</pre>

<b>VB Syntax</b>	DeleteSolutionVariation <SoluParam>
<b>VB Example</b>	<pre>oModule.DeleteSolutionVariation Array( _     Array("width='2in'", "Setup1", "Adaptive_1"), _     Array("width='2in'", "Setup1", "Sweep1"))</pre>

## GetValidSolutionList

Gets all available solution names that exist in a design.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <IncludeImportedSolutions>	Type Boolean	Description Optional. Specifies whether to include imported solutions. Default is false.
<b>Return Value</b>	Array of strings containing solution names.		

<b>Python Syntax</b>	GetValidISolutionList([Optional < <i>IncludeImportedSolutions</i> >])
<b>Python Example</b>	oModule.GetValidISolutionList()

<b>VB Syntax</b>	GetValidISolutionList [Optional < <i>IncludeImportedSolutions</i> >]
<b>VB Example</b>	oModule.GetValidISolutionList

This page intentionally  
left blank.

# 18 - Field Overlays Module Script Commands

Field overlay commands should be executed by the Field Overlays module, which is called "FieldsReporter" in scripts.

```
Set oModule = oDesign.GetModule("FieldsReporter")
oModule.CommandName <args>
```

[AddMarkerToPlot](#)

[AddNamedExpr](#)

[CreateFieldPlot](#)

[DeleteFieldPlot](#)

[EditSurfaceMeshSummaryData](#)

[ExportMarkerTable](#)

[ExportPlotImageWithViewToFile](#)

[ExportSurfaceMeshSummary](#)

[GetFieldPlotName](#)

[ModifyFieldPlot](#)

[ModifyInceptionParameters](#)

[RenameFieldPlot](#)

[RenamePlotFolder](#)

[SetFieldPlotSettings](#)

[SetPlotFolderSettings](#)

[SetPlotsSolutionsContext](#)

[SetPlotsViewSolutionContext](#)

[UpdateAllFieldsPlots](#)

[UpdateQuantityFieldsPlots](#)

## AddMarkerToPlot

Adds a marker to a trace on a named field plot.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<Location>	Array	Array of strings containing X,Y, and Z coordinates for the marker.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	AddMarkerToPlot( <Location>, <PlotName> )
<b>Python Example</b>	<pre>oModule.AddMarkerToPlot (["0.290455877780914in", "-0.616900205612183in", "1.77635683940025e-015in"], "Mag_H1") oModule.AddMarkerToPlot (["-0.317279517650604in",</pre>

```
"1.22481322288513in", "0in"],  
"Mag_H1")
```

<b>VB Syntax</b>	AddMarkerToPlot <Location>, <PlotName>
<b>VB Example</b>	<pre>oModule.AddMarkerToPlot _     Array("0.290455877780914in", _         "-0.616900205612183in", "1.77635683940025e-015in"), _         "Mag_E1" oModule.AddMarkerToPlot _     Array("-0.317279517650604in", _         "1.22481322288513in", "0in"), _         "Mag_E1"</pre>

## CreateFieldPlot

**Note:** Use in conjunction with [GetGeometryIdsForNetLayerCombinations](#) and [GetGeometryIdsForAllNetLayerCombinations](#).

- [GetGeometryIdsForNetLayerCombinations](#) returns ID numbers of all faces and edges of the active design related to the current target combination of nets and layers.
- [GetGeometryIdsForAllNetLayerCombinations](#) returns ID numbers for all possible combinations of nets and layers it is possible to choose.

**Use:** Creates a field/mesh plot "Field" or a visual ray trace ("VRT") plot.

**Command:** **HFSS>Fields>Plot Fields><field\_quantity>**

**Command:** **Maxwell3D or Maxwell2D>Fields>Fields><field\_quantity>**

**Command:** **Maxwell3D>Fields>Fields>Field Line Trace** (for 3D Electrostatic designs - see [Examples](#) below)

**Command:** **Icepak>Fields>Plot Fields><field\_quantity>**

**Syntax:** CreateFieldPlot <PlotParameterArray> ["Field" | "VRT"]

**Return Value:** None

**Parameters:** <PlotParameterArray> "Field"

```
Array("NAME:<PlotName>",
      "SolutionName:=", <string>,
      "QuantityName:=", <string>,
```

```
"PlotFolder:=", <string>,
"UserSpecifyName:=", <int>,
"UserSpecifyFolder:=", <int>,
"IntrinsicVar:=", <string>,
"PlotGeomInfo:=", <PlotGeomArray>,
"FilterBoxes:=", <FilterBoxArray>,
<PlotOnPointsSettings>,
<PlotOnLineSettings>,
<PlotOnSurfaceSettings>,
<PlotOnVolumeSettings>)
```

#### SolutionName

Name of the solution setup and solution formatted as:

```
"<SolveSetupName>: <WhichSolution>",
where <WhichSolution> can be "Adaptive_<n>",
"LastAdaptive", or "PortOnly".
```

For example: "Setup1 : Adaptive\_2"

HFSS and Maxwell require a space on either side of the ':' character. If it is missing, the plot will not be created.

#### QuantityName

*Type of plot to create. Possible values include:*

**Mesh plots:** "Mesh"

**Field plots (HFSS) include:**

```
"Mag_E", "Mag_H", "Mag_Jvol", "Mag_Jsurf",
"ComplexMag_E", "ComplexMag_H", "ComplexMag_Jvol",
"ComplexMag_Jsurf", "Vector_E", "Vector_H",
"Vector_Jvol", "Vector_Jsurf", "Vector_RealPoynting",
"Local_SAR", "Average_SAR"
```

**Field plots (Maxwell) include:**

```
"Mag_E", "Mag_H", "E_Vector", "H_Vector", "Mag_J", "J_Vector",
"ComplexMag_J", "Mag_Jc", "Jc_Vector", "ComplexMag_Jc",
"Energy", "coEnergy", "appEnergy", "Ohmic_Loss", "Total_Loss",
""Hysteresis_Loss", "Dielectric_Loss", "Temperature", "Volume_Force_Density", Average_Surface_Loss_Density
"Surface_Force_Density", "Demag_Coef", "QuantityName_FieldLineTrace", "Surface_Loss_Density", QSurf
```

**Field plots (Mechanical) include - (dependent on solution type):**

```
"Mag_Displacement", "Displacement_Vector", "Temperature", "HeatFlux",
"Mag_HeatFlux", "Surface Loss Density", "Volume Loss Density",
"Linked Heat Transfer Coefficient", "Equivalent Stress"
```

PlotFolder

---

*Name of the folder to which the plot should be added. (Values vary with the design type.) Some possible values include:*

"E Field", "H Field", "Jvol", "Jsurf", "SAR Field", "Jc", "Surface-Loss", QSurf, Temperature, Energy, Average-Surface-Loss-Density, "Dielectric\_Loss", "MeshPlots", "Heat Flux", and "Displacement".

#### UserSpecifyName

0 if default name for plot is used, 1 otherwise.

Not needed. <PlotName> will be respected regardless of whether this flag is set.

#### UserSpecifyFolder

0 if default folder for plot is used, 1 otherwise.

Not needed. The specified PlotFolder will be respected regardless of whether this flag is set.

#### IntrinsicVar

Formatted string that specifies the frequency and phase at which to make the plot.

For example: "Freq='1GHz' Phase='30deg'"

#### IntrinsicVar (*Maxwell*)

Formatted string that specifies the skew slice number at which to make the plot.

For example: "IntrinsicVar:=" , "Slice='2' Time='0.00020000000000000001s'"

## PlotGeomInfo

Creating field plot on selected layer-net pairs:

For example, Maxwell: "PlotGeomInfo :=", [1,"Volume","ObjList",2,"LC1\_1:Top","LC1\_1:Top#L1"]

HFSS example with "Plot on surface" option is not checked

```
"PlotGeomInfo:=" , [1,"Volume","LayerNets",2, "Top",2,"net1","net2",_
"Ground",1,"GND"]
```

The command creates field plot on 2 layer-nets combinations. It starts with type "Volume", and subtype "LayerNets", followed by the number of layer-nets combinations which is 2 in this example.

The two layer-nets combinations are (1) "Top" layer, 2 nets, net names are "net1" and "net2", i.e., [Top, net1] pair, and [Top, net2] pair. (2) "Ground" layer and 1 net, net name is "GND", i.e., [Ground, GND] pair.

HFSS example with "Plot on surface" option is checked

```
"PlotGeomInfo:=" , [1,"Surface","LayerNetsExtFace",2,"Top",2,"net1","net2",_
"Ground",1,"GND"],
```

The command creates field plot on 2 layer-nets combinations. It starts with type "Surface", and subtype "LayerNetsExtFace", followed by the number of layer-nets combinations which is 2 in this example.

The two layer-nets combinations are (1) "Top" layer, 2 nets, net names are "net1" and "net2", i.e., [Top, net1] pair, and [Top, net2] pair. (2) "Ground" layer and 1 net, net name is "GND", i.e., [Ground, GND] pair.

Limitation:

No Layer/Nets name is supported for field plot command recording

Should be able to add it based on this change.

No Layer/Nets name support for pure Layout design field plot command.

```
<PlotGeomArray>
    Array(<NumGeomTypes>, <GeomTypeData>,
<GeomTypeData>, ...)

    For example: Array(4, "Volume", "ObjList", 1, "Box1",
"Surface", "FacesList", 1, "12", "Line", 1,
"Polyline1", "Point", 2, "Point1", "Point2")
```

<NumGeomTypes>

Type: **<int>**

Number of different geometry types (volume, surface, line, point) plotted on at the same time.

<GeomTypeData>

```
<GeomType>, <ListType>, <NumIDs>, <ID>, <ID>, ...)
```

<GeomType>

Type: **<string>**

Possible values are "Volume", "Surface", "Line", "Point".

<ListType>

Type: **<string>**

---

Possible values are "ObjList", or "FacesList".

These are used for the GeomType of "Line" or "Point".

<NumIDs>

Type: <int>

Number of IDs or object names that will follow.

<ID>

Type: <int> or <string>

ID of a face or name of an object, line, or point on which to plot.

<FilterBoxArray>

Array of names of objects to use to restrict the plot range.

Array(<NumFilters>, <ObjName>, <ObjName>, ...)

Example: Array(1, "Box1")

Example: Array(0) no filtering

<PlotOnPointSettings>

Array("NAME:PlotOnPointSettings",

"PlotMarker:=", <bool>,

```
"PlotArrow:=", <bool>)
```

```
<PlotOnLineSettings>
    Array("NAME:PlotOnLineSettings",
        Array("NAME:LineSettingsID",
            "Width:=", <int>,
            "Style:=", <string>),
        "IsoValType:=", <string>,
        "ArrowUniform:=", <bool>,
        "NumofArrow:=", <int>)
```

#### Style

*Possible values are "Cylinder", "Solid", "Dashdash",  
"Dotdot", "Dotdash"*

#### IsoValType

*Possible values are "Tone", "Fringe", "Gourard"*

```
<PlotOnSurfaceSettings>
```

```
    Array("NAME:PlotOnSurfaceSettings",
        "Filled:=", <bool>,
```

```
"IsoValType:=", <string>,
"SmoothShade:=", <bool>,
"AddGrid:=", <bool>,
"MapTransparency:=", <bool>,
"Transparency:=", <double>,
"ArrowUniform:=", <bool>
"ArrowSpacing:=", <double>
"GridColor:=", Array(<int>, <int>, <int>)
```

#### IsoValType

*Possible values are:* "Tone", "Line", "Fringe", "Gourard"

#### GridColor

Array containing the R, G, B components of the color. Components  
should be in the range 0 to 255.

#### <PlotOnVolumeSettings>

```
Array("NAME:PlotOnVolumeSettings",
"PlotIsoSurface:=", <bool>,
"CloudDensity:=", <double>,
```

```
"PointSize:=", <int>,
"ArrowUniform:=", <bool>,
"ArrowSpacing:=", <double>)
```

### Example Field Plot:

```
oModule.CreateFieldPlot Array("NAME:Mag_E1", _
"SolutionName:=", "Setup1 : LastAdaptive", _
"QuantityName:=", "Mag_E", _
"PlotFolder:=", "E Field1", _
>UserSpecifyName:=", 0, _
>UserSpecifyFolder:=", 0, _
"IntrinsicVar:=", "Freq='1GHz' Phase='0deg'", _
"PlotGeomInfo:=", Array( 1, "Surface",_
"FacesList", 1, "7"),_
"FilterBoxes:=", Array(0),
Array("NAME:PlotOnSurfaceSettings", _
"Filled:=", false, _
"IsoValType:=", "Fringe", _
"SmoothShade:=", true, _
>AddGrid:=", false, _
"MapTransparency:=", true, _
```

```
"Transparency:=", 0, _  
"ArrowUniform:=", true, _  
"ArrowSpacing:=", 0.100000001490116, _  
"GridColor:=", Array(255, 255, 255)))
```

### Example Demag\_Coef Field Plot:

```
oModule.CreateFieldPlot Array("NAME:Demag_Coef2", "SolutionName:=", _  
"Setup1 : Transient", "UserSpecifyName:=", 0, "UserSpecifyFolder:=", 0, _  
"QuantityName:=", "Demag_Coef", "PlotFolder:=", "Demag-Coef", "StreamlinePlot:=", _  
false, "AdjacentSidePlot:=", false, "FullModelPlot:=", false, "IntrinsicVar:=", _  
"Time=" & Chr(39) & "0s" & Chr(39) & "", "PlotGeomInfo:=", Array( 1, _  
"Volume", "ObjList", 1, "Mag2_0"), "FilterBoxes:=", Array(1, "Mag1_0"), _  
Array("NAME:PlotOnVolumeSettings", "PlotIsoSurface:=", true, "PointSize:=", 1, _  
"Refinement:=", 0, "CloudSpacing:=", 0.5, "CloudMinSpacing:=", -1, _  
"CloudMaxSpacing:=", -1, Array( "NAME:Arrow3DSpacingSettings", "ArrowUniform:=", _  
true, "ArrowSpacing:=", 0, "MinArrowSpacing:=", 0, "MaxArrowSpacing:=", 0)), _  
"EnableGaussianSmoothing:=", false), "Field"
```

*Parameters:* <PlotParameterArray> "VRT"

```
Array("NAME:<PlotName>",<
```

```
"SolutionName:=", <string>,
"QuantityName:=", <string>,
"PlotFolder:=", <string>,
"UserSpecifyName:=", <int>,
"UserSpecifyFolder:=", <int>,
"IntrinsicVar:=", <string>,
"MaxFrequency:=", "<Number><FreqUnits>", _
"LaunchFrom:=", ["Launch from custom", | "LaunchFromPointID:"], ] _
18007, "RayDensity:=", <int>, _
"NumberBounces:=", <int>, _
"Multi-Bounce Ray Density Control:=", <Boolean>, _
"MBRD Max sub divisions:=", <int>, _
"ShootFilterType:=", ["All Rays" | _
"Rays by index", "start index:=", <int>, "stop index:=", <int>, |_
"index step:=", <int> "Rays in box", "FilterBoxID:=", <objID> | _
"Single ray", "Ray elevation:=", "<real>deg", "Ray azimuth:=", "<real>deg"
"),
"VRT"
```

### Example VRT Plot:

```
Dim oAnsoftApp
```

```
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule

Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")

Set oDesktop = oAnsoftApp.GetAppDesktop()

oDesktop.RestoreWindow

Set oProject = oDesktop.SetActiveProject("abrams_1p5")

Set oDesign = oProject.SetActiveDesign("UseSbr")

Set oModule = oDesign.GetModule("FieldsReporter")

oModule.CreateFieldPlot Array("NAME:VRT_Plot1", _
"SolutionName:=", "Setup1 : LastAdaptive", _
"QuantityName:=", "QuantityName_SBR", _
"PlotFolder:=", "Visual Ray Trace SBR", "UserSpecifyName:=", 0, _
"UserSpecifyFolder:=", 0, "IntrinsicVar:=", "", "MaxFrequency:=", "1GHz", _
"LaunchFrom:=", "Launch from custom", "LaunchFromPointID:=", 18007, _
"RayDensity:=", 2, "NumberBounces:=", 5, "Multi-Bounce Ray Density Control:=", _
false, "MBRD Max sub divisions:=", 2, "ShootFilterType:=", "All Rays"), "VRT"
```

**VB Example:**

```
oModule.CreateFieldPlot Array("NAME:Mag_E1", _  
"SolutionName:=", "Setup1 : LastAdaptive", _  
"QuantityName:=", "Mag_E", _  
"PlotFolder:=", "E Field1", _  
"UserSpecifyName:=", 0, _  
"UserSpecifyFolder:=", 0, _  
"IntrinsicVar:=", "Freq='1GHz' Phase='0deg'", _  
"PlotGeomInfo:=", Array( 1, "Surface", _  
    "FacesList", 1, "7"), _  
"FilterBoxes:=", Array(0),  
Array("NAME:PlotOnSurfaceSettings", _  
    "Filled:=", false, _ "IsoValType:=", "Fringe", _  
    "SmoothShade:=", true, _  
    "AddGrid:=", false, _  
    "MapTransparency:=", true, _  
    "Transparency:=", 0, _  
    "ArrowUniform:=", true, _  
    "ArrowSpacing:=", 0.10000001490116, _  
    "GridColor:=", Array(255, 255, 255))), "Field"
```

## Example Electron Density Plot

```
oModule = oDesign.GetModule("FieldsReporter")
oModule.CreateFieldPlot(
    [
        "NAME:Electron_Density3",
        "SolutionName:=" , "AIR : RFDischarge",
        "UserSpecifyName:=" , 0,
        "UserSpecifyFolder:=" , 0,
        "QuantityName:=" , "Electron_Density",
        "PlotFolder:=" , "RF Discharge Fields",
        "StreamlinePlot:=" , False,
        "AdjacentSidePlot:=" , False,
        "FullModelPlot:=" , False,
        "IntrinsicVar:=" , "Freq=\'0.2000000000000001GHz\' GasPressure=\'0.02kPascal\'",
        "PlotGeomInfo:=" , [1,"Surface","FacesList",1,"48"],
        "FilterBoxes:=" , [0],
        [
            "NAME:PlotOnSurfaceSettings",
            "Filled:=" , False,
            "IsoValType:=" , "Tone",
```

```
"AddGrid:=" , False,
"MapTransparency:=" , True,
"Refinement:=" , 0,
"Transparency:=" , 0,
"SmoothingLevel:=" , 0,
"ShadingType:=" , 0,
[
    "NAME:Arrow3DSpacingSettings",
        "ArrowUniform:=" , True,
        "ArrowSpacing:=" , 0,
        "MinArrowSpacing:=" , 0,
        "MaxArrowSpacing:=" , 0
    ],
    "GridColor:=" , [255,255,255]
],
"EnableGaussianSmoothing:=" , False,
"SurfaceOnly:=" , False
], "Field")
```

## Example Mesh Plot

```
Dim oAnsoftApp
```

```
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule

Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow

Set oProject = oDesktop.SetActiveProject("abrams_1p5")
Set oDesign = oProject.SetActiveDesign("UseSbr")
Set oModule = oDesign.GetModule("FieldsReporter")

oModule.CreateFieldPlot Array("NAME:Mesh1", "SolutionName:=", "Setup1 : LastAdaptive", _
"QuantityName:=", "Mesh", "PlotFolder:=", "MeshPlots", "FieldType:=", "Fields", _
>UserSpecifyName:=", 0, "UserSpecifyFolder:=", 0, "StreamlinePlot:=", false, _
"IntrinsicVar:=", "Freq=" & Chr(39) & "2GHz" & Chr(39) & " Phase=" _
& Chr(39) & "0deg" & Chr(39) & "" & "", "PlotGeomInfo:=", Array(1, _
"Surface", "FacesList", 1, "10568"), "FilterBoxes:=", Array(0), "Real time mode:=", _
true, Array("NAME:MeshSettings", "Scale factor:=", 100, "Transparency:=", 0, _
"Mesh type:=", "Shaded", "Surface only:=", true, "Add grid:=", true, "Refinement:=", _
0, "Use geometry color:=", true, "Mesh line color:=", Array(0, 0, 255), _
```

```
"Filled color:=", Array( 255, 255, 255))), "Field"
```

### VB Example: Maxwell Plot with Skew Slice

```
oModule.CreateFieldPlot Array("NAME:energy2", "SolutionName:=", "Setup1 : Transient", "User-SpecifyName:=", 0, "UserSpecifyFolder:=", 0, "QuantityName:=", "energy", "PlotFolder:=", "Energy", "StreamlinePlot:=", false, "AdjacentSidePlot:=", false, "FullModelPlot:=", false, "IntrinsicVar:=", "Slice=" & Chr(39) & "2" & Chr(39) & " Time=" & Chr(39) & "0.00020000000000000001s" & Chr(39) & "",...)
```

### Python Example: Maxwell Plot with Skew Slice

```
oModule.CreateFieldPlot(  
[  
    "NAME:Mesh1",  
    "SolutionName:=", "Setup1 : Transient",  
    "UserSpecifyName:=", 0,  
    "UserSpecifyFolder:=", 0,  
    "QuantityName:=" "Mesh",  
    "PlotFolder:=", "MeshPlots",  
    "FieldType:=", "Fields",  
    "StreamlinePlot:=" , False,  
    "AdjacentSidePlot:=" , False,  
    "FullModelPlot:=", False,  
    "IntrinsicVar:=", "Slice='2' Time='0.00020000000000000001s'",
```

```
"PlotGeomInfo:=", [1,"Surface","FacesList",1,"2955"],  
"FilterBoxes:=", [0],  
...]
```

### Python Example: Maxwell Average\_Surface\_Loss\_Density Plot

```
oModule.CreateFieldPlot(  
[  
    "NAME:Average_Surface_Loss_Density1",  
    "SolutionName:=" , "Setup1 : Transient",  
    "UserSpecifyName:=" , 0,  
    "UserSpecifyFolder:=" , 0,  
    "QuantityName:=" , "Average_Surface_Loss_Density",  
    "PlotFolder:=" , "Average-Surface-Loss-Density",  
    "StreamlinePlot:=" , False,  
    "AdjacentSidePlot:=" , False,  
    "FullModelPlot:=" , False,  
    "IntrinsicVar:=" , "Time=\\"4us\\" Time0=\\"0s\\\"",  
    "PlotGeomInfo:=", [1,"Surface","FacesList",1,"6"],  
    ...  
])
```

## Python Examples: Maxwell ACConduction Design Plot

```
oModule.CreateFieldPlot()  
[  
    "NAME:Mag_Jc1",  
    "SolutionName:=", "Setup1 : LastAdaptive",  
    "UserSpecifyName:=", 0,  
    "UserSpecifyFolder:=", 0,  
    "QuantityName:=", "Mag_Jc",  
    "PlotFolder:=", "Jc",  
    "StreamlinePlot:=", False,  
    "AdjacentSidePlot:=", False,  
    "FullModelPlot:=", False,  
    "IntrinsicVar:=", "Freq=\\"1000000Hz\\" Phase=\\"0deg\\\"",  
    "PlotGeomInfo:=", [1,"Volume","ObjList",1,"Box1"],  
    "FilterBoxes:=", [1,""],  
    [  
        "NAME:PlotOnVolumeSettings",  
        "PlotIsoSurface:=", True,  
        "PointSize:=", 1,  
        "Refinement:=", 0,
```

```
"CloudSpacing:=", 0.5,
"CloudMinSpacing:=", -1,
"CloudMaxSpacing:=", -1,
"ShadingType:=", 0,
[
    "NAME:Arrow3DSpacingSettings",
    "ArrowUniform:=", True,
    "ArrowSpacing:=", 0,
    "MinArrowSpacing:=", 0,
    "MaxArrowSpacing:=", 0
]
],
"EnableGaussianSmoothing:=", False
], "Field")

oModule.CreateFieldPlot(
[
    "NAME:ComplexMag_Jc1",
    "SolutionName:=", "Setup1 : LastAdaptive",
    "UserSpecifyName:=", 0,
```

```
"UserSpecifyFolder:=", 0,
"QuantityName:=", "ComplexMag_Jc",
"PlotFolder:=", "Jc",
...)

oModule.CreateFieldPlot(
[
  "NAME:Jc_Vector2",
  "SolutionName:=", "Setup1 : LastAdaptive",
  "UserSpecifyName:=", 0,
  "UserSpecifyFolder:=", 0,
  "QuantityName:=", "Jc_Vector",
  "PlotFolder:=", "Jc",
]
...)

oModule.CreateFieldPlot(
[
  "NAME:Dielectric_Loss1",
  "SolutionName:=", "Setup1 : LastAdaptive",
  "UserSpecifyName:=", 0,
```

```
"UserSpecifyFolder:=", 0,  
"QuantityName:=", "Dielectric_Loss",  
"PlotFolder:=", "Dielectric-Loss",  
]  
...)
```

## Maxwell Field Line Trace Plot Examples

### VB Example: Maxwell Field Line Trace Plot

```
oModule.CreateFieldPlot  
Array("NAME:FieldLineTrace_Plot3",  
"SolutionName:=", "Setup1 : LastAdaptive",  
"UserSpecifyName:=", 0,  
"UserSpecifyFolder:=", 0,  
"QuantityName:=", "QuantityName_FieldLineTrace",  
"PlotFolder:=", "Field line trace plot",  
"IntrinsicVar:=", "",  
"Trace Step Length:=", "0.001mm",  
"Use Adaptive Step:=", true,  
"Seeding Faces:=", Array( 1, 15091),  
"Seeding Markers:=", Array(0),  
"Surface Tracing Objects:=", Array(1, 15),  
"Volume Tracing Objects:=", Array(1, 36),  
"Seeding Sampling Option:=", true,  
"Seeding Points Number:=", 15,  
"Fractional of Maximal:=", 0.8,  
"Discrete Seeds Option:=", "Marker Point",  
Array("NAME:InceptionEvaluationSettings",
```

```
"Gas Type:=", 0, "Gas Pressure:=", 1),
Array("NAME:FieldLineTracePlotSettings",
Array("NAME:LineSettingsID",
"Width:=", 1,
"Style:=", "Solid"),
"IsoValType:=", "Tone"))
"FieldLineTrace")

oModule.ModifyInceptionParameters "FieldLineTrace_Plot3",
Array("NAME:InceptionEvaluationSettings",
"Gas Type:=", 2, "Gas Pressure:=", 1,
"Use Inception:=", True,
"Potential U0:=", 0,
"Potential K:=", 1,
"Potential A:=", 1
"Critical Value:=", 5.9426,
"Streamer Constant:=", 6.5,
"Ionization Coefficient Dataset:=", Array( 0))
```

### Python Example: Maxwell Field Line Trace Plot

```
oModule.CreateFieldPlot(
[
    "NAME:FieldLineTrace_Plot3",
    "SolutionName:=", "Setup1 : LastAdaptive",
    "UserSpecifyName:=", 0,
    "UserSpecifyFolder:=", 0,
    "QuantityName:=", "QuantityName_FieldLineTrace",
    "PlotFolder:=", "Field line trace plot",
```

```
"IntrinsicVar:="", "",  
"Trace Step Length:=", "0.001mm",  
"Use Adaptive Step:=", True,  
"Seeding Faces:=", [1,15091],  
"Seeding Markers:=", [0],  
"Surface Tracing Objects:=", [1,15],  
"Volume Tracing Objects:=", [1,36],  
"Seeding Sampling Option:=", True,  
"Seeding Points Number:=", 15,  
"Fractional of Maximal:=", 0.8,  
"Discrete Seeds Option:=", "Marker Point",  
[  
    "NAME:InceptionEvaluationSettings",  
    "Gas Type:=", 0,  
    "Gas Pressure:=", 1  
    "Use Inception:=", True,  
    "Potential U0:=", 0,  
    "Potential K:=", 1,  
    "Potential A:=", 1  
],
```

```
[  
    "NAME:FieldLineTracePlotSettings",  
    [  
        "NAME:LineSettingsID",  
        "Width:=", 1,  
        "Style:=", "Solid"  
    ],  
    "IsoValType:=", "Tone"  
], "FieldLineTrace")
```

### For Q3D Extractor and 2D Extractor, the command details are as follows:

*Use:* Creates a field/mesh plot.

*Command:* **Q3D Extractor or 2D Extractor>Fields**

*Syntax:* CreateFieldPlot <PlotParameterArray>

*Return Value:* None

*Parameters:* <PlotParameterArray>

```
Array ("NAME:<PlotName>",  
      "SolutionName:=", <string>,  
      "QuantityName:=", <string>,
```

```
"PlotFolder:=", <string>,
"UserSpecifyName:=", <int>,
"UserSpecifyFolder:=", <int>,
"IntrinsicVar:=", <string>,
"PlotGeomInfo:=", <PlotGeomArray>,
"FilterBoxes:=", <FilterBoxArray>,
<PlotOnPointSettings>, <PlotOnLineSettings>,
<PlotOnSurfaceSettings>, <PlotOnVolumeSettings>)
```

#### SolutionName

Name of the solution setup and solution formatted as:

```
"<SolveSetupName> : <WhichSolution>"
```

where <WhichSolution> can be "Adaptive\_<n>",

"LastAdaptive", or "PortOnly".

For example: "Setup1 : Adaptive\_2"

HFSS requires a space on both sides of the ':' character. Otherwise, the plot is not created.

#### QuantityName

Type of plot to create. Possible values are:

Mesh plots: "Mesh"

Q3D Field Plots:

Field type	Plot quantity names
AC R/L Fields	"SurfaceJac", "Mag_SurfaceJac"
DC R/L PEC Fields	"SurfaceJdc", "Mag_SurfaceJdc"
DC R/L Fields	"VolumeJdc", "Mag_VolumeJdc", "Phidc"
C Fields	"SmoothQ", "ABS_Q"

2D Extractor Field Plots:

Field type	Plot quantity names
CG Fields	"Mag_Phi", "PhiAtPhase", "Mag_E", "VectorE", "Mag_Jcg", "VectorJcg" and "energyCG"
RL Fields	"Flux Lines", "VectorA", "Mag_B", "VectorB", "Mag_H", "VectorH", "Jrl", "VectorJrl", "energyRL", "coenergy", "appenergy" and "emloss"

PlotFolder

Name of the folder to which the plot should be added. Possible values are: "Q", "ABS\_Q", "JDC Vol", "Phi", "JDC Surf", and "JAC".

UserSpecifyName

0 if default name for plot is used, 1 otherwise.

This parameter is not essential. <PlotName> is respected regardless of whether this flag is set.

UserSpecifyFolder

0 if the default folder for plot is used, 1 otherwise.

This parameter is not essential. The specified PlotFolder is respected regardless of whether this flag is set.

### IntrinsicVar

Formatted string that specifies the frequency and phase at which to create the plot.

For example: "Freq='1GHz' Phase='30deg'"

### <PlotGeomArray>

*Array(<NumGeomTypes>, <GeomTypeData>,*

*<GeomTypeData>, ...)*

For example: *Array(4, "Volume", "ObjList", 1, "Box1",*

*"Surface", "FacesList", 1, "12", "Line", 1,*

*"Polyline1", "Point", 2, "Point1", "Point2")*

### <NumGeomTypes>

Type: <int>

Number of different geometry types (volume, surface, line, point) plotted at the same time.

### <GeomTypeData>

*<GeomType>, <ListType>, <NumIDs>, <ID>, <ID>, ...)*

### <GeomType>

Type: <string>

Possible values are "Volume", "Surface", "Line", "Point".

### <ListType>

Type: <string>

Possible values are "ObjList" or "FacesList".

These are used for GeomType values "Line" or "Point"

<NumIDs>

Type: <int>

Number of IDs or object names that will follow.

<ID>

Type: <int> or <string>

*ID of a face or name of an object, line, or point on which to plot.*

<FilterBoxArray>

Array of object names used to restrict the plot range.

Array(<NumFilters>, <ObjName>, <ObjName>, ...)

Example: Array(1, "Box1")

Example: Array(0) no filtering

<PlotOnPointSettings>

Array("NAME:PlotOnPointSettings",

"PlotMarker:=", <bool>,

"PlotArrow:=", <bool>)

<PlotOnLineSettings>

Array("NAME:PlotOnLineSettings",

Array("NAME:LineSettingsID",

"Width:=", <int>,

```
"Style:=", <string>),
"IsoValType:=", <string>,
"ArrowUniform:=", <bool>,
"NumofArrow:=", <int>

Style
    Possible values are "Cylinder", "Solid", "Dashdash",
    "Dotdot", "Dotdash".

IsoValType
    Possible values are "Tone", "Fringe", "Gourard".

<PlotOnSurfaceSettings>
    Array("NAME:PlotOnSurfaceSettings",
        "Filled:=", <bool>,
        "IsoValType:=", <string>,
        "SmoothShade:=", <bool>,
        "AddGrid:=", <bool>,
        "MapTransparency:=", <bool>,
        "Transparency:=", <double>,
        "ArrowUniform:=", <bool>
        "ArrowSpacing:=", <double>
        "GridColor:=", Array(<int>, <int>, <int>)
```

**IsoValType**

Possible values are: "Tone", "Line", "Fringe", "Gourard".

**GridColor**

Array containing the R, G, B components of the color. Components should be in the range 0 to 255.

## &lt;PlotOnVolumeSettings&gt;

```
Array("NAME:PlotOnVolumeSettings",
"PlotIsoSurface:=", <bool>,
"CloudDensity:=", <double>,
"PointSize:=", <int>,
"ArrowUniform:=", <bool>,
"ArrowSpacing:=", <double>)
```

**DeleteFieldPlot**

Deletes one or more field plots.

<b>UI Access</b>	Right-click on one filed plot, select <b>Delete</b> .		
<b>Parameters</b>	Name <NameArray>	Type Array	Description Array of strings containing the names of the plots to delete.
<b>Return Value</b>	None.		

**Python Syntax**

```
DeleteFieldPlot(<NameArray>)
```

**Python Example**

```
oModule.DeleteFieldPlot(["Mag_E1", "Vector_E1"])
```

**VB Syntax**

```
DeleteFieldPlot <NameArray>
```

**VB Example**

```
oModule.DeleteFieldPlot Array("Mag_E1", "Vector_E1")
```

## EditSurfaceMeshSummaryData

*Use:* Defines or Edits Surface Mesh Summary Data.

*Command:* Create Surface Mesh Summary, **Setup...**

*Syntax:* EditSurfaceMeshSummaryData Array(Array(<Parameters Array>))

*Return Value:* None

*Parameters:*

"NAME:SurfaceMeshSummary"

Type: <string>

"NAME:SurfaceMeshSummary"

"SolutionName:=" <string>

Type: <string>

Solution Name.

"Variation Name:=" <string>

Type: <string>

Variation name.

```
<RowItemsArray>
    Array<MeshRowItems>
        Row data.

<ItemPerRow>
    Array<EntityPerRow>
        Entity ID.

<MeshDataPer Item>
    Array<Mesh Data Category and Stats>
```

*VB Example:*

```
Set oModule = oDesign.GetModule("FieldsReporter")
oModule.EditSurfaceMeshSummaryData Array(Array("NAME:SurfaceMeshSummary", "SolutionName:=", _
"Setup1 : LastAdaptive", "Variation:=", "Nominal", Array("NAME:MeshRowItems", Array
("NAME:ItemPerRow", "EntityPerRow:=", _
"annular_rng", Array("NAME:MeshDataPerItem", "Min Edge Len:=", 0.000166961133900917, "Max Edge
Len:=", _
0.00145730991671888, "Mean Edge Len:=", 0.000524160923260581, "Std Devn Edge Len:=", _
0.000217758706109324, "Min Tri Area:=", 3.4982570283924E-09, "Max Tri Area:=", _
3.73116346661249E-07, "Mean Tri Area:=", 8.91526236529066E-08, "Std Devn Tri Area:=", _
6.89482807990471E-08, "DataState:=", 2)), Array("NAME:ItemPerRow", "EntityPerRow:=", _
"gap", Array("NAME:MeshDataPerItem", "Min Edge Len:=", 0.000105681286042456, "Max Edge Len:=", _
0.000362084671538928, "Mean Edge Len:=", 0.000233090954707978, "Std Devn Edge Len:=", _
```

```
7.55387369793154E-05, "Min Tri Area:=", 1.74640378345369E-09, "Max Tri Area:=", _  
3.78166789228631E-08, "Mean Tri Area:=", 1.56497981245286E-08, "Std Devn Tri Area:=", _  
1.04691637802914E-08, "DataState:=", 2))))
```

Python Syntax	RenameReport (<OldReportName>, <NewReportName>)
Python Example	<pre>oModule.EditSurfaceMeshSummaryData(     [         [             "NAME:SurfaceMeshSummary",             "SolutionName:=" , "Setup1 : LastAdaptive",             "Variation:=" , "Nominal",             [                 "NAME:MeshRowItems",                 [                     "NAME:ItemPerRow",                     "EntityPerRow:=" , "annular_rng",                     [                         "NAME:MeshDataPerItem",                         "Min Edge Len:=" , 0.000166961133900917,                         "Max Edge Len:=" , 0.00145730991671888,</pre>

```
        "Mean Edge Len:="      , 0.000524160923260581,
        "Std Devn Edge Len:=" , 0.000217758706109324,
        "Min Tri Area:="     , 3.4982570283924E-09,
        "Max Tri Area:="     , 3.73116346661249E-07,
        "Mean Tri Area:="    , 8.91526236529066E-08,
        "Std Devn Tri Area:=" , 6.89482807990471E-08,
        "DataState:="         , 2
    ]
],
[
    "NAME:ItemPerRow",
    "EntityPerRow:="      , "gap",
    [
        "NAME:MeshDataPerItem",
        "Min Edge Len:="      , 0.000105681286042456,
        "Max Edge Len:="      , 0.000362084671538928,
        "Mean Edge Len:="     , 0.000233090954707978,
        "Std Devn Edge Len:=" , 7.55387369793154E-05,
        "Min Tri Area:="     , 1.74640378345369E-09,
        "Max Tri Area:="     , 3.78166789228631E-08,
        "Mean Tri Area:="    , 1.56497981245286E-08,
```

```

        "Std Devn Tri Area:=" , 1.04691637802914E-08,
        "DataState:=" , 2
    ]
]
]
]
]
)

```

## ExportPlotImageWithViewToFile [Reporter]

Exports a field plot image to a specified file.

**Note:**

This script replaces ExportPlotImageToFile, which is deprecated.

UI Access	N/A		
Parameters	Name	Type	Description
	<FileName>	String	Full path of file to export.
	<PlotQuantityName>	String	Name of quantity to plot.
	<PlotItemName>	String	Name of fields to plot.
	<PixelSizeX>	Integer	Desired image width, in pixels.
	<PixelSizeY>	Integer	Desired image height, in pixels.
	<ViewOrientation>	String	<i>Optional.</i> Name of orientation to use for plot.
Return Value	Image file is exported to the specified path.		

<b>Python Syntax</b>	ExportPlotImageWithViewToFile(<FileName>, <PlotQuantityName>, <PlotItemName>, <PixelSizeX>, <PixelSizeY>, <ViewOrientation>)
<b>Python Example</b>	<pre>oModule.ExportPlotImageWithViewToFile("E:/MyDir/OptimToutput/magE2.gif", "E Field", "Mag_E2", 1920, 1080, "newot2")</pre>

<b>VB Syntax</b>	ExportPlotImageWithViewToFile <FileName>, <PlotQuantityName>, <PlotItemName>, <PixelSizeX>, <PixelSizeY>, <ViewOrientation>
<b>VB Example</b>	<pre>oModule.ExportPlotImageWithViewToFile "E:/MyDir/OptimToutput/magE2.gif", "E Field", _ "Mag_E2", 1920, 1080, "newot2"</pre>

## ExportSurfaceMeshSummary

*Use:* Exports a Surface Mesh Summary.

**Command: Create Surface Mesh Summary.**

*Syntax:* ExportSurfaceMeshSummary Array(<SolutionName>, <DesignVariationKey> <ExportFileName>)

*Return Value:* None

*Parameters:* <SolutionName>

Type: <string>

*Original name of the plot.*

<DesignVariationKey>

Type: <string>

New name of the plot.

*VB Example:*

```
oModule.ExportSurfaceMeshSummary Array("SolutionName:=", "Setup1 : LastAdaptive", "DesignVariationKey:=", _  
"Nominal", "ExportFileName:=", "D:\documents\surfaceMeshSummaryReport.csv")
```

Python Syntax	ExportSurfaceMeshSummary (<SolutionName>, <DesignVariationKey>, <ExportFileName>)
<b>Python Example</b>	<pre>oModule.ExportSurfaceMeshSummary (     [         "SolutionName:=" , "Setup1 : LastAdaptive",         "DesignVariationKey:=" , "Nominal",         "ExportFileName:=" , "D:\\Program Files\\Documents\\surfaceMeshSummaryReport.csv"     ])</pre>

## GetFieldPlotNames

Gets the names of field overlay plots defined in a design.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array of strings containing field plots names.

<b>Python Syntax</b>	GetFieldPlotNames()
<b>Python Example</b>	<code>oModule.GetFieldPlotNames ()</code>

<b>VB Syntax</b>	GetFieldPlotNames
<b>VB Example</b>	<code>oModule.GetFieldPlotNames</code>

## ModifyFieldPlot

Modifies a plot definition.

<b>UI Access</b>	[product] > Fields > Modify Plot									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>&lt;OriginalName&gt;</i></td> <td>String</td> <td>Name of the plot to be modified.</td> </tr> <tr> <td><i>&lt;PlotParams&gt;</i></td> <td>Array</td> <td>Array containing modified settings.</td> </tr> </tbody> </table>	Name	Type	Description	<i>&lt;OriginalName&gt;</i>	String	Name of the plot to be modified.	<i>&lt;PlotParams&gt;</i>	Array	Array containing modified settings.
Name	Type	Description								
<i>&lt;OriginalName&gt;</i>	String	Name of the plot to be modified.								
<i>&lt;PlotParams&gt;</i>	Array	Array containing modified settings.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	ModifyFieldPlot(<OriginalName>, <PlotParams>)
<b>Python Example</b>	<pre>oModule.ModifyFieldPlot("Vector_E1" , ["NAME:Vector_E2",  "SolutionName:=", "Setup1 : LastAdaptive",  "QuantityName:=", "Vector_E",</pre>

```
"PlotFolder:=", "E Field1",
"UserSpecifyName:=", 0,
"UserSpecifyFolder:=", 0,
"IntrinsicVar:=", "Freq='1GHz' Phase='30deg'",
"PlotGeomInfo:=", [1, "Surface", "FacesList", 1, "7"],
"FilterBoxes:=", [0],
["NAME:PlotOnSurfaceSettings",
"Filled:=", False,
"IsoValType:=", "Fringe",
"SmoothShade:=", True,
"AddGrid:=", False,
"MapTransparency:=", True,
"Transparency:=", 0, _
"ArrowUniform:=", True,
"ArrowSpacing:=", 0.100000001490116,
"GridColor:=", [255, 255, 255]]
])
```

---

**VB Syntax**

```
ModifyFieldPlot <OriginalName>, <PlotParams>
```

**VB Example**

```
oModule.ModifyFieldPlot "Vector_E1" ,_
    Array("NAME:Vector_E2", _
        "SolutionName:=", "Setup1 : LastAdaptive", _
        "QuantityName:=", "Vector_E", _
        "PlotFolder:=", "E Field1", _
        "UserSpecifyName:=", 0, _
        "UserSpecifyFolder:=", 0, _
        "IntrinsicVar:=","Freq='1GHz' Phase='30deg'", _
        "PlotGeomInfo:=", Array(1, "Surface","FacesList", 1, "7"), _
        "FilterBoxes:=", Array(0), _
        Array("NAME:PlotOnSurfaceSettings", _
            "Filled:=", false, _
            "IsoValType:=", "Fringe", _
            "SmoothShade:=", true, _
            "AddGrid:=", false, _
            "MapTransparency:=", true, _
            "Transparency:=", 0, _
            "ArrowUniform:=", true, _
            "ArrowSpacing:=", 0.100000001490116, _
            "GridColor:=", Array(255, 255, 255)))
```

## ModifyInceptionParameters

**Use:** For field line trace plots, allows you to set inception voltage evaluation parameters.

UI Access	Right-click on a <b>Field Line Trace Plot</b> in the project tree and select <b>Inception Voltage Evaluation</b>		
<b>Parameters</b>	Name	Type	Description
	<NAME>	string	The name of the inception value settings
	<Gas Type>	int	0 for Dry Air, 1 for SF6, 2 for User Defined
	<Gas Pressure>	int	Gas pressure in Bar
	<Use Inception>	bool	<b>True</b> to use the inception parameters U0, K, A
	<Potential U0>	int	U0 parameter
	<Potential K>	int	Streamer constant (empirical value)
<Potential A>	int	A parameter	
<b>Return Value</b>	None		

<b>Python Syntax</b>	ModifyInceptionParameters (<NAME>, <Gas Type>, <Gas Pressure>, <Use Inception>, <Potential U0>, <Potential K>, <Potential A>)
<b>Python Example</b>	<pre> oModule.ModifyInceptionParameters("FieldLineTrace_Plot1", [     "NAME:InceptionEvaluationSettings",     "Gas Type:=" , 1,     "Gas Pressure:=" , 1,     "Use Inception:=" , True,     "Potential U0:=" , 0,     "Potential K:=" , 1,     "Potential A:=" , 1 ]) </pre>

<b>VB Syntax</b>	ModifyInceptionParameters (<NAME>, <Gas Type>, <Gas Pressure>, <Use Inception>, <Potential U0>, <Potential K>, <Potential A>)
<b>VB Example</b>	<pre> oModule.ModifyInceptionParameters "FieldLineTrace_Plot1", Array("NAME:InceptionEvaluationSettings", "Gas Type:=", 1, "Gas Pressure:=", 1, "Use Inception:=", false, "Potential U0:=", 0, "Potential K:=", 1, "Potential A:=", 1) </pre>

## RenameFieldPlot

Renames a plot.

<b>UI Access</b>	Right-click the plot you want to rename in the project tree, and then click <b>Rename</b> on the shortcut menu.									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;OldName&gt;</td> <td>String</td> <td>Original name of the plot.</td> </tr> <tr> <td>&lt;NewName&gt;</td> <td>String</td> <td>New name of the plot.</td> </tr> </tbody> </table>	Name	Type	Description	<OldName>	String	Original name of the plot.	<NewName>	String	New name of the plot.
Name	Type	Description								
<OldName>	String	Original name of the plot.								
<NewName>	String	New name of the plot.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	RenameFieldPlot(<OldName>, <NewName>)
<b>Python Example</b>	<pre> oModule.RenameFieldPlot( "Vector_E1", "Vector_E2") </pre>

<b>VB Syntax</b>	RenameFieldPlot <OldName>, <NewName>
<b>VB Example</b>	<pre> oModule.RenameFieldPlot _ </pre>

	"Vector_E1", "Vector_E2"
--	--------------------------

## RenamePlotFolder

Renames a plot folder.

<b>UI Access</b>	Right-click a plot folder in the project tree, and then click <b>Rename</b> on the shortcut menu.											
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;OldName&gt;</td><td>String</td><td>Original name of the folder.</td></tr><tr><td>&lt;NewName&gt;</td><td>String</td><td>New name of the folder.</td></tr></table>			Name	Type	Description	<OldName>	String	Original name of the folder.	<NewName>	String	New name of the folder.
Name	Type	Description										
<OldName>	String	Original name of the folder.										
<NewName>	String	New name of the folder.										
<b>Return Value</b>	None.											

<b>Python Syntax</b>	RenamePlotFolder(<OldName>, <NewName>)
<b>Python Example</b>	<pre>oModule.RenamePlotFolder(     "E Field", "Surface Plots")</pre>

<b>VB Syntax</b>	RenamePlotFolder <OldName>, <NewName>
<b>VB Example</b>	<pre>oModule.RenamePlotFolder _     "E Field", "Surface Plots"</pre>

## SetFieldPlotSettings

Sets plot attributes.

<b>UI Access</b>	<b>[product] &gt; Fields &gt; Modify Plot Attributes</b>											
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td><i>&lt;PlotName&gt;</i></td> <td>String</td> <td>Name of the plot to modify.</td> </tr> <tr> <td><i>&lt;PlotItemAttributes&gt;</i></td> <td>Array</td> <td> <p>Structured array.</p> <pre>Array ("NAME:FieldsPlotItemSettings",       &lt;PlotOnPointsSettings&gt;,       &lt;PlotOnLineSettings&gt;,       &lt;PlotOnSurfaceSettings&gt;,       &lt;PlotOnVolumeSettings&gt;)</pre> <p><i>See description of CreateFieldPlot command for details.</i></p> </td> </tr> </table>	Name	Type	Description	<i>&lt;PlotName&gt;</i>	String	Name of the plot to modify.	<i>&lt;PlotItemAttributes&gt;</i>	Array	<p>Structured array.</p> <pre>Array ("NAME:FieldsPlotItemSettings",       &lt;PlotOnPointsSettings&gt;,       &lt;PlotOnLineSettings&gt;,       &lt;PlotOnSurfaceSettings&gt;,       &lt;PlotOnVolumeSettings&gt;)</pre> <p><i>See description of CreateFieldPlot command for details.</i></p>		
Name	Type	Description										
<i>&lt;PlotName&gt;</i>	String	Name of the plot to modify.										
<i>&lt;PlotItemAttributes&gt;</i>	Array	<p>Structured array.</p> <pre>Array ("NAME:FieldsPlotItemSettings",       &lt;PlotOnPointsSettings&gt;,       &lt;PlotOnLineSettings&gt;,       &lt;PlotOnSurfaceSettings&gt;,       &lt;PlotOnVolumeSettings&gt;)</pre> <p><i>See description of CreateFieldPlot command for details.</i></p>										
<b>Return Value</b>	None.											

<b>Python Syntax</b>	<code>SetFieldPlotSettings(&lt;PlotName&gt; &lt;PlotItemAttributes&gt;)</code>
<b>Python Example</b>	<pre>oModule.SetFieldPlotSettings ("Mag_E2",    ["NAME:FieldsPlotItemSettings",     ["NAME:PlotOnLineSettings",      ["NAME:LineSettingsID",       "Width:=", 4,       "Style:=", "Cylinder"),       "IsoValType:=", "Tone",       "ArrowUniform:=", True,</pre>

```
"NumofArrow:=", 100),  
[ "NAME:PlotOnSurfaceSettings",  
  "Filled:=", False,  
  "IsoValType:=", "Tone",  
  "SmoothShade:=", True,  
  "AddGrid:=", False,  
  "MapTransparency:=", True,  
  "Transparency:=", 0,  
  "ArrowUniform:=", True,  
  "ArrowSpacing:=", 0.100000001490116,  
  "GridColor:=", [255, 255, 255]]]  
)
```

<b>VB Syntax</b>	SetFieldPlotSettings <PlotName> <PlotItemAttributes>
<b>VB Example</b>	<pre>oModule.SetFieldPlotSettings "Mag_E2", _     Array("NAME:FieldsPlotItemSettings", _         Array("NAME:PlotOnLineSettings", _             Array("NAME:LineSettingsID", _                 "Width:=", 4, _</pre>

```

"Style:=", "Cylinder"),
"IsoValType:=", "Tone",
"ArrowUniform:=", true,
"NumofArrow:=", 100),
Array("NAME:PlotOnSurfaceSettings",
"Filled:=", false,
"IsoValType:=", "Tone",
"SmoothShade:=", true,
"AddGrid:=", false,
"MapTransparency:=", true,
"Transparency:=", 0,
"ArrowUniform:=", true,
"ArrowSpacing:=", 0.100000001490116,
"GridColor:=", Array(255, 255, 255)))

```

## SetPlotFolderSettings

Sets the attributes of all plots in the specified folder.

UI Access	[product] > Fields > Modify Plot Attributes		
Parameters	Name <PlotFolderName>	Type String	Description Name of the folder with the attributes to modify.
	<PlotFolderAttributes>	Array	Structured array.  Array("NAME:FieldsPlotSettings",

			<pre>"Real time mode:=", &lt;bool&gt;, &lt;ColorMapSettings&gt;, &lt;Scale3DSettings&gt;, &lt;Marker3DSettings&gt;, &lt;Arrow3DSettings&gt;)</pre>
<i>&lt;ColorMapSettings&gt;</i>	Array	Structured array.	<pre>Array("NAME:ColorMapSettings", "ColorMapType:=", &lt;string Possible values are "Uniform", "Ramp", "Spectrum", "SpectrumType:=", &lt;string Possible values are "Rainbow", "Temperature", "Magenta", "Gray", "UniformColor:=", &lt;array containing the R, G, B components of the color. Components should be in the range 0 to 255.&gt;, "RampColor:=", &lt;array containing the R, G, B com- ponents of the color. Components should be in the range 0 to 255.&gt;)</pre>
<i>&lt;Scale3DSettings&gt;</i>	Array	Structured array.	<pre>Array("NAME:Scale3DSettings", "m_nLevels:=", &lt;int&gt;, "m_autoScale:=", &lt;bool&gt;, "minvalue:=", &lt;double&gt;, "maxvalue:=", &lt;double&gt;,</pre>

		<pre>"log:=", &lt;bool&gt;, "IntrinsicMin:=", &lt;double&gt;, "IntrinsicMax:=", &lt;double&gt;)</pre>
<i>&lt;Marker3DSettings&gt;</i>	Array	<p>Structured array.</p> <pre>Array("NAME:Marker3DSettings", "MarkerType:=", &lt;integer 9:Sphere 10: Box 11: Tetrahedron 12: Octahedron default: Sphere&gt;, "MarkerMapSize:=", &lt;bool&gt;, "MarkerMapColor:=", &lt;bool&gt;, "MarkerSize:=", &lt;double&gt;)</pre>
<i>&lt;Arrow3DSettings&gt;</i>	Array	<p>Structured array.</p> <pre>Array("NAME:Arrow3DSettings", "ArrowType:=", &lt;integer 0: Line 1: Cylinder 2: Umbrella default: Line&gt;, "ArrowMapSize:=", &lt;bool&gt;,</pre>

			<pre>"ArrowMapColor:=", &lt;bool&gt;, "ShowArrowTail:=", &lt;bool&gt;, "ArrowSize:=", &lt;double&gt;)</pre>
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>SetPlotFolderSettings(&lt;PlotFolderName&gt;, &lt;PlotFolderAttributes&gt;)</code>
<b>Python Example</b>	<pre>oModule.SetPlotFolderSettings("E Field1", ["NAME:FieldsPlotSettings",  "Real time mode:=", True,  ["NAME:ColorMapSettings",  "ColorMapType:=", "Spectrum",  "SpectrumType:=", "Rainbow",  "UniformColor:=", [127, 255, 255],  "RampColor:=", [255, 127, 127]],  ["NAME:Scale3DSettings",  "m_nLevels:=", 27,  "m_autoScale:=", True,  "minvalue:=", 9.34379863739014,  "maxvalue:=", 13683.755859375,</pre>

```

    "log:=", False,
    "IntrinsicMin:=", 9.34379863739014,
    "IntrinsicMax:=", 13683.755859375),
    [ "NAME:Marker3DSettings",
      "MarkerType:=", 0,
      "MarkerMapSize:=", True,
      "MarkerMapColor:=", False,
      "MarkerSize:=", 0.25],
    [ "NAME:Arrow3DSettings",
      "ArrowType:=", 1,
      "ArrowMapSize:=", True,
      "ArrowMapColor:=", True,
      "ShowArrowTail:=", True,
      "ArrowSize:=", 0.25]]
)

```

<b>VB Syntax</b>	SetPlotFolderSettings <PlotFolderName>, <PlotFolderAttributes>
<b>VB Example</b>	<pre> oModule.SetPlotFolderSettings "E Field1", _   Array("NAME:FieldsPlotSettings", _     "Real time mode:=", true, _     Array("NAME:ColorMapSettings", _ </pre>

```
"ColorMapType:=", "Spectrum", _
"SpectrumType:=", "Rainbow", _
"UniformColor:=", Array(127, 255, 255), _
"RampColor:=", Array(255, 127, 127)), _
Array("NAME:Scale3DSettings", _
"m_nLevels:=", 27, _
"m_autoScale:=", true, _
"minvalue:=", 9.34379863739014, _
"maxvalue:=", 13683.755859375, _
"log:=", false, _
"IntrinsicMin:=", 9.34379863739014, _
"IntrinsicMax:=", 13683.755859375), _
Array("NAME:Marker3DSettings", _
"MarkerType:=", 0, _
"MarkerMapSize:=", true, _
"MarkerMapColor:=", false, _
"MarkerSize:=", 0.25), _
Array("NAME:Arrow3DSettings", _
"ArrowType:=", 1, _
"ArrowMapSize:=", true, _
```

```
"ArrowMapColor:=", true, _  
"ShowArrowTail:=", true, _  
"ArrowSize:=", 0.25))
```

## SetPlotsViewSolutionContext

Sets the context of named field overlay plots to the specified time using the intrinsic Time value of the active window. Also updates the associated view solution context for each plot.

UI Access	[product] > Fields > Set Context to Active Window		
Parameters	Name	Type	Description
	<Plots>	Array	Names of the plots to modify.
	<SolName>	String	Name of a transient analysis solution.
	<SolTime>	String	Intrinsic variable name (must be "Time") and value.
Return Value	None.		

Python Syntax	SetPlotsViewSolutionContext(<Plots>, <SolName>, <SolTime>)
Python Example	<pre>oModule.SetPlotsViewSolutionContext(     ["Flux_Lines1", "Mag_H1"],     "Setup1 : Transient", "Time=-1")</pre>

VB Syntax	SetPlotsViewSolutionContext <Plots>, <SolName>, <SolTime>
VB Example	<pre>oModule.SetPlotsViewSolutionContext _</pre>

	Array("Flux_Lines1", "Mag_H1"), "Setup1 : Transient", "Time=-1"
--	--------------------------------------------------------------------

## UpdateAllFieldsPlots

Updates the All Fields Plots.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	UpdateAllFieldsPlots()
<b>Python Example</b>	oModule.UpdateAllFieldsPlots()

<b>VB Syntax</b>	UpdateAllFieldsPlots
<b>VB Example</b>	oModule.UpdateAllFieldsPlots

## UpdateQuantityFieldsPlots

Updates the Quantity Fields Plots.

<b>UI Access</b>	N/A
------------------	-----

<b>Parameters</b>	Name <i>&lt;FolderName&gt;</i>	Type String	Description Folder containing the plotted quantities.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	UpdateQuantityFieldsPlots( <i>&lt;FolderName&gt;</i> )
<b>Python Example</b>	<code>oModule.UpdateQuantityFieldsPlots ("fldplt")</code>

<b>VB Syntax</b>	UpdateQuantityFieldsPlots <i>&lt;FolderName&gt;</i>
<b>VB Example</b>	<code>oModule.UpdateQuantityFieldsPlots "fldplt"</code>

This page intentionally  
left blank.

# 19 - Fields Calculator Script Commands

Fields Calculator commands should be executed by the Field Overlays module, which is called "FieldsReporter" in Maxwell scripts.

```
Set oModule = oDesign.GetModule("FieldsReporter")
oModule.CommandName <args>
```

The command associated with each of the following scripting commands will be a button pressed in the Fields Calculator.

[AddNamedExpression](#)

[AddNamedExpr](#)

[CalcOp](#)

[CalculatorRead](#)

[CalcStack](#)

[CalculatorWrite](#)

[ChangeGeomSettings](#)

[ClcEval](#)

[ClcMaterial](#)

[ClearAllNamedExpr](#)

[CopyNamedExprToStack](#)

[DeleteNamedExpr](#)

[EnterComplex](#)

[EnterComplexVector](#)

[EnterLine](#)

[EnterPoint](#)[EnterQty](#)[EnterScalar](#)[EnterScalarFunc](#)[EnterSurf](#)[EnterVector](#)[EnterVectorFunc](#)[EnterVol](#)[ExportOnGrid \[Fields Calculator\]](#)[ExportToFile \[Fields Calculator\]](#)[GetTopEntryValue](#)[LoadNamedExpressions](#)[SaveNamedExpressions](#)

## AddNamedExpression

Creates a named expression using the expression at the top of the stack.

<b>UI Access</b>	Click <b>Add</b> in the <b>Fields Calculator</b> dialogue.		
<b>Parameters</b>	Name	Type	Description
	<ExprName>	String	Name for the new named expression.
<b>Return Value</b>	None.		

---

<b>Python Syntax</b>	AddNamedExpression(< <i>ExprName</i> >, < <i>FieldType</i> >)
<b>Python Example</b>	<code>oModule.AddNamedExpression ("Mag_JxE", "Fields")</code>

<b>VB Syntax</b>	AddNamedExpression < <i>ExprName</i> >, < <i>FieldType</i> >
<b>VB Example</b>	<code>oModule.AddNamedExpression "Mag_JxE", "Fields"</code>

## AddNamedExpr

Creates a named expression using the expression at the top of the stack.

<b>UI Access</b>	Click <b>Add</b> in the <b>Fields Calculator</b> dialogue.						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;<i>ExprName</i>&gt;</td> <td>String</td> <td>Name for the new named expression.</td> </tr> </tbody> </table>	Name	Type	Description	< <i>ExprName</i> >	String	Name for the new named expression.
Name	Type	Description					
< <i>ExprName</i> >	String	Name for the new named expression.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	AddNamedExpr(< <i>ExprName</i> >)
<b>Python Example</b>	<code>oModule.AddNamedExpr ("Mag_JxE")</code>

<b>VB Syntax</b>	AddNamedExpr < <i>ExprName</i> >
<b>VB Example</b>	<code>oModule.AddNamedExpr "Mag_JxE"</code>

## CalcOp

Performs a calculator operation.

<b>UI Access</b>	Operation commands like <b>Mag</b> , <b>+</b> , etc.		
<b>Parameters</b>	Name <i>&lt;OpString&gt;</i>	Type String	Description The text on the corresponding calculator button.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CalcOp( <i>&lt;OpString&gt;</i> )
<b>Python Example</b>	<code>oModule.CalcOp ("+")</code>

<b>VB Syntax</b>	CalcOp <i>&lt;OpString&gt;</i>
<b>VB Example</b>	<code>oModule.CalcOp "+"</code>

## CalcRead(deprecated)

Reads a file that is written out by the CalcWrite command, and puts the result into a calculator numeric.

<b>UI Access</b>	Click <b>Read...</b> in the <b>Fields Calculator</b> dialog.		
<b>Parameters</b>	Name <i>&lt;FileName&gt;</i>	Type String	Description Name of the file to be read.
	<i>&lt;SoluName&gt;</i>	Type String	Description Specified solution to read in.

	<code>&lt;VarArray&gt;</code>	Array	Array of variables to read.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>CalcRead(&lt;FileName&gt;, &lt;SoluName&gt;, &lt;VarArray&gt;)</code>
<b>Python Example</b>	<pre>oModule.CalcRead(     "c:\example.reg",     "Setup1: LastAdaptive",     ["Freq:=", "10GHz", "Phase:=", "0deg"])</pre>

<b>VB Syntax</b>	<code>CalcRead &lt;FileName&gt;, &lt;SoluName&gt;, &lt;VarArray&gt;</code>
<b>VB Example</b>	<pre>oModule.CalcRead _     "c:\example.reg", _     "Setup1: LastAdaptive", _     Array("Freq:=", "10GHz", "Phase:=", "0deg")</pre>

## CalculatorRead

Gets a register file and applies it to the calculator stack.

<b>UI Access</b>	Click <b>Read...</b> in the <b>Fields Calculator</b> dialog.								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;FileName&gt;</code></td> <td>String</td> <td>Path to and including name of input register file.</td> </tr> </tbody> </table>			Name	Type	Description	<code>&lt;FileName&gt;</code>	String	Path to and including name of input register file.
Name	Type	Description							
<code>&lt;FileName&gt;</code>	String	Path to and including name of input register file.							

	<i>&lt;SoluName&gt;</i>	String	Specified solution to read in.
	<i>&lt;FieldType&gt;</i>	String	Type of specified field.
	<i>&lt;VarArray&gt;</i>	Array	Array of variable names, value pairs.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	CalculatorRead(<FileName>, <SoluName>, <FieldType>, <VarArray>)
<b>Python Example</b>	<pre>oModule.CalculatorRead(     "c:\\example.reg",     "Setup1: LastAdaptive", "Fields",     ["Freq:=", "10GHz", "Phase:=", "0deg"])</pre>

<b>VB Syntax</b>	CalculatorRead <FileName>, <SoluName>, <FieldType>, <VarArray>
<b>VB Example</b>	<pre>oModule.CalculatorRead _     "c:\\example.reg", _     "Setup1: LastAdaptive", "Fields", _     Array("Freq:=", "10GHz", "Phase:=", "0deg")</pre>

## CalcStack

Performs an operation on the stack.

<b>UI Access</b>	Stack operation buttons such as <b>Push</b> and <b>Pop</b> .								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;OpString&gt;</code></td> <td>String</td> <td>The text on the corresponding calculator button.</td> </tr> </tbody> </table>			Name	Type	Description	<code>&lt;OpString&gt;</code>	String	The text on the corresponding calculator button.
Name	Type	Description							
<code>&lt;OpString&gt;</code>	String	The text on the corresponding calculator button.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	<code>CalcStack(&lt;OpString&gt;)</code>
<b>Python Example</b>	<code>oModule.CalcStack("push")</code>

<b>VB Syntax</b>	<code>CalcStack &lt;OpString&gt;</code>
<b>VB Example</b>	<code>oModule.CalcStack "push"</code>

## CalculatorWrite

Writes contents of top register to file.

<b>UI Access</b>	Click <b>Write...</b> in the <b>Fields Calculator</b> dialog.														
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;OutputFilePath&gt;</code></td> <td>String</td> <td>Path to and including name of output register file.</td> </tr> <tr> <td><code>&lt;SoluNameArray&gt;</code></td> <td>Array</td> <td>Array of strings containing solution names.</td> </tr> <tr> <td><code>&lt;VarArray&gt;</code></td> <td>Array</td> <td>Array of variable names, value pairs.</td> </tr> </tbody> </table>			Name	Type	Description	<code>&lt;OutputFilePath&gt;</code>	String	Path to and including name of output register file.	<code>&lt;SoluNameArray&gt;</code>	Array	Array of strings containing solution names.	<code>&lt;VarArray&gt;</code>	Array	Array of variable names, value pairs.
Name	Type	Description													
<code>&lt;OutputFilePath&gt;</code>	String	Path to and including name of output register file.													
<code>&lt;SoluNameArray&gt;</code>	Array	Array of strings containing solution names.													
<code>&lt;VarArray&gt;</code>	Array	Array of variable names, value pairs.													
<b>Return Value</b>	None.														

<b>Python Syntax</b>	CalculatorWrite(<OutputFilePath>, <SoluNameArray>, <VarArray>)
<b>Python Example</b>	<pre> oModule.CalculatorWrite("c:\test.reg",     ["Solution:=", "Setup1 : LastAdaptive"],     ["Freq:=", "1GHz", "Phase:=", "0deg"] ) </pre>

<b>VB Syntax</b>	CalculatorWrite <OutputFilePath>, <SoluNameArray>, <VarArray>
<b>VB Example</b>	<pre> oModule.CalculatorWrite "c:\test.reg", _     Array("Solution:=", "Setup1 : LastAdaptive"), _     Array("Freq:=", "1GHz", "Phase:=", "0deg") </pre>

## ChangeGeomSettings

Changes the line discretization setting.

<b>UI Access</b>	In the <b>Fields Calculator</b> dialog box, click on <b>Geom Settings...</b>						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;LineDiscr&gt;</td> <td>Integer</td> <td>Line discretization value.</td> </tr> </tbody> </table>	Name	Type	Description	<LineDiscr>	Integer	Line discretization value.
Name	Type	Description					
<LineDiscr>	Integer	Line discretization value.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	ChangeGeomSettings(<LineDiscr>)
----------------------	---------------------------------

<b>Python Example</b>	<code>oModule.ChangeGeomSettings(500)</code>
-----------------------	----------------------------------------------

<b>VB Syntax</b>	<code>ChangeGeomSettings &lt;LineDiscr&gt;</code>
<b>VB Example</b>	<code>oModule.ChangeGeomSettings 500</code>

## ClcEval

Evaluates the expression at the top of the stack using the provided solution name and variable values.

<b>UI Access</b>	Click <b>Eval</b> in the <b>Fields Calculator</b> dialog.												
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;SoluName&gt;</code></td> <td>String</td> <td>Name of specified solution.</td> </tr> <tr> <td><code>&lt;VarArray&gt;</code></td> <td>Array</td> <td>Array of variable name, value pairs.</td> </tr> <tr> <td><code>&lt;FieldType&gt;</code></td> <td>String</td> <td>Optional. Type of specified field.</td> </tr> </tbody> </table>	Name	Type	Description	<code>&lt;SoluName&gt;</code>	String	Name of specified solution.	<code>&lt;VarArray&gt;</code>	Array	Array of variable name, value pairs.	<code>&lt;FieldType&gt;</code>	String	Optional. Type of specified field.
Name	Type	Description											
<code>&lt;SoluName&gt;</code>	String	Name of specified solution.											
<code>&lt;VarArray&gt;</code>	Array	Array of variable name, value pairs.											
<code>&lt;FieldType&gt;</code>	String	Optional. Type of specified field.											
<b>Return Value</b>	None.												

<b>Python Syntax</b>	<code>ClcEval(&lt;SoluName&gt;, &lt;VarArray&gt;, [Optional &lt;FieldType&gt;])</code>
<b>Python Example</b>	<pre>oModule.ClcEval(     "Setup1: LastAdaptive",     ["Freq:=", "10GHz",      "Phase:=", "0deg"])</pre>

<b>VB Syntax</b>	ClcEval <SolName>, <VarArray>, [Optional <FieldType>]
<b>VB Example</b>	<pre>oModule.ClcEval "Setup1: LastAdaptive", _     Array ("Freq:=", "10GHz", _         "Phase:=", "0deg")</pre>

## ClcMaterial

Performs a material operation on the top stack element.

<b>UI Access</b>	Click <b>Matl...</b> in the <b>Fields Calculator</b> dialog.									
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;MatString&gt;</td><td>String</td><td>The material property to apply.</td></tr><tr><td>&lt;OpString&gt;</td><td>String</td><td>Name of operation. Possible values are "mult", or "div".</td></tr></table>	Name	Type	Description	<MatString>	String	The material property to apply.	<OpString>	String	Name of operation. Possible values are "mult", or "div".
Name	Type	Description								
<MatString>	String	The material property to apply.								
<OpString>	String	Name of operation. Possible values are "mult", or "div".								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	ClcMaterial(<MatString>, <OpString>)
<b>Python Example</b>	<pre>oModule.ClcMaterial("Permeability (mu)" "mult")</pre>

<b>VB Syntax</b>	ClcMaterial <MatString>, <OpString>
<b>VB Example</b>	<pre>oModule.ClcMaterial "Permeability (mu)" "mult"</pre>

## ClearAllNamedExpr

Clears all user-defined named expressions from the list.

<b>UI Access</b>	Click <b>ClearAll</b> in the <b>Fields Calculator</b> dialog.
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	<code>ClearAllNamedExpr()</code>
<b>Python Example</b>	<code>oModule.ClearAllNamedExpr ()</code>

<b>VB Syntax</b>	<code>ClearAllNamedExpr</code>
<b>VB Example</b>	<code>oModule.ClearAllNamedExpr</code>

## CopyNamedExprToStack

Copies the named expression selected to the calculator stack.

<b>UI Access</b>	Select a named expression and then click <b>Copy to stack</b> .						
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><code>&lt;ExprName&gt;</code></td> <td>String</td> <td>Name of the expression to be copied to the top of the stack.</td> </tr> </table>	Name	Type	Description	<code>&lt;ExprName&gt;</code>	String	Name of the expression to be copied to the top of the stack.
Name	Type	Description					
<code>&lt;ExprName&gt;</code>	String	Name of the expression to be copied to the top of the stack.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	CopyNamedExprToStack(<ExprName>)
<b>Python Example</b>	<code>oModule.CopyNamedExprToStack ("Mag_JxE")</code>

<b>VB Syntax</b>	CopyNamedExprToStack <ExprName>
<b>VB Example</b>	<code>oModule.CopyNamedExprToStack "Mag_JxE"</code>

## DeleteNamedExpr

Deletes the selected named expression from the list.

<b>UI Access</b>	Select a named expression and then click <b>Delete</b> .						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;ExprName&gt;</td><td>String</td><td>Name of specified named expression.</td></tr></table>	Name	Type	Description	<ExprName>	String	Name of specified named expression.
Name	Type	Description					
<ExprName>	String	Name of specified named expression.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	DeleteNamedExpr(<ExprName>)
<b>Python Example</b>	<code>oModule.DeleteNamedExpr ("Mag_JxE")</code>

<b>VB Syntax</b>	DeleteNamedExpr <ExprName>
<b>VB Example</b>	<code>oModule.DeleteNamedExpr "Mag_JxE"</code>

## EnterComplex

Enters a complex number onto the stack.

<b>UI Access</b>	Click <b>Number</b> , and then click <b>Scalar. Complex</b> option is selected.		
<b>Parameters</b>	Name <i>&lt;ComplexNum&gt;</i>	Type String	Description String of complex value. Ex. "1 + 2j".
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EnterComplex(<ComplexNum>)
<b>Python Example</b>	oModule.EnterComplex ("1 + 2 j")

<b>VB Syntax</b>	EnterComplex <ComplexNum>
<b>VB Example</b>	oModule.EnterComplex "1 + 2 j"

## EnterComplexVector

Enters a complex vector onto the stack.

<b>UI Access</b>	Click <b>Number</b> , and then click <b>Vector. Complex</b> option is selected.		
<b>Parameters</b>	Name <i>&lt;ComplexVector&gt;</i>	Type Array	Description Array of strings containing X, Y and Z complex values.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EnterComplexVector(<ComplexVector>)
<b>Python Example</b>	<pre>oModule.EnterComplexVector ([     "1 + 2 j",     "1 + 2 j",     "1 + 2 j" ])</pre>

<b>VB Syntax</b>	EnterComplexVector <ComplexVector>
<b>VB Example</b>	<pre>oModule.EnterComplexVector _     Array("1 + 2 j",_         "1 + 2 j",_         "1 + 2 j")</pre>

## EnterLine

Enters a line defined in the 3D Modeler editor.

<b>UI Access</b>	Click <b>Geometry</b> and then select <b>Line</b>		
<b>Parameters</b>	Name	Type	Description

	<i>&lt;LineName&gt;</i>	String	Name of a line defined in the 3D Modeler editor.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	EnterLine(<LineName>)
<b>Python Example</b>	<code>oModule.EnterLine("Line1")</code>

<b>VB Syntax</b>	EnterLine <LineName>
<b>VB Example</b>	<code>oModule.EnterLine "Line1"</code>

## EnterPoint

Enters a point defined in the 3D Modeler editor.

<b>UI Access</b>	Click <b>Geometry</b> and then select <b>Point</b> .								
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td><i>&lt;PointName&gt;</i></td> <td>String</td> <td>Name of a point defined in the 3D Modeler editor.</td> </tr> </table>			Name	Type	Description	<i>&lt;PointName&gt;</i>	String	Name of a point defined in the 3D Modeler editor.
Name	Type	Description							
<i>&lt;PointName&gt;</i>	String	Name of a point defined in the 3D Modeler editor.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	EnterPoint(<PointName>)
<b>Python Example</b>	<code>oModule.EnterPoint("Point1")</code>

<b>VB Syntax</b>	EnterPoint <PointName>
<b>VB Example</b>	<code>oModule.EnterPoint "Point1"</code>

## EnterQty

Enters a field quantity.

<b>UI Access</b>	Click <b>Quantity</b> , and then select from the list.						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;FieldQty&gt;</td><td>String</td><td>The field quantity to be entered onto the stack.</td></tr></table>	Name	Type	Description	<FieldQty>	String	The field quantity to be entered onto the stack.
Name	Type	Description					
<FieldQty>	String	The field quantity to be entered onto the stack.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	EnterQty(<FieldQty>)
<b>Python Example</b>	<code>oModule.EnterQty ("E")</code>

<b>VB Syntax</b>	EnterQty <FieldQty>
<b>VB Example</b>	<code>oModule.EnterQty "E"</code>

## EnterScalar

Enters a scalar onto the stack.

<b>UI Access</b>	Click <b>Number</b> and then click <b>Scalar</b> . <b>Complex</b> option not selected.
------------------	----------------------------------------------------------------------------------------

<b>Parameters</b>	Name <code>&lt;Scalar&gt;</code>	Type Double	Description The real number to enter onto the stack.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>EnterScalar(&lt;Scalar&gt;)</code>
<b>Python Example</b>	<code>oModule.EnterScalar(3.14159265358979)</code>

<b>VB Syntax</b>	<code>EnterScalar &lt;Scalar&gt;</code>
<b>VB Example</b>	<code>oModule.EnterScalar 3.14159265358979</code>

## EnterScalarFunc

Enters a scalar function.

<b>UI Access</b>	Click <b>Function</b> and then select <b>Scalar</b> .		
<b>Parameters</b>	Name <code>&lt;VarName&gt;</code>	Type String	Description Name of a variable to enter as a scalar function onto the stack.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>EnterScalarFunc(&lt;VarName&gt;)</code>
<b>Python Example</b>	<code>oModule.EnterScalarFunc ("Phase")</code>

<b>VB Syntax</b>	EnterScalarFunc <VarName>
<b>VB Example</b>	oModule.EnterScalarFunc "Phase"

## EnterSurf

Enters a surface defined in the 3D Modeler editor.

<b>UI Access</b>	Click <b>Geometry</b> and then select <b>Surface</b> .						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;SurfName&gt;</td><td>String</td><td>Name of a surface defined in the 3D Modeler editor.</td></tr></table>	Name	Type	Description	<SurfName>	String	Name of a surface defined in the 3D Modeler editor.
Name	Type	Description					
<SurfName>	String	Name of a surface defined in the 3D Modeler editor.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	EnterSurf(<SurfName>)
<b>Python Example</b>	oModule.EnterSurf("Rectangle1")

<b>VB Syntax</b>	EnterSurf <SurfName>
<b>VB Example</b>	oModule.EnterSurf "Rectangle1"

## EnterVector

Enters a vector onto the stack.

<b>UI Access</b>	Click <b>Number</b> , and then click <b>Vector</b> . <b>Complex</b> option not selected.
------------------	------------------------------------------------------------------------------------------

<b>Parameters</b>	Name <code>&lt;VectorArray&gt;</code>	Type Array	Description Array of strings containing X, Y and Z components of the vector.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>EnterVector(&lt;VectorArray&gt;)</code>
<b>Python Example</b>	<code>oModule.EnterVector([1.0, 1.0, 1.0])</code>

<b>VB Syntax</b>	<code>EnterVector &lt;VectorArray&gt;</code>
<b>VB Example</b>	<code>oModule.EnterVector Array(1.0, 1.0, 1.0)</code>

## EnterVectorFunc

Enters a vector function.

<b>UI Access</b>	Click <b>Function</b> and then select <b>Vector</b> .		
<b>Parameters</b>	Name <code>&lt;VarNames&gt;</code>	Type Array	Description Array of strings containing names of a variable for the X, Y, and Z coordinates, respectively, to enter as a vector function on the stack.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>EnterVectorFunc(&lt;VarNames&gt;)</code>
<b>Python Example</b>	<code>oModuleEnterVectorFunc(["X", "Y", "Z"])</code>

<b>VB Syntax</b>	EnterVectorFunc <VarNames>
<b>VB Example</b>	<code>oModuleEnterVectorFunc Array("X", "Y", "Z")</code>

## EnterVol

Enters a volume defined in the 3D Modeler editor.

<b>UI Access</b>	Click <b>Geometry</b> and then select <b>Volume</b> .						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;VolumeName&gt;</td><td>String</td><td>Name of a volume defined in the 3D Modeler editor.</td></tr></table>	Name	Type	Description	<VolumeName>	String	Name of a volume defined in the 3D Modeler editor.
Name	Type	Description					
<VolumeName>	String	Name of a volume defined in the 3D Modeler editor.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	EnterVol(<VolumeName>)
<b>Python Example</b>	<code>oModule.EnterVol ("Box1")</code>

<b>VB Syntax</b>	EnterVol <VolumeName>
<b>VB Example</b>	<code>oModule.EnterVol "Box1"</code>

## ExportOnGrid [Fields Calculator]

Evaluates the top stack element at a set of points specified by a grid and exports the data to a file.

<b>UI Access</b>	Click <b>Export</b> , and then click <b>On Grid</b> .		
<b>Parameters</b>	Name	Type	Description
	<OutputFile>	String	Name of the output file.
	<Min>	Array	Minimum values for the coordinate components of the grid system
	<Max>	Array	Maximum values for the coordinate components of the grid system
	<Spacing>	Array	Spacing values for the coordinate components of the grid system
	<SolnName>	String	Name of the simulation setup
	<VarVals>	Array	Array of strings containing setup definitions.
	<IncludePoints>	Boolean	Optional. Specifies whether include points in the output file.
	<CSType>	String	Optional. Type of coordinate system. "Cartesian" (default)   "Cylindrical"   "Spherical"
	<Offsets>	Array	Optional. Origin for the offset coordinate system. For Cartesian, x, y, z, for Cylindrical, R, Phi, Z, for Spherical, Rho, Theta, Phi.
	<ByCount>	Boolean	Optional.
<b>Return Value</b>	None.		

**Note:**Regarding the **ExportOnGrid** legacy script which only has “IncludePtInOutput” argument (the last Boolean one), AEDT can still read it and assign other new arguments as default values. Those default values are RefCSName = “global”, PtInSI = “True”, FieldInRefCS = “False”

<b>Python Syntax</b>	ExportOnGrid(<OutputFile>, <Min>, <Max>, <Spacing>, <SolnName>, <SolnParameters>, [<ExportOption>, "IncludePointsInOutput:=", <boolean>], "RefCSName:=", <CSName>, "PtsInSI:=", <boolean>, "FieldRefCS:=", <boolean>], "<CSType>", [<Offsets>, <ByCount>])
<b>Python Example</b>	<pre> oModule.ExportOnGrid("C:\offset_grid_model_unit_ref.fld",                      ["-1mm", "16mm", "0mm"],                      ["1mm", "18mm", "1mm"],                      ["2mm", "2mm", "1mm"],                      "4500MHz : LastAdaptive", </pre>

```

        ["Freq:=", "4.5GHz", "Phase:="      , "0deg"] ,
        [
            "NAME:ExportOption",
            "IncludePtInOutput:="   , True,
            "RefCSName:="          , "offset",
            "PtInSI:="              , False,
            "FieldInRefCS:="        , True
        ],
        "Cartesian", ["0mm", "0mm", "0mm"], False
    )
)

```

<b>VB Syntax</b>	ExportOnGrid(<OutputFile>, <Min>, <Max>, <Spacing>, <SolName>, <SolParameters>, [<ExportOption>, "IncludePointsInOutput:=", <boolean>], "RefCSName:=", <CSName>, "PtsInSI:=", <boolean>, "FieldRefCS:=", <boolean>], "<CSType>", [<Offsets>, <ByCount>])
<b>VB Example</b>	<pre> oModule.ExportOnGrid "C:/Grid.fld", _ Array( "0mm", "0deg", "-25mm"), _ Array("20mm", "90deg", "125mm"), _ Array("10mm", "45deg", "50mm"), _ "Setup1 : LastAdaptive", _ Array("Freq:=", "10000Hz", "Phase:=", "0deg"), _ </pre>

```
true, "Cylindrical", _  
    Array("0mm", "0mm", "0mm")  
  
oModule.ExportOnGrid("C:\offset_grid_model_unit_ref.fld",  
    Array("-1mm", "16mm", "0mm"),  
    Array("1mm", "18mm", "1mm"),  
    Array("2mm", "2mm", "1mm"),  
    "4500MHz : LastAdaptive",  
    Array("Freq:=", "4.5GHz", "Phase:=" , "0deg"),  
    [  
        "NAME:ExportOption",  
        "IncludePtInOutput:=" , True,  
        "RefCSName:=" , "offset",  
        "PtInSI:=" , False,  
        "FieldInRefCS:=" , True  
    ],  
    "Cartesian", Array("0mm", "0mm", "0mm"), False  
)
```

## ExportToFile [Fields Calculator]

Evaluates the top stack element at a set of points specified in an external file and exports the data to a file.

**Note:**

Regarding to legacy **ExportToFile** script which only has “IncludePtInOutput” argument (the last Boolean one), AEDT can still read it and assign other new arguments as default values. Those default values are RefCSName = “global”, PtInSI = “True”, FieldInRefCS = “False”

<b>UI Access</b>	Click <b>Export</b> , and then click <b>To File</b> .																					
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ReportName&gt;</td> <td>String</td> <td>Name of report to be exported.</td> </tr> <tr> <td>&lt;FileName&gt;</td> <td>String</td> <td>Full path of the exported image file name; with extension of .txt - Post processor format file .csv - Comma-delimited data file .tab - Tab-separated file .dat - Ansys plot data file</td> </tr> <tr> <td>&lt;solution&gt;</td> <td></td> <td>Solution Name</td> </tr> <tr> <td>&lt;freq&gt;</td> <td></td> <td>frequency</td> </tr> <tr> <td>&lt;phase&gt;</td> <td></td> <td>phase</td> </tr> <tr> <td>&lt;Export Options&gt;</td> <td></td> <td>Whether to include points in output, RefCSName, Pts in SI units &lt;boolean&gt;, Field in RefCS, &lt;boolean&gt;.</td> </tr> </tbody> </table>	Name	Type	Description	<ReportName>	String	Name of report to be exported.	<FileName>	String	Full path of the exported image file name; with extension of .txt - Post processor format file .csv - Comma-delimited data file .tab - Tab-separated file .dat - Ansys plot data file	<solution>		Solution Name	<freq>		frequency	<phase>		phase	<Export Options>		Whether to include points in output, RefCSName, Pts in SI units <boolean>, Field in RefCS, <boolean>.
Name	Type	Description																				
<ReportName>	String	Name of report to be exported.																				
<FileName>	String	Full path of the exported image file name; with extension of .txt - Post processor format file .csv - Comma-delimited data file .tab - Tab-separated file .dat - Ansys plot data file																				
<solution>		Solution Name																				
<freq>		frequency																				
<phase>		phase																				
<Export Options>		Whether to include points in output, RefCSName, Pts in SI units <boolean>, Field in RefCS, <boolean>.																				
<b>Return Value</b>	None.																					

<b>Python Syntax</b>	ExportToFile(<ReportName>, <FileName>, <solution>, ["Freq:=", <freq>, "Phase:=", "<phase>"], <ExportOptions>)
<b>Python Example</b>	<pre> oModule.ExportToFile("C:\\\\offset_file_model_unit_ref.fld", "C:\\\\offset_SI.pts", "4500MHz : LastAdaptive", [     "Freq:="          , "4.5GHz",     "Phase:="         , "0deg" ], [     "NAME:ExportOption",     "IncludePtInOutput:=" , True,     "RefCSName:="        , "offset",     "PtInSI:="           , False,     "FieldInRefCS:="     , True ]) </pre>

<b>VB Syntax</b>	ExportToFile <ReportName>, <FileName>
<b>VB Example</b>	<pre> oModule.ExportToFile("C:\\\\offset_file_model_unit_ref.fld", "C:\\\\offset_SI.pts", "4500MHz : LastAdaptive", [     "Freq:="          , "4.5GHz", ] </pre>

```
    "Phase:="           , "0deg"
] ,
[
    "NAME:ExportOption",
    "IncludePtInOutput:=" , True,
    "RefCSName:="         , "offset",
    "PtInSI:="            , False,
    "FieldInRefCS:="      , True
])
```

## GetTopEntryValue

Gets the value of the top entry of the calculator stack.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<SolnName>	String	Name of specified solution.
<b>Return Value</b>	Array of strings containing top entry values.		

<b>Python Syntax</b>	GetTopEntryValue(<SolnName>, <VarVals>)
----------------------	-----------------------------------------

<b>Python Example</b> <pre> oModule.GetTopEntryValue(     "Setup1:LastAdaptive",     ["Freq:=", "1GHz", "Phase:=", "0deg", "x_size:=", "2mm"] ) </pre>
--------------------------------------------------------------------------------------------------------------------------------------------------------

<b>VB Syntax</b> <pre>GetTopEntryValue &lt;SolnName&gt;, &lt;VarVals&gt;</pre>
<b>VB Example</b> <pre> oModule.GetTopEntryValue "Setup1:LastAdaptive", _ Array("Freq:=", "1GHz", "Phase:=", "0deg", _ "x_size:=", "2mm") </pre>

## LoadNamedExpressions

Loads a named expression definition from a saved file.

<b>UI Access</b>	In the Fields Calculator, click <b>Load From...</b> in the Library area.												
<b>Parameters</b>	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;FileName&gt;</td> <td>String</td> <td>Filename and full path to the file to hold the named expression definition.</td> </tr> <tr> <td>&lt;FieldType&gt;</td> <td>String</td> <td>For products with just one filed type, it is set to "Fields".</td> </tr> <tr> <td>&lt;NamedExpr&gt;</td> <td>Array</td> <td>rray of strings containing the names of expression definitions to load from the file.</td> </tr> </tbody> </table>	Name	Type	Description	<FileName>	String	Filename and full path to the file to hold the named expression definition.	<FieldType>	String	For products with just one filed type, it is set to "Fields".	<NamedExpr>	Array	rray of strings containing the names of expression definitions to load from the file.
Name	Type	Description											
<FileName>	String	Filename and full path to the file to hold the named expression definition.											
<FieldType>	String	For products with just one filed type, it is set to "Fields".											
<NamedExpr>	Array	rray of strings containing the names of expression definitions to load from the file.											
<b>Return Value</b>	None.												

<b>Python Syntax</b>	LoadNamedExpressions(<FileName>, <FieldType>, <NamedExpr>)
----------------------	------------------------------------------------------------

**Python Example**

```
oModule.LoadNamedExpressions(  
    "C:\Ansoft\PersonalLib\smth.clc",  
    "Fields", ["smoothedtemp"])
```

**VB Syntax**

```
LoadNamedExpressions <FileName>, <FieldType>, <NamedExpr>
```

**VB Example**

```
oModule.LoadNamedExpressions _  
    "C:\Ansoft\PersonalLib\smth.clc", _  
    "Fields", Array("smoothedtemp")
```

## SaveNamedExpressions

Saves a named expression definition to a file.

<b>UI Access</b>	In the Fields Calculator, click <b>Save To...</b> in the Library area.	
<b>Parameters</b>		
Name	Type	Description
<FileName>	String	Filename and full path to the file to hold the named expression definition.
<NamedExprs>	Array	Array of strings containing the names of expression definitions to load from the file.
<Overwrite>	Boolean	Specifies whether to overwrite the file.
<b>Return Value</b>	None.	

**Python Syntax**

```
SaveNamedExpressions(<FileName>, <NamedExprs>, <Overwrite>)
```

<b>Python Example</b>	<pre>oModule.SaveNamedExpressions(     "C:\Ansoft\PersonalLib\smth.clc",     ["smoothedtemp"], True)</pre>
-----------------------	--------------------------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	SaveNamedExpressions <FileName>, <NamedExprs>, <Overwrite>
<b>VB Example</b>	<pre>oModule.SaveNamedExpressions _     "C:\Ansoft\PersonalLib\smth.clc", _     Array("smoothedtemp"), true</pre>

This page intentionally  
left blank.

# 20 - Motion Setup Script Commands

Motion setup commands should be executed by the **ModelSetup** module.

```
Set oModule = oDesign.GetModule("ModelSetup")
oModule.CommandName <args>
```

## Conventions Used in the Motion Setup Chapter

<AssignmentObjects>

Type: Array of strings

An array of object names.

## General Motion Setup Script Commands

Following are general script commands recognized by the **ModelSetup** module:

- [DeleteMotionSetup](#)
- [ReassignMoving](#)

## DeleteMotionSetup

**Use:** Deletes the motion setup.

**Command:** Maxwell3D or Maxwell2D>Model>Motion Setup>Unassign Band

**Syntax:** DeleteMotionSetup

**Return Value:** None

**Parameters:** List of strings. A list of names of motion setups to delete.

**Example:** oModule.DeleteMotionSetup(["MotionSetup2"])

The [ ] can contain more than one name as it is a list.

## ReassignMoving

**Use:** Specifies a new geometry assignment for moving objects.

**Command:** Maxwell3D or Maxwell2D>Model>Motion Setup>Add Selected Objects or  
Maxwell3D or Maxwell2D>Model>Motion Setup>Remove Selected Objects

**Syntax:** ReassignMoving Array("Name:Moving", "Objects:=", <AssignmentObjects>)

**Return Value:** None

**Example:** oModule.ReassignMoving Array ("NAME:Moving", \_  
"Objects:=", Array ("Box4", "Box1"))

## Commands to Create and Edit the Band

Following are script commands for creating and editing bands that are recognized by the  
**ModelSetup** module:

### Note:

In the following commands, all named data can be included/excluded as desired and may appear in any order.

- [AssignBand](#)
- [EditMotionSetup](#)
- [GetMotionSetupNames](#)

## AssignBand

**Use:** Assigns the selected object as a band.

**Command:** Maxwell3D or Maxwell2D>Model>Motion Setup>Assign Band

**Syntax:**AssignBand <BanddataArray>

**Return Value:** None

*Parameters:*

<BanddataArray>

```
Array("NAME:Band",
"Move Type:=", <Translate/Rotate>,
"Coordinate System:::", <CoordinateSystemName>,
"Axis:=", <X/Y/Z>,
"Is Positive:=", <bool>,
"InitPos:=", <value>,
"NegativePos:=", <value>,
"PositivePos:=", <value>,
"TranslatePeriodic:=", <bool>,
"Consider Mechanical Transient:=", <bool>,
"velocity:=", <value>,
"objects:=", <AssignmentObjects>)
```

**Example:** Assign band as translate, do not consider mechanical transient.

```
oModule.AssignBand _
Array("NAME:Band", "Move Type:=", "Translate",
"Coordinate System:=", "Global", "Axis:=", "Z",
"Is Positive:=", true, "InitPos:=", "0mm", _
"NegativePos:=", "0mm", "PositivePos:=", "1mm",
```

```
"TranslatePeriodic:=", true,  
"Consider Mechanical Transient:=", false,  
"Velocity:=", "0m_per_sec",  
"Objects:=", Array("band"))
```

**Example:** Assign band as translate, consider mechanical transient.

```
oModule.AssignBand _  
Array("NAME:Band", _  
"Move Type:=", "Translate",  
"Coordinate System:=", "Global", "Axis:=", "Z",  
"Is Positive:=", true, "InitPos:=", "0mm",  
"NegativePos:=", "0mm", "PositivePos:=", "1mm",  
"TranslatePeriodic:=", true,  
"Consider Mechanical Transient:=", true,  
"Velocity:=", "0m_per_sec", "Mass:=", "1kg",  
"Damping:=", "1", "Load Force:=", "1nNewton"  
"Objects:=", Array("band"))
```

**Example:** Assign band as rotate, do not consider mechanical transient.

```
oModule.AssignBand _  
Array("NAME:Band", "Move Type:=", "Rotate",  
"Coordinate System:=", "Global", "Axis:=", "Z",
```

```
"Is Positive:=", true, "InitPos:=", "0deg",
"HasRotateLimit:=", false, "NonCylindrical:=",
false, "Consider Mechanical Transient:=", false,
"Angular Velocity:=", "0deg_per_sec", "Objects:=",
Array("band"))
```

**Example:** Assign band as rotate, consider mechanical transient.

```
oModule.AssignBand _
Array("NAME:Band", "Move Type:=", "Rotate",
"Coordinate System:=", "Global", "Axis:=", "Z",
"Is Positive:=", true, "InitPos:=", "0deg",
"HasRotateLimit:=", false, "NonCylindrical:=",
false, "Consider Mechanical Transient:=", true,
"Angular Velocity:=", "0deg_per_sec",
"Moment of Inertia:=", "1", "Damping:=", "0",
"Load Torque:=", "0NewtonMeter", "Objects:=",
Array("band"))
```

## EditMotionSetup

**Use:** Edits the motion setup.

**Command:** Double-click the moving item in the project tree to edit it.

**Syntax:** EditMotionSetup <SetupName>, <Array>

**Return Value:** None

**Parameters:** <SetupName>

The name of the motion setup to be edited.

```
<BanddataArray>

Array("NAME:Data",
"Move Type:=", <Translate/Rotate>,
"Coordinate System:=", <CoordinateSystemName>,
"PostProcessing Coordinate System:=", <CoordinateSystemName>
"Axis:=", <X/Y/Z>,
"Is Positive:=", <bool>,
"InitPos:=", <value>,
"NegativePos:=", <value>,
"PositivePos:=", <value>,
"TranslatePeriodic:=", <bool>,
"Consider Mechanical Transient:=", <bool>,
"velocity:=", <value>,
"Mass:=", <value>,
"Damping:=", <value>,
"Load Force:=", <value>)
```

**Example:** Translational Periodic motion, no mechanical transient.

```
oModule.EditMotionSetup "MotionSetup1", Array("NAME:Data", "Move Type:=", "Translate",
```

```
"Coordinate System:=", "Global",
"PostProcessing Coordinate System:=",
"Global", "Axis:=", "X", "Is Positive:=", true,
"InitPos:=", "0meter", "NegativePos:=", "0meter",
"PositivePos:=", "0.1meter",
"TranslatePeriodic:=", true,
"Consider Mechanical Transient:=", false,
"Velocity:=", "0m_per_sec")
```

## GetMotionSetupNames

**Use:** Gets the motion setup parameters being used for the object. Applicable to 2D and 3D Transient solution types.

**Command:** None

**Syntax:** GetMotionSetupNames ()

**Return Value:** List of motion setup names

**Parameters:** None

**Example:** oModule.GetMotionSetupNames ()

## Other Commands Recognized By the ModelSetup Module

The following command is also recognized by the **ModelSetup** module:

- [GetSymmetryMultiplier](#)
- [SetSymmetryMultiplier](#)

## **GetSymmetryMultiplier**

**Use:** Returns the symmetry multiplier of the model. Applicable to 2D and 3D Transient solution types.

**Command:** None

**Syntax:** oModule.GetSymmetryMultiplier ()

**Return Value:** The symmetry multiplier integer value.

**Parameters:** None

**Example:** oModule.GetSymmetryMultiplier ()

## **SetSymmetryMultiplier**

**Use:** Sets the symmetry multiplier. This symmetry multiplier will be automatically applied to all input quantities including: input voltage, inductance, resistance, load torque, mass, damping, external circuit; and all output quantities including: induced voltages, flux linkages in every winding, stranded loss, solid loss, core loss, torque and force.

**Command:** Maxwell3D or Maxwell2D>Model>Set Symmetry Multiplier

**Syntax:** SetSymmetryMultiplier <int>

**Return Value:** None

**Parameters:** <int>

**Example:** oModule.SetSymmetryMultiplier 2

# 21 - Parameter Setup Script Commands

Parameter setup commands should be executed by the **MaxwellParameterSetup** module.

```
Set oModule = oDesign.GetModule("MaxwellParameterSetup")
oModule.CommandName <args>
```

## Conventions Used in the Parameter Chapter

<ParameterName>

Type: <string>

Name of a parameter.

<ParameterNameArray>

Type: Array of strings

An array of the names in a group of parameters.

<AssignmentObjects>

Type: Array of strings

An array of object names.

## General Parameter Setup Script Commands

Following are general script commands recognized by the **MaxwellParameterSetup** module:

- [DeleteParameters](#)
- [DeleteAllParameters](#)

- [RenameParameter](#)
- [ReassignParameter](#)

### Commands to Create and Edit Parameters

Following are script commands for creating and editing parameters that are recognized by the **MaxwellParameterSetup** module:

- [AssignForce](#)
- [EditForce](#)
- [AssignTorque](#)
- [EditTorque](#)
- [AssignMatrix](#)
- [EditMatrix](#)
- [DeleteReduceMatrix](#)
- [RenameReduceMatrix](#)
- [AddReduceOp](#)
- [DeleteReduceOp](#)
- [EditReduceOp](#)
- [RenameReduceOp](#)

## AddReduceOp

Adds a reduce matrix group.

UI Access	Right-click on the matrix and select either <b>Join in Series...</b> or <b>Join in Parallel...</b>		
Parameters	Name	Type	Description

	<MatrixName>	String	Name of parent matrix
	<ReduceMatrixName>	String	Name of reduce matrix
	<ReduceOpArray>	Array	Array("NAME:<GroupName>","Type:=", one of "Join in Series" or "Join in Parallel", "Sources:=", "<Source1>, <Source2>, ...")
<b>Return Value</b>	None		

<b>Python Syntax</b>	AddReduceOp (<MatrixName>, <ReduceMatrixName>, <ReduceOpArray>)
<b>Python Example</b>	<pre> oModule = oDesign.GetModule("MaxwellParameterSetup") oModule.AddReduceOp ("Matrix1", "ReduceMatrix1", ["NAME:JoinSeries1", "Type:=", "Join in Series", "Sources:=", "Current1,Current3" ]) </pre>

<b>VB Syntax</b>	AddReduceOp <MatrixName> <ReduceMatrixName> <ReduceOpArray>
<b>VB Example</b>	<pre> Set oModule = oDesign.GetModule("MaxwellParameterSetup") oModule.AddReduceOp "Matrix1", "ReduceMatrix1", Array("NAME:JoinSeries1", "Type:=", "Join in Series", "Sources:=", "Current1,Current3") oModule.AddReduceOp "Matrix1", "ReduceMatrix2", </pre>

	Array("NAME:JoinParallel1", "Type:=", "Join in Parallel", "Sources:=", "Current2,Current4")
--	------------------------------------------------------------------------------------------------

## AssignForce

Creates a force.

UI Access	Maxwell3D or Maxwell2D>Parameters>Assign>Force		
Parameters	Name <i>&lt;ForceArray&gt;</i>	Type Array	Description Array containing force data:  Array ("NAME:<ForceName>", "IsVirtual:=", <boolean>, "ReferenceCS:=", <string>, "Objects:=", <array of objects>)
Return Value	None		

Python Syntax	AssignForce (<ForceArray>)
Python Example	<pre>oModule = oDesign.GetModule("MaxwellParameterSetup") oModule.AssignForce(     [ "NAME:Force1",       "Reference CS:=" , "Global",       "Is Virtual:=" , True,</pre>

	<pre>"Objects:=" , ["Core"] ])</pre>
--	--------------------------------------

<b>VB Syntax</b>	AssignForce <ForceArray>
<b>VB Example</b>	<pre>oModule.AssignForce     Array("NAME:Force1",         "Is Virtual:=", true,         "Reference CS:=", "Global",         "Objects:=", Array("Box1"))</pre>

## [Beta] AssignLayoutForce

Creates a layout force.

<b>UI Access</b>	Maxwell3D>Parameters>Assign>Force from Layout						
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;ForceArray&gt;</td> <td>Array</td> <td>           Array containing force data:            Array ("NAME:&lt;ForceName&gt;",           "ReferenceCS:=", &lt;string&gt;,           &lt;array of nets and layers choices&gt;)         </td> </tr> </table>	Name	Type	Description	<ForceArray>	Array	Array containing force data: Array ("NAME:<ForceName>",           "ReferenceCS:=", <string>,           <array of nets and layers choices>)
Name	Type	Description					
<ForceArray>	Array	Array containing force data: Array ("NAME:<ForceName>",           "ReferenceCS:=", <string>,           <array of nets and layers choices>)					
<b>Return Value</b>	None						

<b>Python Syntax</b>	AssignLayoutForce (<ForceArray>)
<b>Python Example</b>	<pre>oModule = oDesign.GetModule("MaxwellParameterSetup") oModule.AssignLayoutForce(     [ "NAME:LayoutForce1",         "Reference CS:=" , "Global",         [ "NAME:NetsAndLayersChoices",             [                 "NAME:LC1_1",                 [                     "NAME:NetLayerSetMap",                     [                         "NAME:net1",                         "LayerSet:=" , ["Ground", "Top"]                     ],                     [                         "NAME:net2",                         "LayerSet:=" , ["Top"]                     ]                 ]             ]         ]     ] )</pre>

	]
	])

<b>VB Syntax</b>	AssignLayoutForce <ForceArray>
<b>VB Example</b>	<pre> oModule.AssignLayoutForce Array("NAME:LayoutForce4", "Reference CS:=", "Global", Array("NAME:NetsAndLayersChoices", Array("NAME:LC1_1", Array("NAME:NetLayerSetMap", Array("NAME:net1", "LayerSet:=", Array("Ground", "Top")), Array("NAME:net2", "LayerSet:=", Array("Top"))))) </pre>

## AssignMatrix

**Use:** Creates a matrix.

**Command:** Maxwell3D or Maxwell2D>Parameters>Assign>Matrix

**Syntax:** AssignMatrix <MatrixArray>

**Return Value:** None

*Parameters:*

<MatrixArray>

```

Array("NAME:<MatrixName>",
Array("NAME:MatrixEntry",
Array("NAME:MatrixEntry",
"Source:=", <string>,
"NumberOfTurns:=", <int>),

```

```
...
)
Array("NAME:MatrixGroup",
Array("NAME:MatrixGroup",
"GroupName:=", <string>,
"NumberOfBranches:=", <int>,
"Sources:=", <nameArray>),
...
))
```

***Example:***

```
oModule.AssignMatrix _
Array("NAME:Matrix1", _
Array("NAME:MatrixEntry", _
Array("NAME:MatrixEntry", _
"Source:=", "Current1", _
"NumberOfTurns:=", "1"),
Array("NAME:MatrixEntry", _
"Source:=", "Current3", _
"NumberOfTurns:=", "1")),
Array("NAME:MatrixGroup", _
```

```
Array("NAME:MatrixGroup", _  
"GroupName:=", "Group1", _  
"NumberOfBranches:=", "1", _  
"Sources:=", "Current1,Current3"))
```

## AssignTorque

**Use:** Creates a torque.

**Command:** Maxwell3D or Maxwell2D>Parameters>Assign>Torque

**Syntax:** AssignTorque <TorqueArray>

**Return Value:** None

*Parameters:*

<ForceArray>

```
Array("NAME:<TorqueName>",  
"Is Virtual:=", <bool>,  
"Coordinate System:=", <string>  
"Axis:=", <string>,  
"Is Positive:=", <bool>  
"Objects:=", <AssignmentObjects>)
```

**Example:**

```
oModule.AssignTorque _  
Array("NAME:Torque1", "Is Virtual:=", true, _  
"Coordinate System:=", "Global", "Axis:=", "Z", _
```

```
"Is Positive:=", true, "Objects:=", Array("Box3"))
```

## DeleteAllParameters

**Use:** Deletes all parameters.

**Command:** Maxwell3D or Maxwell2D>Parameters>Delete All

**Syntax:** DeleteAllParameters

**Return Value:** None

**Example:** oModule.DeleteAllParameters

## DeleteParameters

Deletes one or more specified parameters.

<b>UI Access</b>	Delete button in Maxwell List dialog box (Maxwell3D or Maxwell2D>List)								
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;NameArray&gt;</td><td>string</td><td>An array of parameter names</td></tr></table>			Name	Type	Description	<NameArray>	string	An array of parameter names
Name	Type	Description							
<NameArray>	string	An array of parameter names							
<b>Return Value</b>	None								

<b>Python Syntax</b>	DeleteParameters ([<NameArray>])
<b>Python Example</b>	oModule.DeleteParameters(["Force1", "Torque1"])

<b>VB Syntax</b>	DeleteParameters <NameArray>
------------------	------------------------------

<b>VB Example</b>	<code>oModule.DeleteParameters Array("Force1", "Torque1")</code>
-------------------	------------------------------------------------------------------

## DeleteReduceMatrix

Deletes a specified Reduce Matrix in the project.

<b>UI Access</b>	Right-click on the desired reduce matrix item in Program Manager, then select Delete		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;MatrixName&gt;</code>	String	The name of the parent matrix
<b>Return Value</b>	None		

<b>Python Syntax</b>	<code>DeleteReduceMatrix (&lt;MatrixName&gt;, &lt;ReduceMatrixName&gt;)</code>
<b>Python Example</b>	<code>oModule.DeleteReduceMatrix ("Matrix1", "ReduceMatrix1")</code>

<b>VB Syntax</b>	<code>DeleteReduceMatrix &lt;MatrixName&gt;, &lt;ReduceMatrixName&gt;</code>
<b>VB Example</b>	<code>oModule.DeleteReduceMatrix "Matrix1", "ReduceMatrix1"</code>

## DeleteReduceOp

Deletes a specified Reduce Matrix group in the project.

<b>UI Access</b>	Right-click on the desired reduce matrix group item in Program Manager, then select Delete		
<b>Parameters</b>	Name	Type	Description

	<MatrixName>	String	The name of the parent matrix
	<ReduceMatrixName>	String	The name of the reduce matrix
	<GroupName>	String	The name of the reduce matrix group
<b>Return Value</b>	None		

<b>Python Syntax</b>	DeleteReduceOp (<MatrixName>, <ReduceMatrixName>, <GroupName>)
<b>Python Example</b>	<code>oModule.DeleteReduceOp ("Matrix1", "ReduceMatrix1", "JoinSeries1")</code>

<b>VB Syntax</b>	DeleteReduceOp <MatrixName>, <ReduceMatrixName>, <GroupName>
<b>VB Example</b>	<code>oModule.DeleteReduceOp "Matrix1", "ReduceMatrix1", "JoinSeries1"</code>

## EditForce

**Use:** Edits a force parameter.

**Command:** Double-click the parameter in the project tree to edit it.

**Syntax:** EditForce <ParameterName>, <ForceArray>

**Return Value:** None

## [Beta] EditLayoutForce

Creates a layout force.

<b>UI Access</b>	Right-click the desired Layout Force Parameter in the project manager and select <b>Properties...</b> to open the <b>Layout Force Setup</b> dialog for editing.								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;ForceArray&gt;</td> <td>Array</td> <td>           Array containing force data:              Array ("NAME:&lt;ForceName&gt;",            "ReferenceCS:=", &lt;string&gt;,            &lt;array of nets and layers choices&gt;)         </td> </tr> </tbody> </table>			Name	Type	Description	<ForceArray>	Array	Array containing force data:  Array ("NAME:<ForceName>", "ReferenceCS:=", <string>, <array of nets and layers choices>)
Name	Type	Description							
<ForceArray>	Array	Array containing force data:  Array ("NAME:<ForceName>", "ReferenceCS:=", <string>, <array of nets and layers choices>)							
<b>Return Value</b>	None								

<b>Python Syntax</b>	EditLayoutForce (<ForceArray>)
<b>Python Example</b>	<pre> oModule = oDesign.GetModule("MaxwellParameterSetup") oModule.EditLayoutForce(     [ "NAME:LayoutForce4",       "Reference CS:=" , "Global",       [ "NAME:NetsAndLayersChoices",         [           "NAME:LC1_1",           [             "NAME:NetLayerSetMap",             [               "NAME:net1", </pre>

```
    "LayerSet:=" , ["Ground"]  
] ,  
[  
    "NAME:net2",  
    "LayerSet:=" , ["Top"]  
]  
]  
]  
]  
]  
])
```

<b>VB Syntax</b>	EditLayoutForce <ForceArray>
<b>VB Example</b>	<pre>Set oModule = oDesign.GetModule("MaxwellParameterSetup") oModule.EditLayoutForce  Array("NAME:LayoutForce4", "Reference CS:=", "Global", Array("NAME:NetsAndLayersChoices", Array("NAME:LC1_1", Array("NAME:NetLayerSetMap", Array("NAME:net1", "LayerSet:=", Array("Ground"))), Array("NAME:net2", "LayerSet:=", Array("Top"))))))</pre>

## EditMatrix

**Use:** Edits a matrix parameter.

**Command:** Double-click the parameter in the project tree to edit it.

**Syntax:** EditMatrix <ParameterName>, <MatrixArray>

**Return Value:** None

## EditReduceOp

Edits a reduce matrix group.

<b>UI Access</b>	Right-click on the reduce matrix group and select <b>Properties....</b>															
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;MatrixName&gt;</td> <td>String</td> <td>Name of parent matrix</td> </tr> <tr> <td>&lt;ReduceMatrixName&gt;</td> <td>String</td> <td>Name of reduce matrix</td> </tr> <tr> <td>&lt;GroupName&gt;</td> <td>String</td> <td>Current name of the reduce matrix group</td> </tr> <tr> <td>&lt;ReduceOpArray&gt;</td> <td>Array</td> <td>Array("NAME:&lt;NewGroupName&gt;", "Type:=", one of "Join in Series" or "Join in Parallel", "Sources:=", "&lt;Source1&gt;, &lt;Source2, ...")</td> </tr> </tbody> </table>	Name	Type	Description	<MatrixName>	String	Name of parent matrix	<ReduceMatrixName>	String	Name of reduce matrix	<GroupName>	String	Current name of the reduce matrix group	<ReduceOpArray>	Array	Array("NAME:<NewGroupName>", "Type:=", one of "Join in Series" or "Join in Parallel", "Sources:=", "<Source1>, <Source2, ...")
Name	Type	Description														
<MatrixName>	String	Name of parent matrix														
<ReduceMatrixName>	String	Name of reduce matrix														
<GroupName>	String	Current name of the reduce matrix group														
<ReduceOpArray>	Array	Array("NAME:<NewGroupName>", "Type:=", one of "Join in Series" or "Join in Parallel", "Sources:=", "<Source1>, <Source2, ...")														
<b>Return Value</b>	None															

<b>Python Syntax</b>	EditReduceOp (<MatrixName>, <ReduceMatrixName>, <GroupName>, <ReduceOpArray>)
<b>Python Example</b>	<pre> oModule = oDesign.GetModule("MaxwellParameterSetup") oModule.EditReduceOp ("Matrix1", "ReduceMatrix1", "Group1" [ "NAME:NewSeriesGroup", "Type:=", "Join in Series", </pre>

```
"Sources:=", "Current1,Current3"  
])
```

<b>VB Syntax</b>	EditReduceOp <MatrixName>, <ReduceMatrixName>, <GroupName>, <ReduceOpArray>
<b>VB Example</b>	Set oModule = oDesign.GetModule("MaxwellParameterSetup") oModule.EditReduceOp "Matrix1", "ReduceMatrix1", "Group1" Array("NAME:NewSeriesGroup", "Type:=", "Join in Series", "Sources:=", "Current1,Current3")

## EditTorque

**Use:** Edits a torque parameter.

**Command:** Double-click the parameter in the project tree to edit it.

**Syntax:** EditTorque <ParameterName>, <TorqueArray>

**Return Value:** None

## ReassignParameter

**Use:** Specifies a new geometry assignment for a parameter.

**Command:** Maxwell3D or Maxwell2D>Parameters>Reassign

**Syntax:** ReassignParameter Array("Name:<ParameterName>", "Objects:=", <AssignmentObjects>)

**Return Value:** None

**Example:** oModule.ReassignParameter Array("NAME:Force1", \_  
"Objects:=", Array("Box2"))

## RenameParameter

**Use:** Renames a parameter.

**Command:** Right-click the parameter item in the project tree, and click **Rename**.

**Syntax:** RenameParameter <OldName>, <NewName>

**Return Value:** None

*Parameters:*

<OldName>

Type: <string>

<NewName>

Type: <string>

**Example:** oModule.RenameParameter "Force1", "test"

## RenameReduceMatrix

Renames a specified Reduce Matrix in the project.

<b>UI Access</b>	Right-click on the desired reduce matrix item in Program Manager, then select Rename		
<b>Parameters</b>	Name	Type	Description
	<MatrixName>	String	The name of the parent matrix
	<OldReduceMatrixName>	String	The old name of the reduce matrix
	<NewReduceMatrixName>	String	The new name of the reduce matrix
<b>Return Value</b>	None		

<b>Python Syntax</b>	RenameReduceMatrix (<MatrixName>, <OldReduceMatrixName>, <NewReduceMatrixName>)
<b>Python Example</b>	oModule.RenameReduceMatrix ("Matrix1", "ReduceMatrix1", "ReduceMatrix2")

<b>VB Syntax</b>	RenameReduceMatrix <MatrixName>, <OldReduceMatrixName>, <NewReduceMatrixName>
<b>VB Example</b>	oModule.RenameReduceMatrix "Matrix1", "ReduceMatrix1", "ReduceMatrix2"

## RenameReduceOp

Renames a specified Reduce Matrix group in the project.

<b>UI Access</b>	Right-click on the desired reduce matrix group item in Program Manager, then select Rename															
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;MatrixName&gt;</td><td>String</td><td>The name of the parent matrix</td></tr><tr><td>&lt;ReduceMatrixName&gt;</td><td>String</td><td>The name of the reduce matrix</td></tr><tr><td>&lt;OldGroupName&gt;</td><td>String</td><td>The old name of the reduce matrix group</td></tr><tr><td>&lt;NewGroupName&gt;</td><td>String</td><td>The new name of the reduce matrix group</td></tr></tbody></table>	Name	Type	Description	<MatrixName>	String	The name of the parent matrix	<ReduceMatrixName>	String	The name of the reduce matrix	<OldGroupName>	String	The old name of the reduce matrix group	<NewGroupName>	String	The new name of the reduce matrix group
Name	Type	Description														
<MatrixName>	String	The name of the parent matrix														
<ReduceMatrixName>	String	The name of the reduce matrix														
<OldGroupName>	String	The old name of the reduce matrix group														
<NewGroupName>	String	The new name of the reduce matrix group														
<b>Return Value</b>	None															

<b>Python Syntax</b>	RenameReduceOp (<MatrixName>, <ReduceMatrixName>, <OldGroupName>, <NewGroupName>)
<b>Python Example</b>	oModule.RenameReduceOp ("Matrix1", "ReduceMatrix1", "Series1", "Series2")

<b>VB Syntax</b>	RenameReduceOp <MatrixName>, <ReduceMatrixName>, <OldGroupName>, <NewGroupName>
<b>VB Example</b>	oModule.RenameReduceOp "Matrix1", "ReduceMatrix1", "Series1", "Series2"

This page intentionally  
left blank.

## 22 - User Defined Document Script Commands

The product has to implement the [GetModule](#) call to create the UserDefinedDocument scripting object (e.g., Check AltraSimDesign.cpp (function GetMgrIDispatch())). To access the UserDefinedDocuments scripting object, use:

```
Set oModule = oDesign.GetModule("UserDefinedDocuments")
```

Once you have the scripting object, you can use the following methods:

- [AddDocument](#)
- [EditDocument](#)
- [RenameDocument](#)
- [DeleteDocument](#)
- [UpdateDocument](#)
- [ViewHtmlDocument](#)
- [ViewPdfDocument](#)
- [SaveHtmlDocumentAs](#)
- [SavePdfDocumentAs](#)
- [GetDocumentDefinitionNames](#)
- [DeleteAllDocuments](#)
- [UpdateAllDocuments](#)

You can find examples of how to use these methods on each of the methods' pages. A complete [example of a Python script](#) is also available, as is an [example with a line by line explanation](#).

### AddDocument

Creates a new document based on provided data and traces. The document names come from UserDefinedDocument folder under syslib, userlib, and personallib. This creates a document and places it in the Project Manager under **Results > Documents**.

UI Access	Right click on <b>Results &gt; Create Document</b> . Choose a document name.		
Parameters	Name	Type	Description
	<data>	Array	Data that defines the document.
Return Value	None		

Python Syntax	AddDocument (<data>, <traces>)
Python Example	<pre>oModule.AddDocument (     [         "NAME:Design Summary",         "",         "SysLib",         "DesignSummary",         [ "NAME:Inputs" ]     ],     [ "NAME:DocTraces" ] )</pre>

VB Syn- tax	AddDocument <data>, <traces>
----------------	------------------------------

**VB Example**

In this example, the document names come from the UserDefinedDocument folder in the syslib, userlib, and personallib folders. This creates a document and places it in the Documents folder under results.

```
oModule.AddDocument _
    Array("NAME:Design Summary", _
        "", "SysLib", "DesignSummary", _
        Array("NAME:Inputs")), _
    Array("NAME:DocTraces")
```

The following example is explained in [Explication of a Sample UDD Script](#).

```
oModule.AddDocument Array("NAME:Test Report1", "Test Report", "SysLib", _
    "Examples/TestUDDInputs", Array("NAME:Inputs", Array("NAME:DLMetrics", "Solution", _
        "Data Line Metrics", -1, -1), Array("NAME:DQ0", "Trace", "DQ0", -1, -1), _
        Array("NAME:DQS", "Trace", "DQS", -1, -1), _
        Array("NAME:Name", "Text", "User Name", Array("Sita Ramesh")), _
        Array("NAME:Summary", "Bool", "Display Summary", Array(true)), _
        Array("NAME:Version", "Number", "Script Version"))), _
    Array("NAME:DocTraces", Array("NAME:DLMetrics", _
        Array("User Defined", "", "DDR3 AC-Timing 4-DQ1", Array("Context:=", ""), _
            Array("Index:=", Array("All"), "Trise:=", Array("Nominal"), "Tfall:=", _
                Array("Nominal"), "Pulse_Width:=", Array("Nominal")), _
                "Data_Rate:=", Array("Nominal"), "Length:=", Array("Nominal"))), _
            Array("Probe Component:=", Array(""), Array()), _
            Array("NAME:DQ0", Array( _
```

```
"Standard", "DQ0", "NexximTransient", Array("NAME:Context", "SimValueContext:=", Array(_  
1, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0, _  
"DE", false, "0", "DP", false, "20000000", "DT", false, "0.001", "WE", _  
false, "100ns", "WM", false, "100ns", "WN", false, "0ps", "WS", false, "0ps")), _  
Array("Time:=", Array("All"), "Trise:=", Array("Nominal"), "Tfall:=", Array("Nominal"), _  
"Pulse_Width:=", Array("Nominal"), "Data_Rate:=", Array("Nominal"), _  
"Length:=", Array("Nominal")), Array("Probe Component:=", Array("DQ0")), Array())))
```

## DeleteAllDocuments

Deletes all documents in the object.

<b>UI Access</b>	Right click on <b>Documents</b> in the Project Manager and click <b>Delete All Documents</b> .
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	DeleteAllDocuments()
<b>Python Example</b>	oModule.DeleteAllDocuments ()

<b>VB Syntax</b>	DeleteAllDocuments
------------------	--------------------

<b>VB Example</b>	<code>oModule.DeleteAllDocuments</code>
-------------------	-----------------------------------------

## DeleteDocument

Deletes a specified document.

<b>UI Access</b>	Right click on the created document in the Project Manager under <b>Results &gt; Documents</b> and click <b>Delete</b> .						
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td><code>&lt;name&gt;</code></td> <td>String</td> <td>Name of the document to be deleted.</td> </tr> </table>	Name	Type	Description	<code>&lt;name&gt;</code>	String	Name of the document to be deleted.
Name	Type	Description					
<code>&lt;name&gt;</code>	String	Name of the document to be deleted.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	<code>DeleteDocument(&lt;names&gt;)</code>
<b>Python Example</b>	<code>oModule.DeleteDocument ("Project Design Summary")</code>

<b>VB Syntax</b>	<code>DeleteDocument &lt;names&gt;</code>
<b>VB Example</b>	<code>oModule.DeleteDocument "Project Design Summary"</code>

## EditDocument

Edits specified documents. If the document pops up a dialog box, the user can make a change the inputs for the document. The document is regenerated and updated. A new one is *not* created.

<b>UI Access</b>	Right click on the created document in the Project Manager under <b>Results &gt; Documents</b> and click <b>Modify document</b> .
------------------	-----------------------------------------------------------------------------------------------------------------------------------

Parameters	Name	Type	Description
	<originalName>	String	Name of the original document
	<modifiedData>	Array	New data to add to or modify the document
	<modifiedTraces>	Array	Trace data for the inputs of the document
Return Value	None.		

<b>Python Syntax</b>	EditDocument( <name>,<data>,<traces> )
<b>Python Example</b>	<pre>oModule.EditDocument("Design Summary", [     "NAME:Design Summary",     "",     "SysLib",     "DesignSummary",     ["NAME:Inputs"] ], ["NAME:DocTraces"] )</pre>

<b>VB Syntax</b>	EditDocument <name> , <data> , <traces>
------------------	-----------------------------------------

<b>VB Example</b>	<pre> oModule.EditDocument "Design Summary", _ Array("NAME:Design Summary", "", "SysLib", _ "DesignSummary", Array("NAME:Inputs")), Array("NAME:DocTraces") </pre>
-------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------

## GetDocumentDefinitionNames

Document definition names are the list of names that can be used to create a document. They appear when you click on **Create document**. This method returns the filenames of document definitions according to the files in the installation directories:

- syslib/UserDefinedDocuments
- userlib/UserDefinedDocuments
- personallib/UserDefinedDocuments

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;separator&gt;</td> <td>String</td> <td>Separator used to convey the directory "level"</td> </tr> </tbody> </table>	Name	Type	Description	<separator>	String	Separator used to convey the directory "level"
Name	Type	Description					
<separator>	String	Separator used to convey the directory "level"					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	GetDocumentDefinitionNames(<separator>)
<b>Python Example</b>	<code>oModule.GetDocumentDefinitionNames ("")</code>

<b>VB Syntax</b>	GetDocumentDefinitionNames <separator>
<b>VB Example</b>	<code>oModule.GetDocumentDefinitionNames "</code>

## GetDocumentNames

Retrieves the names for all documents.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	Array of strings containing document names.

<b>Python Syntax</b>	GetDocumentNames()
<b>Python Example</b>	<code>oModule.GetDocumentNames ()</code>

<b>VB Syntax</b>	GetDocumentNames
<b>VB Example</b>	<code>oModule.GetDocumentNames</code>

## RenameDocument

Changes the name of a document.

<b>UI Access</b>	Right click on the created document in the Project Manager under <b>Results&gt; Documents</b> and click <b>Rename</b> .											
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><code>&lt;oldName&gt;</code></td><td>String</td><td>Current name of the document</td></tr><tr><td><code>&lt;newName&gt;</code></td><td>String</td><td>New name of the document</td></tr></tbody></table>			Name	Type	Description	<code>&lt;oldName&gt;</code>	String	Current name of the document	<code>&lt;newName&gt;</code>	String	New name of the document
Name	Type	Description										
<code>&lt;oldName&gt;</code>	String	Current name of the document										
<code>&lt;newName&gt;</code>	String	New name of the document										

<b>Return Value</b>	None.
---------------------	-------

<b>Python Syntax</b>	RenameDocument(<oldName>, <newName>)
<b>Python Example</b>	<code>oModule.RenameDocument ("Design Summary", "Project Design Summary")</code>

<b>VB Syntax</b>	RenameDocument <oldName>, <newName>
<b>VB Example</b>	<code>oModule.RenameDocument "Design Summary", "Project Design Summary"</code>

## SaveHtmlDocumentAs

Saves a pre-existing HTML file to a different name and/or location.

<b>UI Access</b>	Right click on the created document in the Project Manager under <b>Results &gt; Documents</b> and click <b>Save As &gt; HTML</b> .									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;name&gt;</td> <td>String</td> <td>Name of the document to be saved.</td> </tr> <tr> <td>&lt;saveTo&gt;</td> <td>String</td> <td>File path to save the document to.</td> </tr> </tbody> </table>	Name	Type	Description	<name>	String	Name of the document to be saved.	<saveTo>	String	File path to save the document to.
Name	Type	Description								
<name>	String	Name of the document to be saved.								
<saveTo>	String	File path to save the document to.								
<b>Return Value</b>	none									

<b>Python Syntax</b>	SaveHtmlDocumentAs(<name>, <saveTo>)
<b>Python Example</b>	<code>oModule.SaveHtmlDocumentAs ("Design Summary 1", "DS1.html")</code>

<b>VB Syntax</b>	SaveHtmlDocumentAs <name> <saveTo>
<b>VB Example</b>	oModule.SaveHtmlDocumentAs "Design Summary 1" "DS1.html"

## SavePdfDocumentAs

Saves a pre-existing PDF file to a different name and/or location.

<b>UI Access</b>	Right click on the created document in the Project Manager under <b>Results &gt; Documents</b> and click <b>Save As &gt; PDF</b> .									
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;name&gt;</td><td>String</td><td>Name of the document to be saved.</td></tr><tr><td>&lt;saveTo&gt;</td><td>String</td><td>File path to save the document at.</td></tr></tbody></table>	Name	Type	Description	<name>	String	Name of the document to be saved.	<saveTo>	String	File path to save the document at.
Name	Type	Description								
<name>	String	Name of the document to be saved.								
<saveTo>	String	File path to save the document at.								
<b>Return Value</b>	None.									

<b>Python Syntax</b>	SavePdfDocumentAs(<name>, <saveTo>)
<b>Python Example</b>	oModule.SavePdfDocumentAs("Design Summary 1", "DS1.pdf")

<b>VB Syntax</b>	SavePdfDocumentAs <name>, <saveTo>
<b>VB Example</b>	oModule.SavePdfDocumentAs "Design Summary 1", "DS1.pdf"

## UpdateAllDocuments

Refreshes the contents of all created documents. This action is made on the folder rather than the individual document.

---

<b>UI Access</b>	Right click on <b>Results &gt; Documents</b> in the Project Manager and click <b>Update All Documents</b> .
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	<code>UpdateAllDocuments()</code>
<b>Python Example</b>	<code>oModule.UpdateAllDocuments ()</code>

<b>VB Syntax</b>	<code>UpdateAllDocuments</code>
<b>VB Example</b>	<code>oModule.UpdateAllDocuments</code>

## UpdateDocument

Refreshes the contents of the selected document.

<b>UI Access</b>	Right click on the created document in the Project Manager under <b>Results &gt; Documents</b> and click <b>Update Document</b> .		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;name&gt;</code>	String	Name of the document to be updated
<b>Return Value</b>	None.		

<b>Python Syntax</b>	UpdateDocument(<name>)
<b>Python Example</b>	<code>oModule.UpdateDocument ("Test UDD Report")</code>

<b>VB Syntax</b>	UpdateDocument <name>
<b>VB Example</b>	<code>oModule.UpdateDocument "Test UDD Report"</code>

## ViewHtmlDocument

Displays a pre-existing document as HTML.

<b>UI Access</b>	Right click on the created document in the Project Manager under <b>Results &gt; Documents</b> and click <b>View Xml Document</b> .								
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;name&gt;</td><td>String</td><td>Name of the document to be viewed as a HTML</td></tr></table>			Name	Type	Description	<name>	String	Name of the document to be viewed as a HTML
Name	Type	Description							
<name>	String	Name of the document to be viewed as a HTML							
<b>Return Value</b>	None								

<b>Python Syntax</b>	ViewHtmlDocument(<name>)
<b>Python Example</b>	<code>oModule.ViewHtmlDocument ("Design Summary 1")</code>

<b>VB Syntax</b>	ViewHtmlDocument <name>
<b>VB Example</b>	<code>oModule.ViewHtmlDocument "Design Summary 1"</code>

## ViewPdfDocument

Displays a pre-existing document as a PDF file.

<b>UI Access</b>	Right click on the created document in the Project Manager under <b>Results &gt; Documents</b> and click <b>View PDF Document</b> .								
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;name&gt;</td> <td>String</td> <td>Name of the document to be viewed as a PDF</td> </tr> </table>			Name	Type	Description	<name>	String	Name of the document to be viewed as a PDF
Name	Type	Description							
<name>	String	Name of the document to be viewed as a PDF							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	ViewPdfDocument(<name>)
<b>Python Example</b>	oModule.ViewPdfDocument ("Design Summary 1")

<b>VB Syntax</b>	ViewPdfDocument <name>
<b>VB Example</b>	oModule.ViewPdfDocument "Design Summary 1"

## Explication of a Sample UDD Script

This VB script defines a document. It is a portion of one of the UDD example VB scripts.

```
Array("NAME:Test Report",           ' Name of the document
      "Test Report",                 ' Description of the document
      "SysLib",                      ' Location of the python script: Syslib, Userlib, PeronalLib, etc.
      "TestUDDReport",                ' Relative path of the script in the UserDefinedDocuments folder
```

' This array is the start of the input definition.

```
Array("NAME:Inputs",  
      ' Document Inputs keyword  
  
'This array contains the Solution input.  
Array("NAME:DLMetrics",  
      "Solution",  
      "Data Line Metrics",  
      -1,  
      -1),  
      ' Input name  
      ' Solution Input Type  
      ' Input Description  
      ' Solution ID  
      ' Report ID  
  
'This array contains the trace input.  
Array("NAME:DQ0",  
      "Trace",  
      "DQ0",  
      -1,  
      -1),  
      ' Input name  
      ' Trace Input Type  
      ' Input Description  
      ' Solution ID  
      ' Report ID  
  
'This array contains the text input.  
Array("NAME:Name",  
      "Text",  
      "User Name",  
      Array("Sita Ramesh")),  
      ' Input name  
      ' Text Input Type  
      ' Input Description  
      ' Default Value  
  
'This array contains the Bool input.  
Array("NAME:Summary",  
      "Bool",  
      "Display Summary",  
      Array(true)),  
      ' Input name  
      ' Boolean Input Type  
      ' Input Description  
      ' Default Value  
  
'This array contains the number input.
```

```

Array("NAME:Version",
      ' Input name
      "Number",
      ' Number Input Type
      "Script Version",
      ' Input Description
      Array(1021))),           ' Default Value

'This array contains trace selection for the solution and trace inputs.
Array("NAME:DocTraces",           ' Document traces keyword

'This array has input for "DLMetrics".
Array("NAME:DLMetrics",           ' Input name

'This array defines a trace similar to the UDO. This trace definition is a User defined solution
Array("User Defined", "", "DDR3 AC-Timing 4-DQ1", Array("Context:=", ""), Array("Index:=", Array
("All"), "Trise:=", Array("Nominal"), "Tfall:=", Array("Nominal"), "Pulse_Width:=", Array("Nom-
inal"), "Data_Rate:=", Array("Nominal"), "Length:=", Array("Nominal")), Array("Probe Com-
ponent:=", Array(""))), Array()))

'This array is for input "DQ0".
Array("NAME:DQ0",

'This array defines a trace similar to the UDO. This trace definition is a Standard solution.
Array("Standard", "DQ0", "NexximTransient", Array("NAME:Context", "SimValueContext:=", Array(1,
0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0, "DE", false, "0", "DP",
false, "20000000", "DT", false, "0.001", "WE", false, "100ns", "WM", false,
"100ns", "WN", false, "0ps", "WS", false, "0ps")), Array("Time:=", Array("All"), "Trise:=",
Array(
"Nominal"), "Tfall:=", Array("Nominal"), "Pulse_Width:=", Array("Nominal"), "Data_Rate:=", Array
("Nominal"), "Length:=", Array("Nominal")), Array("Probe Component:=", Array(
"DQ0")), Array())))

```

## Example Python Script: Defining a Document

This script creates a user-defined solution and a document.

```
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("BJT_Inverter")
oDesign = oProject.SetActiveDesign("Nexxim1")
oModule = oDesign.GetModule("UserDefinedSolutionModule")

oModule.CreateUserDefinedSolution("UDS Distance Trace Arithmetic Result1", "SysLib", "TraceArith-
metic/Distance Sweep Trace Arithmetic",
[
    "Offset 1:=", "0",
    "Scale 1:=", "1",
    "Offset 2:=", "0",
    "Scale 2:=", "1",
    "Operation:=", "Add"
],
[
[
    [
        "Standard",
        "probe1",
        "Transient",
        [
            style="font-family: monospace;">"NAME:Context",
            style="font-family: monospace;">"SimValueContext:=", [1,0,2,0,False,False,-
1,1,0,1,1,"",0,0,"DE",False,"0","DP",False,"500000000","DT",False,"0.001","NUMLEVELS",False,"0","-
WE",False,"10us","WM",False,"10us","WN",False,"0ns","WS",False,"0ns"]
        ],
        [
            "Time:=", ["All"]
        ],
        [
            "Probe Component:=", ["V(Port1)"]
        ]
    ]
]
```

```
        ],
        []
    ],
    [
        "Standard",
        "probe2",
        "Transient",
        [
            "NAME:Context",
            "SimValueContext:=", [1,0,2,0,False,False,-
1,1,0,1,1,"",0,0,"DE",False,"0","DP",False,"500000000","DT",False,"0.001","NUMLEVELS",False,"0",-"
"WE",False,"10us","WM",False,"10us","WN",False,"0ns","WS",False,"0ns"]
        ],
        [
            "Time:=", ["All"]
        ],
        [
            "Probe Component:=", ["V(Port1)"]
        ],
        []
    ]
],
[])
oModule = oDesign.GetModule("UserDefinedDocuments")
oModule.AddDocument(
[
    "NAME:Test Report",
    "Test Report",
    "SysLib",
    "TestUDDInputs",
    [
        "NAME:Inputs",
        [
            "NAME:UDS1",

```

```
        "Solution",
        "UDS Distance Trace Arithmetic Result1",
        1000000,
        0
    ],
    [
        "NAME:UDS2",
        "Solution",
        "UDS Distance Trace Arithmetic Result2",
        1000000,
        2
    ]
],
[
    "NAME:DocTraces",
    [
        "NAME:UDS1",
        [
            "User Defined",
            "",
            "UDS Distance Trace Arithmetic Result1",
            [
                "Context:=", ""
            ],
            [
                "Distance:=", ["All"]
            ],
            [
                "Probe Component:=", []
            ],
            []
        ]
    ]
]
```

```
        ],
    ],
    [
        "NAME:UDS2",
        [
            "User Defined",
            "",
            "UDS Distance Trace Arithmetic Result1",
            [
                "Context:=", ""
            ],
            [
                "Distance:=", ["All"]
            ],
            [
                "Probe Component:=", []
            ],
            []
        ]
    ],
    []
)
])
```

This page intentionally  
left blank.

# 23 - User Defined Solutions Commands

User Defined Solution commands should be executed by the "UserDefinedSolutionModule" module.

```
Set oDesign = oProject.SetActiveDesign("TestDesign1")
Set oModule =
oDesign.GetModule("UserDefinedSolutionModule")
```

The list of commands is as follows:

[CreateUserDefinedSolution](#)

[DeleteUserDefinedSolutions](#)

[EditUserDefinedSolution](#)

## CreateUserDefinedSolution

Creates a new user defined solution.

UI Access	Right-click on <b>Results &gt; Create User Defined Solution</b> .		
Parameters	Name	Type	Description
	<SoluName>	String	Name of user defined solution.
	<LibType>	String	Indicates the library where the UDS plugin file is located. This parameter must be one of the following values: "SysLib", "UserLib", "PersonalLib".
	<RelativePath>	String	The path of the UDS plugin file relative to the "UserDefinedOutputs" sub-directory of the library specified by <LibType>.
	<PropList>	Array	Strings specify name-value pairs corresponding to the UDS properties specified in the plugin file.

		<p>For example:</p> <pre>Array("multiply_factor:=", "2.0", "component_name:=",       "resistor1")</pre>
<ProbeSelections>	Array	Name of the probe being specified. Note: this must match a probe name specified in the UDS plugin file.
<DynamicProbes>	Array	Array of <ProbeSelection>'s, representing the probes that are used by dynamic-probes.
<b>Return Value</b>	String name of created user defined solution.	

<b>Python Syntax</b>	CreateUserDefinedSolution(<SoluName>, <LibType>, <RelativePath>, <PropList>, <ProbeSelections>, <DynamicProbes>)
<b>Python Example</b>	<pre>oModule.CreateUserDefinedSolution (     "ConstantTimestep1", "SysLib",     "ConstantTimestep", [], [], [])</pre>

<b>VB Syntax</b>	CreateUserDefinedSolution <SoluName>, <LibType>, <RelativePath>, <PropList>, <ProbeSelections>, <DynamicProbes>
<b>VB Example</b>	<pre>oModule.CreateUserDefinedSolution "User Defined Solution 1", _     "SysLib", "Example", _     Array("multiplication_factor:=", "1.2"), _     Array(Array("Modal Solution Data",         "Probe 1", "Setup1 : LastAdaptive", _</pre>

```

Array(), Array("Freq:=", Array( "All")), _
Array("Probe Component:=", Array("dB(S(1,1))"), Array()), _
Array( "Modal Solution Data", "Probe 2",
"Setup1 : LastAdaptive", Array(), _
Array("Freq:=", Array( "All")), _
Array("Probe Component:=", Array("mag(S(1,1))"), Array()), _
Array(Array("Modal Solution Data", _
"Dynamic Probe 1", "Setup1 : LastAdaptive", Array(), _
Array("Freq:=", Array( "All")), _
Array("Probe Component:=", Array("Freq")), Array())))

```

## DeleteUserDefinedSolutions

Deletes one or more user defined solutions.

<b>UI Access</b>	Delete button from the <b>User Defined Solutions</b> dialog.						
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;SolNames&gt;</td> <td>Array</td> <td>Array of strings containing names of User Defined Solutions to be deleted.</td> </tr> </table>	Name	Type	Description	<SolNames>	Array	Array of strings containing names of User Defined Solutions to be deleted.
Name	Type	Description					
<SolNames>	Array	Array of strings containing names of User Defined Solutions to be deleted.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	DeleteUserDefinedSolutions(<SolNames>)
<b>Python Example</b>	oModule.DeleteUserDefinedSolutions(["Solution1", "Solution2"])

<b>VB Syntax</b>	DeleteUserDefinedSolutions <SolNames>
<b>VB Example</b>	<pre>oModule.DeleteUserDefinedSolutions _     Array("Solution1", "Solution2")</pre>

## EditUserDefinedSolution

Modifies an existing user defined solution.

<b>UI Access</b>	Edit button from the <b>User Defined Solutions</b> dialog box.																										
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;SolName&gt;</td> <td>String</td> <td>Name of user defined solution to be edited.</td> </tr> <tr> <td>&lt;NewSolName&gt;</td> <td>String</td> <td>New name for the specified user defined solution.</td> </tr> <tr> <td>&lt;LibType&gt;</td> <td>String</td> <td>Indicates the library where the UDS plugin file is located. This parameter must be one of the following values: "SysLib", "UserLib", "PersonalLib".</td> </tr> <tr> <td>&lt;RelativePath&gt;</td> <td>String</td> <td>The path of the UDS plugin file relative to the "UserDefinedOutputs" sub-directory of the library specified by &lt;LibType&gt;.</td> </tr> <tr> <td>&lt;PropList&gt;</td> <td>Array</td> <td>           Strings specify name-value pairs corresponding to the UDS properties specified in the plugin file.             For example:   <pre>Array("multiply_factor:=", "2.0", "component_name:=", "resistor1")</pre> </td> </tr> <tr> <td>&lt;ProbeSelections&gt;</td> <td>Array</td> <td>Name of the probe being specified. Note: this must match a probe name specified in the UDS plugin file.</td> </tr> <tr> <td>&lt;DynamicProbes&gt;</td> <td>Array</td> <td>Array of &lt;ProbeSelection&gt;'s, representing the probes that are used by dynamic-probes.</td> </tr> </table>			Name	Type	Description	<SolName>	String	Name of user defined solution to be edited.	<NewSolName>	String	New name for the specified user defined solution.	<LibType>	String	Indicates the library where the UDS plugin file is located. This parameter must be one of the following values: "SysLib", "UserLib", "PersonalLib".	<RelativePath>	String	The path of the UDS plugin file relative to the "UserDefinedOutputs" sub-directory of the library specified by <LibType>.	<PropList>	Array	Strings specify name-value pairs corresponding to the UDS properties specified in the plugin file.  For example:  <pre>Array("multiply_factor:=", "2.0", "component_name:=", "resistor1")</pre>	<ProbeSelections>	Array	Name of the probe being specified. Note: this must match a probe name specified in the UDS plugin file.	<DynamicProbes>	Array	Array of <ProbeSelection>'s, representing the probes that are used by dynamic-probes.
Name	Type	Description																									
<SolName>	String	Name of user defined solution to be edited.																									
<NewSolName>	String	New name for the specified user defined solution.																									
<LibType>	String	Indicates the library where the UDS plugin file is located. This parameter must be one of the following values: "SysLib", "UserLib", "PersonalLib".																									
<RelativePath>	String	The path of the UDS plugin file relative to the "UserDefinedOutputs" sub-directory of the library specified by <LibType>.																									
<PropList>	Array	Strings specify name-value pairs corresponding to the UDS properties specified in the plugin file.  For example:  <pre>Array("multiply_factor:=", "2.0", "component_name:=", "resistor1")</pre>																									
<ProbeSelections>	Array	Name of the probe being specified. Note: this must match a probe name specified in the UDS plugin file.																									
<DynamicProbes>	Array	Array of <ProbeSelection>'s, representing the probes that are used by dynamic-probes.																									
<b>Return Value</b>	String name of update user defined solution.																										

<b>Python Syntax</b>	EditUserDefinedSolution(<SolName>, <NewSolName>, <LibType>, <RelativePath>, <PropList>, <ProbeSelections>, <DynamicProbes>)
<b>Python Example</b>	<pre> oModule.EditUserDefinedSolution("ConstantTimestep1" "ConstantTimestep1After", "SysLib", "ConstantTimestep", [], [], []) </pre>

<b>VB Syntax</b>	EditUserDefinedSolution <SolName>, <NewSolName>, <LibType>, <RelativePath>, <PropList>, <ProbeSelections>, <DynamicProbes>
<b>VB Example</b>	<pre> oModule.CreateUserDefinedSolution "User Defined Solution 1", _ "User Defined Solution 1", "SysLib", "Example", _ Array("multiplication_factor:=", "1.2"), _ Array(Array("Modal Solution Data", "Probe 1", "Setup1 : LastAdaptive", _ Array(), Array("Freq:=", Array( "All"))), _ Array("Probe Component:=", Array("dB(S(1,1))))), Array(), _ Array( "Modal Solution Data", "Probe 2", "Setup1 : LastAdaptive", Array(), _ Array("Freq:=", Array( "All")), _ Array("Probe Component:=", Array("mag(S(1,1)))), Array()), _ Array(Array("Modal Solution Data", _ "Dynamic Probe 1", "Setup1 : LastAdaptive", Array(), _ </pre>

	<pre>Array("Freq:=", Array( "All")), _ Array("Probe Component:=", Array("Freq")), Array()))</pre>
--	-------------------------------------------------------------------------------------------------------

# 24 - Network Data Explorer Script Commands

Network Data Explorer (NDE) scripting uses three objects, each with unique script commands:

- **Network Data Explorer** – top-level object obtained by calling `oNDE=oDesktop.GetTool("ndExplorer")`. Network Data Explorer commands are called using `oNDE`.
- **Network Data** – single set of S-parameters, corresponding to a single entry in the UI tree. Network Data commands are called using `oData`.
- **Post Process Settings** – settings that can be applied and removed from network data without making permanent changes to the underlying data. Post Process Settings commands are called using `oPostProc`.

Examples using the above objects:

```
oNDE = oDesktop.GetTool("ndExplorer")
oData = oNDE.Open("D:\folder\test.s2p")
success = oPostProc.AddDiffPair(2, 1, "Diff1", "Comm1", 100, 25)
```

The following commands are described in this section:

**Warning: The following commands are preliminary. Their syntax may change. Be prepared to make changes to legacy scripts.**

[AddDiffPair](#)

[Cascade \(SPISim\)](#)

[ClearDiffPairs](#)

[Clone](#)

[Close](#)

[Combine \(SPISim\)](#)  
[Deembed \(SPISim\)](#)  
[DeembedBack \(SPISim\)](#)  
[DeembedFront \(SPISim\)](#)  
[DisableDiffPairs](#)  
[EnableDiffPairs](#)  
[ExportCitiFile](#)  
[ExportFullWaveSpice](#)  
[ExportMatlab](#)  
[ExportNMFDATA](#)  
[ExportNetworkData](#)  
[ExportSpreadsheet](#)  
[ExportTouchstone](#)  
[ExportTouchstone2](#)  
[Extract \(SPISim\)](#)  
[GetFrequencies](#)  
[GetFrequencyCount](#)  
[GetName](#)  
[GetPortCount](#)  
[GetPortNumber](#)

---

[GetPostProcSettings](#)

[GetSolutionVariation](#)

[GetVariation](#)

[HasSameData](#)

[LoadSolution](#)

[Open](#)

[Rename \(SPISim\)](#)

[Renormalize \(SPISim\)](#)

[Reorder](#)

[Reorder \(SPISim\)](#)

[Reset](#)

[SetAllPortImpedances](#)

[SetAllPortImpedances](#)

[SetPortDeembedDistance](#)

[SetPortImpedance](#)

[SetPostProcSettings](#)

[Smooth](#)

[Stretch \(SPISim\)](#)

[Terminate](#)

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## AddDiffPair

Specifies a differential pair from two terminal ports.

UI Access	From the <b>Network Data Explorer</b> tab, select <b>Differential Pairs</b> in the <b>NDE</b> ribbon to open the <b>Differential Pairs</b> window. Then select differential pairs from the <b>Pairs</b> group box and click <b>Add Pairs &gt;&gt;</b> .																										
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>&lt;positiveTerminal&gt;</code></td> <td>Integer</td> <td>The portNumber of the positive terminal.</td> </tr> <tr> <td><code>&lt;negativeTerminal&gt;</code></td> <td>Integer</td> <td>The portNumber of the negative terminal.</td> </tr> <tr> <td><code>&lt;diffName&gt;</code></td> <td>String</td> <td>The new name of the differential pin (e.g., Diff1).</td> </tr> <tr> <td><code>&lt;commName&gt;</code></td> <td>String</td> <td>The new name of the common pin (e.g., Comm1).</td> </tr> <tr> <td><code>&lt;diffImpedance&gt;</code></td> <td>Double</td> <td>The differential pin characteristic impedance.</td> </tr> <tr> <td><code>&lt;commImpedance&gt;</code></td> <td>Double</td> <td>The common pin characteristic impedance.</td> </tr> <tr> <td><code>&lt;matched&gt;</code></td> <td>Boolean</td> <td>Specify whether to use matched pair.  <b>Note:</b> The default value is <b>False</b>.</td> </tr> </tbody> </table>			Name	Type	Description	<code>&lt;positiveTerminal&gt;</code>	Integer	The portNumber of the positive terminal.	<code>&lt;negativeTerminal&gt;</code>	Integer	The portNumber of the negative terminal.	<code>&lt;diffName&gt;</code>	String	The new name of the differential pin (e.g., Diff1).	<code>&lt;commName&gt;</code>	String	The new name of the common pin (e.g., Comm1).	<code>&lt;diffImpedance&gt;</code>	Double	The differential pin characteristic impedance.	<code>&lt;commImpedance&gt;</code>	Double	The common pin characteristic impedance.	<code>&lt;matched&gt;</code>	Boolean	Specify whether to use matched pair.  <b>Note:</b> The default value is <b>False</b> .
Name	Type	Description																									
<code>&lt;positiveTerminal&gt;</code>	Integer	The portNumber of the positive terminal.																									
<code>&lt;negativeTerminal&gt;</code>	Integer	The portNumber of the negative terminal.																									
<code>&lt;diffName&gt;</code>	String	The new name of the differential pin (e.g., Diff1).																									
<code>&lt;commName&gt;</code>	String	The new name of the common pin (e.g., Comm1).																									
<code>&lt;diffImpedance&gt;</code>	Double	The differential pin characteristic impedance.																									
<code>&lt;commImpedance&gt;</code>	Double	The common pin characteristic impedance.																									
<code>&lt;matched&gt;</code>	Boolean	Specify whether to use matched pair.  <b>Note:</b> The default value is <b>False</b> .																									
Return Value	Boolean.																										

Python Syntax	<code>AddDiffPair()</code>
Python Example	<code>success = oPostProc.AddDiffPair(2, 1, "Diff1", "Comm1", 100, 25)</code>

<b>VB Syntax</b>	AddDiffPair
<b>VB Example</b>	Set success = oPostProc.AddDiffPair 2, 1, "Diff1", "Comm1", 100, 25

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## Cascade (SPISim)

Creates new network data from a group of network data objects cascaded together. The network data must have an even number of terminal ports and must have the same number of ports.

**Note:** Cascade opens and interacts with **SPISim** to complete.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Cascade</b> from the drop-down menu.		
<b>Parameters</b>	<b>Name</b> <dataArray>	<b>Type</b> Array	<b>Description</b> These network data objects are cascaded together to create the new network data.
<b>Return Value</b>	Network IDispatch. <b>Note:</b> Cascade returns <b>None</b> if the specified network data does not have compatible terminal ports defined.		

<b>Python Syntax</b>	Cascade()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Cascade([oData1, oData2, oData3])</pre>

<b>VB Syntax</b>	Cascade
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData2 = oNDE.Cascade Array(oData1, oData2, oData3)</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## ClearDiffPairs

Clears all differential pair definitions. All other post-processing settings remain the same.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Differential Pairs</b> in the <b>NDE</b> ribbon to open the <b>Differential Pairs</b> window. Then select differential pairs from the rightmost box and click <b>&lt;&lt; Remove Pairs</b> .
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	ClearDiffPairs()
----------------------	------------------

<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.ClearDiffPairs()</pre>
-----------------------	-----------------------------------------------------------------------------

<b>VB Syntax</b>	ClearDiffPairs
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") oPostProc.ClearDiffPairs</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## Clone

Creates a copy of the network data object.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select the network data object to clone. Then select <b>Clone</b> from the <b>NDE</b> ribbon.
<b>Parameters</b>	None.
<b>Return Value</b>	Network IDispatch.

<b>Python Syntax</b>	Clone()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData1 = oData.Clone()</pre>

<b>VB Syntax</b>	Clone
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData1 = oData.Clone</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## Close

Closes the network data object. The object will no longer be accessible.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, click <b>Close</b> in the <b>NDE</b> ribbon, or select <b>File &gt; Close</b> .
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	Close()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData.Close()</pre>

<b>VB Syntax</b>	Close
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer")</pre>

---

	<code>oData.Close</code>
--	--------------------------

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## Combine (SPISim)

Combines frequencies from multiple network data objects to create a new network data with all of the frequencies.

**Note:** Combine opens and interacts with **SPISim** to complete.

UI Access	From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Combine</b> from the drop-down menu.			
Parameters	<b>Name</b> <code>&lt;dataArray&gt;</code>	<b>Type</b> Array	<b>Description</b> These network data objects are combined to create the new network data.	
	<b>Name</b> <code>&lt;freqTol&gt;</code>	<b>Type</b> Double	<b>Description</b> If $\text{abs}(f1 - f2) < \text{freqTol}$ , $f1$ and $f2$ are considered the same frequency.  <b>Note:</b> The default value is <b>100</b> .	
Return Value	Network IDispatch.  <b>Note:</b> Combine returns <b>None</b> if the network data does not have the same number of ports.			

<b>Python Syntax</b>	Combine()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData4 = oNDE.Combine([oData1, oData2, oData3], 1000)</pre>

<b>VB Syntax</b>	Combine
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData4 = oNDE.Combine Array(oData1, oData2, oData3), 1000</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## Deembed (SPISim)

Creates a new network data, deembedding the frontData and the backData from the front and back of the originalData.

**Note:** Deembed opens and interacts with **SPISim** to complete.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Deembed</b> from the drop-down menu to open the <b>Deembed...</b> window, and choose <b>Given 3 S-params in order of: Total, Front, Back</b> from the <b>Settings</b> group box.									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;originalData&gt;</td> <td>Object</td> <td>Original network data that is deembedded.</td> </tr> <tr> <td>&lt;frontData&gt;</td> <td>Object</td> <td>Network data that is deembedded from the front of the original.</td> </tr> </tbody> </table>	Name	Type	Description	<originalData>	Object	Original network data that is deembedded.	<frontData>	Object	Network data that is deembedded from the front of the original.
Name	Type	Description								
<originalData>	Object	Original network data that is deembedded.								
<frontData>	Object	Network data that is deembedded from the front of the original.								

	<code>&lt;backData&gt;</code>	Object	Network data that is deembeded from the back of the original.
	Network IDispatch.		
<b>Return Value</b>	<p><b>Note:</b> Deembed returns <b>None</b> if the specified network data does not have compatible terminal ports defined.</p>		

<b>Python Syntax</b>	<code>Deembed()</code>
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Deembed(oData1, oDataFront, oDataBack)</pre>

<b>VB Syntax</b>	<code>Deembed</code>
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData2 = oNDE.Deembed oData1, oDataFront, oDataBack</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## DeembedBack (SPISim)

Creates a new network data, dembedding the backData from the back of the originalData.

**Note:** DeembedBack opens and interacts with **SPISim** to complete.

UI Access	From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Deembed</b> from the drop-down menu to open the <b>Deembed...</b> window, and choose <b>Given 3 S-params in order of: Total, Back</b> from the <b>Settings</b> group box.											
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;originalData&gt;</td> <td>Object</td> <td>Original network data that is deembeded.</td> </tr> <tr> <td>&lt;backData&gt;</td> <td>Object</td> <td>Network data that is deembeded from the back of the original.</td> </tr> </tbody> </table>			Name	Type	Description	<originalData>	Object	Original network data that is deembeded.	<backData>	Object	Network data that is deembeded from the back of the original.
Name	Type	Description										
<originalData>	Object	Original network data that is deembeded.										
<backData>	Object	Network data that is deembeded from the back of the original.										
Return Value	<p>Network IDispatch.</p> <p><b>Note:</b> DeembedBack returns <b>None</b> if the specified network data does not have compatible terminal ports defined.</p>											

Python Syntax	DeembedBack()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.DeembedBack(oData1, oDataBack)</pre>

VB Syntax	DeembedBack
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData2 = oNDE.DeembedBack oData1, oDataBack</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## DeembedFront (SPISim)

Creates a new network data, deembedding the frontData from the front of the originalData.

**Note:** DeembedFront opens and interacts with **SPISim** to complete.

UI Access	From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Deembed</b> from the drop-down menu to open the <b>Deembed...</b> window, and choose <b>Given 3 S-params in order of: Total, Front</b> from the <b>Settings</b> group box.											
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;originalData&gt;</td> <td>Object</td> <td>Original network data that is deembedded.</td> </tr> <tr> <td>&lt;frontData&gt;</td> <td>Object</td> <td>Network Data that is deembedded from the front of the original.</td> </tr> </tbody> </table>			Name	Type	Description	<originalData>	Object	Original network data that is deembedded.	<frontData>	Object	Network Data that is deembedded from the front of the original.
Name	Type	Description										
<originalData>	Object	Original network data that is deembedded.										
<frontData>	Object	Network Data that is deembedded from the front of the original.										
Return Value	<p>Network IDispatch.</p> <p><b>Note:</b> DeembedFront returns <b>None</b> if the specified network data does not have compatible terminal ports defined.</p>											

Python Syntax	DeembedFront()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.DeembedFront(oData1, oDataFront)</pre>

<b>VB Syntax</b>	DeembedFront
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData2 = oNDE.DeembedFront oData1, oDataFront</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## DisableDiffPairs

Deactivates all differential pair definitions. This script does not remove them, so they [can be reactivated](#).

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Differential Pairs</b> in the <b>NDE</b> ribbon to open the <b>Differential Pairs</b> window. Once differential pairs have been added to the rightmost box (i.e., select differential pairs from the Pairs group box and click <b>Add Pairs &gt;&gt;</b> ), remove check marks from the <b>Enabled</b> column to deactivate the corresponding differential pairs.
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	DisableDiffPairs()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.DisableDiffPairs()</pre>

<b>VB Syntax</b>	DisableDiffPairs
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") oPostProc.DisableDiffPairs</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## EnableDiffPairs

Enables all differential pair definitions.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Differential Pairs</b> in the <b>NDE</b> ribbon to open the <b>Differential Pairs</b> window. Once differential pairs have been added to the rightmost box (i.e., select differential pairs from the Pairs group box and click <b>Add Pairs &gt;&gt;</b> ), add check marks to the <b>Enabled</b> column to activate the corresponding differential pairs.
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	EnableDiffPairs()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oPostProc.EnableDiffPairs()</pre>

<b>VB Syntax</b>	EnableDiffPairs
------------------	-----------------

**VB Example**

```
Set oNDE = oDesktop.GetTool("ndExplorer")
Set success = oPostProc.EnableDiffPairs
```

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## ExportCitiFile

Writes the S-parameters to disk in Citifile format (\*.cit text file).

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>File &gt; Save As</b> to open a <b>Save As</b> explorer window. Then select <b>Citifile (*.cit)</b> from the <b>Save as type:</b> drop-down menu.		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	<filePath>	String	The full path to the file.
	<matrixType>	Integer	<p>One of the following:</p> <ul style="list-style-type: none"> <li>• 0 – S-Parameters</li> <li>• 1 – Y-Parameters</li> <li>• 2 – Z-Parameters</li> <li>• 3 – Gamma</li> <li>• 4 – Impedance</li> </ul> <p><b>Note:</b> The default value is <b>0</b>.</p>
	<formalType>	Integer	<p>One of the following:</p> <ul style="list-style-type: none"> <li>• 0 – Magnitude/Angle (degrees)</li> </ul>

			<ul style="list-style-type: none"> <li>• 1 – Real/Imaginary</li> <li>• 2 – dB/Angle (degrees)</li> </ul> <p><b>Note:</b> The default value is <b>0</b>.</p>
	<code>&lt;precision&gt;</code>	Integer	The number of significant figures.  <b>Note:</b> The default value is <b>10</b> .
	<code>&lt;frequencies&gt;</code>	Array of Doubles	A subset of the calculated frequencies. Any array values that don't have calculated data will be ignored. An empty array will cause all frequencies to be written.  <b>Note:</b> The default value is an empty array.
<b>Return Value</b>	Boolean True if file was successfully written; else False.		

<b>Python Syntax</b>	<code>ExportCitiFile()</code>
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.ExportCitiFile("D:\folder\ne133.cit", 0, 1, 12, [])</pre>

<b>VB Syntax</b>	<code>ExportCitiFile</code>
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set success = oData.ExportCitiFile "D:\folder\ne133.cit", 0, 1, 12, Array()</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## ExportMatlab

Writes the S-parameters to disk in Matlab format (\*.m text file).

UI Access	From the <b>Network Data Explorer</b> tab, select <b>File &gt; Save As</b> to open a <b>Save As</b> explorer window. Then select <b>MATLAB (*.m)</b> from the <b>Save as type:</b> drop-down menu.		
Parameters	<b>Name</b>	<b>Type</b>	<b>Description</b>
	<code>&lt;filePath&gt;</code>	String	The full path to the file.

- |                   |                                 |             |                                                                                                                                                                                                         |
|-------------------|---------------------------------|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Parameters</b> | <b>Name</b>                     | <b>Type</b> | <b>Description</b>                                                                                                                                                                                      |
|                   | <code>&lt;matrixType&gt;</code> | Integer     | <p>One of the following:</p> <ul style="list-style-type: none"> <li>• 0 – S-Parameters</li> <li>• 1 – Y-Parameters</li> <li>• 2 – Z-Parameters</li> <li>• 3 – Gamma</li> <li>• 4 – Impedance</li> </ul> |

**Note:** The default value is **0**.

- |                   |                                 |             |                                                                                                                                                                                |
|-------------------|---------------------------------|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Parameters</b> | <b>Name</b>                     | <b>Type</b> | <b>Description</b>                                                                                                                                                             |
|                   | <code>&lt;formalType&gt;</code> | Integer     | <p>One of the following:</p> <ul style="list-style-type: none"> <li>• 0 – Magnitude/Angle (degrees)</li> <li>• 1 – Real/Imaginary</li> <li>• 2 – dB/Angle (degrees)</li> </ul> |

			<b>Note:</b> The default value is <b>0</b> .
	<code>&lt;precision&gt;</code>	Integer	The number of significant figures.  <b>Note:</b> The default value is <b>10</b> .
	<code>&lt;frequencies&gt;</code>	Array of Doubles	A subset of the calculated frequencies. Any array values that don't have calculated data will be ignored. An empty array will cause all frequencies to be written.  <b>Note:</b> The default value is an empty array.
<b>Return Value</b>	Boolean True if file was successfully written; else False.		

<b>Python Syntax</b>	<code>ExportMatlab()</code>
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.ExportMatlab("D:\folder\ne133.m", 0, 1, 12, [])</pre>

<b>VB Syntax</b>	<code>ExportMatlab</code>
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool "ndExplorer" Set success = oData.ExportMatlab "D:\folder\ne133.m", 0, 1, 12, Array()</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## ExportSpreadsheet

Writes the S-parameters to disk in spreadsheet format (\*.tab text file).

UI Access	From the <b>Network Data Explorer</b> tab, select <b>File &gt; Save As</b> to open a <b>Save As</b> explorer window. Then select <b>Data Table (spreadsheet) (*.tab)</b> from the <b>Save as type:</b> drop-down menu.		
Parameters	<b>Name</b>	<b>Type</b>	<b>Description</b>
	<code>&lt;filePath&gt;</code>	String	The full path to the file.
<code>&lt;matrixType&gt;</code>		Integer	One of the following: <ul style="list-style-type: none"> <li>• 0 – S-Parameters</li> <li>• 1 – Y-Parameters</li> <li>• 2 – Z-Parameters</li> <li>• 3 – Gamma</li> <li>• 4 – Impedance</li> </ul> <p><b>Note:</b> The default value is <b>0</b>.</p>
<code>&lt;formalType&gt;</code>		Integer	One of the following: <ul style="list-style-type: none"> <li>• 0 – Magnitude/Angle (degrees)</li> <li>• 1 – Real/Imaginary</li> <li>• 2 – dB/Angle (degrees)</li> </ul>

			<b>Note:</b> The default value is <b>0</b> .
	<precision>	Integer	The number of significant figures.  <b>Note:</b> The default value is <b>10</b> .
	<frequencies>	Array of Doubles	A subset of the calculated frequencies. Any array values that don't have calculated data will be ignored. An empty array will cause all frequencies to be written.  <b>Note:</b> The default value is an empty array.
	<renormalize>	Boolean	If <b>True</b> , will renormalize the data to the value of renormImpedance before writing.  <b>Note:</b> The default value is <b>False</b> .
	<renormImpedance>	Double	Data will be renormalized to this impedance before writing only if renormalize is <b>True</b> .  <b>Note:</b> The default value is <b>50 ohms</b> .
<b>Return Value</b>	Boolean True if file was successfully written; else False.		

<b>Python Syntax</b>	ExportSpreadsheet()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.ExportSpreadsheet("D:\folder\ne133.tab", 0, 1, 12, [], True, 23)</pre>

<b>VB Syntax</b>	ExportSpreadsheet
<b>VB Example</b>	<pre>Set oData=oDesktop.GetTool("ndExplorer") success = oData.ExportSpreadsheet "D:\folder\ne133.tab", 0, 1, 12, Array(), True, 23</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## ExportTouchstone

Writes the S-parameters to disk in Touchstone 1.0 format (\*.snp text file).

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>File &gt; Save As</b> to open a <b>Save As</b> explorer window. Then select <b>Touchstone Format 1.0 (*.snp)</b> from the <b>Save as type:</b> drop-down menu.											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;filePath&gt;</td> <td>String</td> <td>The full path to the file.</td> </tr> <tr> <td>&lt;matrixType&gt;</td> <td>Integer</td> <td>           One of the following:           <ul style="list-style-type: none"> <li>• 0 – S-Parameters</li> <li>• 1 – Y-Parameters</li> <li>• 2 – Z-Parameters</li> <li>• 3 – Gamma</li> <li>• 4 – Impedance</li> </ul> </td> </tr> </tbody> </table>	Name	Type	Description	<filePath>	String	The full path to the file.	<matrixType>	Integer	One of the following: <ul style="list-style-type: none"> <li>• 0 – S-Parameters</li> <li>• 1 – Y-Parameters</li> <li>• 2 – Z-Parameters</li> <li>• 3 – Gamma</li> <li>• 4 – Impedance</li> </ul>		
Name	Type	Description										
<filePath>	String	The full path to the file.										
<matrixType>	Integer	One of the following: <ul style="list-style-type: none"> <li>• 0 – S-Parameters</li> <li>• 1 – Y-Parameters</li> <li>• 2 – Z-Parameters</li> <li>• 3 – Gamma</li> <li>• 4 – Impedance</li> </ul>										

		<b>Note:</b> The default value is <b>0</b> .
<code>&lt;formalType&gt;</code>	Integer	<p>One of the following:</p> <ul style="list-style-type: none"> <li>• 0 – Magnitude/Angle (degrees)</li> <li>• 1 – Real/Imaginary</li> <li>• 2 – dB/Angle (degrees)</li> </ul> <p><b>Note:</b> The default value is <b>0</b>.</p>
<code>&lt;precision&gt;</code>	Integer	<p>The number of significant figures.</p> <p><b>Note:</b> The default value is <b>10</b>.</p>
<code>&lt;frequencies&gt;</code>	Array of Doubles	<p>A subset of the calculated frequencies. Any array values that don't have calculated data will be ignored. An empty array will cause all frequencies to be written.</p> <p><b>Note:</b> The default value is an empty array.</p>
<code>&lt;renormalize&gt;</code>	Boolean	<p>If <b>True</b>, will renormalize the data to the value of <code>renormImpedance</code> before writing.</p> <p><b>Note:</b> The default value is <b>False</b>.</p>
<code>&lt;renormImpedance&gt;</code>	Double	<p>Data will be renormalized to this impedance before writing only if <code>renormalize</code> is <b>True</b>.</p> <p><b>Note:</b> The default value is <b>50 ohms</b>.</p>
<code>&lt;freqUnits&gt;</code>	String	<p>One of the following: "Hz", "KHz", "MHz", "GHz".</p>

			<b>Note:</b> The default value is "Hz".
<b>Return Value</b>	Boolean True if file was successfully written; else False.		

<b>Python Syntax</b>	ExportTouchstone()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.ExportTouchstone("D:\folder\ne133.s2p", 0, 1, 12, [], True, 23, "GHz")</pre>

<b>VB Syntax</b>	ExportTouchstone
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") success = oData.ExportTouchstone "D:\folder\ne133.s2p", 0, 1, 12, Array(), True, 23, "GHz"</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## ExportTouchstone2

Writes the S-parameters to disk in Touchstone 2 format (\*.ts text file).

UI Access	From the <b>Network Data Explorer</b> tab, select <b>File &gt; Save As</b> to open a <b>Save As</b> explorer window. Then select <b>Touchstone 2 Format (*.ts)</b> from the <b>Save as type:</b> drop-down menu.		
Parameters	<b>Name</b>	<b>Type</b>	<b>Description</b>
	<code>&lt;filePath&gt;</code>	String	The full path to the file.
	<code>&lt;matrixType&gt;</code>	Integer	<p>One of the following:</p> <ul style="list-style-type: none"> <li>• 0 – S-Parameters</li> <li>• 1 – Y-Parameters</li> <li>• 2 – Z-Parameters</li> <li>• 3 – Gamma</li> <li>• 4 – Impedance</li> </ul> <p><b>Note:</b> The default value is <b>0</b>.</p>
	<code>&lt;formalType&gt;</code>	Integer	<p>One of the following:</p> <ul style="list-style-type: none"> <li>• 0 – Magnitude/Angle (degrees)</li> <li>• 1 – Real/Imaginary</li> <li>• 2 – dB/Angle (degrees)</li> </ul> <p><b>Note:</b> The default value is <b>0</b>.</p>
	<code>&lt;precision&gt;</code>	Integer	The number of significant figures.
	<code>&lt;frequencies&gt;</code>	Array of Doubles	A subset of the calculated frequencies. Any array values that don't have calculated data will be ignored. An empty array will cause all frequencies to be written.

			<b>Note:</b> The default value is an empty array.
<renormalize>	Boolean	If <b>True</b> , will renormalize the data to the value of renormImpedance before writing.	<b>Note:</b> The default value is <b>False</b> .
<renormImpedance>	Double	Data will be renormalized to this impedance before writing only if renormalize is <b>True</b> .	<b>Note:</b> The default value is <b>50 ohms</b> .
<freqUnits>	String	One of the following: "Hz", "KHz", "MHz", "GHz".	<b>Note:</b> The default value is <b>"Hz"</b> .
<b>Return Value</b>	Boolean True if file was successfully written; else False.		

<b>Python Syntax</b>	ExportTouchstone2()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.ExportTouchstone2("D:\folder\ne133.ts", 0, 1, 12, [], True, 23, "GHz")</pre>

<b>VB Syn-</b>	ExportTouchstone2
----------------	-------------------

<b>tax</b>	
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set success = oData.ExportTouchstone2 "D:\folder\nel33.ts", 0, 1, 12, Array(), True, 23, "GHz"</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## Extract (SPISim)

Creates a new smaller network data after removing data for the specified terminal port numbers.

**Note:** Extract opens and interacts with **SPISim** to complete.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Extract</b> from the drop-down menu.		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.
	<portNumbers>	Array of Integers	The port numbers that will be retained (integers between 1 and <i>n</i> ).
<b>Return Value</b>	Network IDispatch.		

<b>Python Syntax</b>	Extract()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer")</pre>

	<code>oData2 = oNDE.Extract(oData1, [3, 2])</code>
--	----------------------------------------------------

<b>VB Syntax</b>	Extract
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool "ndExplorer" Set oData2 = oNDE.Extract oData1, Array(3, 2)</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## GetFrequencies

Returns the frequency values with calculated data in the network data.

<b>UI Access</b>	None.
<b>Parameters</b>	None.
<b>Return Value</b>	Array of doubles.

<b>Python Syntax</b>	<code>GetFrequencies()</code>
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") Freqs = oData.GetFrequencies()</pre>

<b>VB Syntax</b>	GetFrequencies
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set Freqs = oData.GetFrequencies</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## GetFrequencyCount

Returns the number of frequencies with calculated data in the network data.

<b>UI Access</b>	None.
<b>Parameters</b>	None.
<b>Return Value</b>	Integer number of frequencies.

<b>Python Syntax</b>	GetFrequencyCount()
<b>Python Example</b>	<pre>oData = oDesktop.GetTool("ndExplorer") numFreqs = oData.GetFrequencyCount()</pre>

<b>VB Syntax</b>	GetFrequencyCount
<b>VB Example</b>	<pre>Set oData = oDesktop.GetTool("ndExplorer") Set numFreqs = oData.GetFrequencyCount</pre>

## GetName

Returns the name of the active design.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.
<b>Return Value</b>	String indicating the name of the active design.

<b>Python Syntax</b>	GetName()
<b>Python Example</b>	<code>design_name = oDesign.GetName()</code>

<b>VB Syntax</b>	GetName
<b>VB Example</b>	<code>design_name = oDesign.GetName</code>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## GetPortCount

Returns the total number of ports.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Edit Ports</b> in the <b>NDE</b> ribbon to open the <b>Port properties</b> window.
<b>Parameters</b>	None.
<b>Return Value</b>	Integer number of ports.

<b>Python Syntax</b>	GetPortCount()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") num = oData.GetPortCount()</pre>

<b>VB Syntax</b>	GetPortCount
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") num = oData.GetPortCount</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## GetPortNumber

Returns the terminal port number for a given terminal port name.

**Note:** GetPortNumber can be used in other commands that require a port number, if a port name is preferable.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, ensure the <b>Full Port Names</b> check box is activated. Port numbers will be
------------------	---------------------------------------------------------------------------------------------------------------------------

	represented in the object portNames.						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;portName&gt;</td><td>String</td><td>The designation of the port number.</td></tr></tbody></table>	Name	Type	Description	<portName>	String	The designation of the port number.
Name	Type	Description					
<portName>	String	The designation of the port number.					
<b>Return Value</b>	Integer.						

<b>Python Syntax</b>	GetPortNumber()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") index = oPostProc.GetPortNumber("Input35")</pre>

<b>VB Syntax</b>	GetPortNumber
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") index = oPostProc.GetPortNumber "Input35"</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## GetPostProcSettings

Returns a copy of the IDispatch to the postprocessing settings for the network data (oPostProc).

**Note:** Making changes to the PostProcSettings will not change the network data. To make changes, call oData.SetPostProcSettings to change the network data.

<b>UI Access</b>	None.
<b>Parameters</b>	None.
<b>Return Value</b>	PostProcSettings IDispatch

<b>Python Syntax</b>	GetPostProcSettings()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc = oData.GetPostProcSettings()</pre>

<b>VB Syntax</b>	GetPostProcSettings
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oPostProc = oData.GetPostProcSettings</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## GetVariation

Returns the variation key for the network data.

<b>UI Access</b>	None.
<b>Parameters</b>	None.
<b>Return Value</b>	String variation key.

<b>Python Syntax</b>	GetVariation()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") variation = oData.GetVariation()</pre>

<b>VB Syntax</b>	GetVariation
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set variation = oData.GetVariation</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## HasSameData

Compares one network data object with another network data. Actual calculated values will be compared and no interpolation will be done.

<b>UI Access</b>	None.
------------------	-------

Parameters	Name	Type	Description
	<NetworkData>	Object	The second network data to compare against.
	<matrixType>	Integer	<p>One of the following:</p> <ul style="list-style-type: none"> <li>• 0 – S-Parameters</li> <li>• 1 – Y-Parameters</li> <li>• 2 – Z-Parameters</li> <li>• 3 – Gamma</li> <li>• 4 – Impedance</li> </ul> <p><b>Note:</b> The default value is <b>0</b>.</p>
	<comparePortNames>	Boolean	If <b>True</b> , port names must be identical.  <b>Note:</b> The default value is <b>True</b> .
	<compareNoise>	Boolean	If <b>True</b> , and both network data have noise data, the comparison will fail unless the noise values also match.  <b>Note:</b> The default value is <b>10</b> .
	<relativeTolerance>	Double	If the absoluteTolerance test fails, then $\text{abs}(\text{value1} - \text{value2}) / \max(\text{abs}(\text{value1}), \text{abs}(\text{value2}))$ must be less than this to be considered equal.  <b>Note:</b> The default value is <b>1e-14</b> .
	<absoluteTolerance>	Double	If <b>True</b> , then $\text{abs}(\text{value1} - \text{value2}) < \text{absoluteTolerance}$ .  <b>Note:</b> The default value is <b>0</b> .

<b>Return Value</b>	Boolean True if the compared network data have the same frequency and matrix values; otherwise, False.
---------------------	--------------------------------------------------------------------------------------------------------

<b>Python Syntax</b>	HasSameData()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.HasSameData(oData2, 0, True, False, 1e-10, 0)</pre>

<b>VB Syntax</b>	HasSameData
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set success = oData.HasSameData oData2, 0, True, False, 1e-10, 0</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## LoadSolution

Loads the specified design (all variations) and returns an integer ID.

<b>UI Access</b>	After analyzing a design, from the <b>Project Manager</b> window, expand the <b>Project Tree</b> and Analysis folders. Then right-click on the completed analysis icon and select <b>Network Data Explorer</b> to open the solution in the <b>Network Data Explorer</b> tab.								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt; projectName &gt;</td> <td>String</td> <td>Full path to the Electronics Desktop file.</td> </tr> </tbody> </table>			Name	Type	Description	< projectName >	String	Full path to the Electronics Desktop file.
Name	Type	Description							
< projectName >	String	Full path to the Electronics Desktop file.							

	<code>&lt;designName&gt;</code>	String	Name of the design.
	<code>&lt;solutionName&gt;</code>	String	Name of the solution.
<b>Return Value</b>	Integer ID (identifying the solution); < 0 if the solution is not loaded.		

<b>Python Syntax</b>	<code>LoadSolution()</code>
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") id = oNDE.LoadSolution("D:\folder\Tee.aedt", "HFSS_Test", "Setup1:Sweep1")</pre>

<b>VB Syntax</b>	<code>LoadSolution</code>
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") id = oNDE.LoadSolution "D:\folder\Tee.aedt", "HFSS_Test", "Setup1:Sweep1"</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## Open

Reads contents of file and returns the IDispatch for the new network data.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Open</b> in the <b>NDE</b> ribbon, or select <b>File &gt; Open</b> to open an explorer window. Then navigate to the required S-parameter file.			
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> </table>	Name	Type	Description
Name	Type	Description		

	<code>&lt;fileName&gt;</code>	String	Full path to the file containing S-parameters.
<b>Return Value</b>	Network IDispatch.		

<b>Python Syntax</b>	<code>Open()</code>
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData = oNDE.Open("D:\folder\test.s2p")</pre>

<b>VB Syntax</b>	<code>Open</code>
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData = oNDE.Open "D:\folder\test.s2p"</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## Rename (SPISim)

Creates a new network data of the same size with the terminal ports renamed.

**Note:** Rename opens and interacts with **SPISim** to complete.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Rename</b> from the
------------------	----------------------------------------------------------------------------------------------------------------------------------

	drop-down menu.		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.
<b>Return Value</b>	Network IDispatch.		

<b>Python Syntax</b>	Rename()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Rename(oData1, ["input", "control", "output"])</pre>

<b>VB Syntax</b>	Rename
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool "ndExplorer" Set oData2 = oNDE.Rename oData1, Array("input", "control", "output")</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## Renormalize (SPISim)

Creates a new network data of the same size after renormalizing the terminal ports using the specified impedance values.

**Note:** Rename opens and interacts with **SPISim** to complete.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Renormalize</b> from the drop-down menu.		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.
<b>Return Value</b>	Network IDispatch. <b>Note:</b> Renormalize returns <b>None</b> if there is not an impedance value for each port.		

<b>Python Syntax</b>	Renormalize()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Renormalize(oData1, [3, 2])</pre>

<b>VB Syntax</b>	Renormalize
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Renormalize oData1, Array(3, 2)</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## Reorder

Rearranges the terminal port entries but does not change the port names (i.e., switching first and third terminal port positions will give switched values for S(1,1) and S(3,3) but S(Port1,Port1) and S(Port3,Port3) do not change).

UI Access	From the <b>Network Data Explorer</b> tab, select <b>Edit Ports</b> in the <b>NDE</b> ribbon to open the <b>Port properties</b> window. <b>Click+drag</b> the terminal port rows to reorder them. Make changes, as necessary, then click <b>OK</b> .								
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;newOrder&gt;</td> <td>Array of Integers</td> <td>Must have one entry for each port; all numbers 1 to <i>n</i> must be present.</td> </tr> </tbody> </table>			Name	Type	Description	<newOrder>	Array of Integers	Must have one entry for each port; all numbers 1 to <i>n</i> must be present.
Name	Type	Description							
<newOrder>	Array of Integers	Must have one entry for each port; all numbers 1 to <i>n</i> must be present.							
Return Value	None.								

Python Syntax	Reorder()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.Reorder([3, 2, 1])</pre>

VB Syntax	Reorder
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") oPostProc.Reorder Array(3, 2, 1)</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## Reorder (SPISim)

Creates a new network data with the same number of terminal ports and with the port names in the same order. The data is also reordered so it corresponds to the new order (e.g., the data for S(Port1, Port1) may correspond to S(Port3, Port3)).

**Note:** Reorder opens and interacts with **SPISim** to complete.

UI Access	From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Reorder</b> from the drop-down menu.		
Parameters	Name	Type	Description
	<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.
Return Value	Network IDispatch.  <b>Note:</b> Reorder returns <b>None</b> if the array argument does not contain all the terminal port numbers, in any order.		

Python Syntax	Reorder()
---------------	-----------

<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Reorder([3, 2, 1])</pre>
-----------------------	-----------------------------------------------------------------------------------

<b>VB Syntax</b>	Reorder
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData2 = oNDE.Reorder Array(3, 2, 1)</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## Reset

Resets the post-processing to what it was when the network data was created.

**Note:** This is not equivalent to setting post-processing to an empty state. The network data may have been read from a file or created from solution data that already had post-processing settings.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Reset postprocessing</b> in the <b>NDE</b> ribbon.
<b>Parameters</b>	None.
<b>Return Value</b>	None.

<b>Python Syntax</b>	<code>Reset()</code>
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer")</pre>

	<code>oPostProc.Reset()</code>
--	--------------------------------

<b>VB Syntax</b>	Reset
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") oPostProc.Reset</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## SetAllPortImpedances

Sets all terminal port impedances in a single call. The impedances are in an array of real or complex values.

<b>UI Access</b>	Through the UI, the terminal ports can <b>only</b> be edited individually. From the <b>Network Data Explorer</b> tab, select <b>Edit Ports</b> in the <b>NDE</b> ribbon to open the <b>Port properties</b> window. Make changes, then click <b>OK</b> .								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td><code>&lt;impedances&gt;</code></td><td>Array of Doubles or String</td><td>Must have one entry for each port or a single complex (or real) value which will be applied to all ports.</td></tr></tbody></table>			Name	Type	Description	<code>&lt;impedances&gt;</code>	Array of Doubles or String	Must have one entry for each port or a single complex (or real) value which will be applied to all ports.
Name	Type	Description							
<code>&lt;impedances&gt;</code>	Array of Doubles or String	Must have one entry for each port or a single complex (or real) value which will be applied to all ports.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	<code>SetAllPortImpedances()</code>
----------------------	-------------------------------------

<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.SetAllPortImpedances([23, "2+3i", 50]) -or- oData.SetAllPortImpedances(50)</pre>
-----------------------	---------------------------------------------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	SetAllPortImpedances
<b>VB Example</b>	<pre>Set oData=oDesktop.GetTool("ndExplorer") oPostProc.SetAllPortImpedances Array(23, "2+3i", 50) -or- oData.SetAllPortImpedances 50</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## SetPortDeembedDistance

Sets terminal port impedance for a single port.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Edit Ports</b> in the <b>NDE</b> ribbon to open the <b>Port properties</b> window. Make changes, then click <b>OK</b> .											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;portNumber&gt;</td> <td>Integer</td> <td>The port number of the changed terminal port (integers between 1 and n).</td> </tr> <tr> <td>&lt;distance&gt;</td> <td>String</td> <td>Impedance with units (e.g., "20mm"); if no units, meters are assumed</td> </tr> </tbody> </table>			Name	Type	Description	<portNumber>	Integer	The port number of the changed terminal port (integers between 1 and n).	<distance>	String	Impedance with units (e.g., "20mm"); if no units, meters are assumed
Name	Type	Description										
<portNumber>	Integer	The port number of the changed terminal port (integers between 1 and n).										
<distance>	String	Impedance with units (e.g., "20mm"); if no units, meters are assumed										
<b>Return Value</b>	None.											

<b>Python Syntax</b>	SetPortDeembedDistance()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.SetPortDeembedDistance(2, "20mm")</pre>

<b>VB Syntax</b>	SetPortDeembedDistance
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") oPostProc.SetPortDeembedDistance 2, "20mm"</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## SetPortImpedance

Sets port impedance for a single port.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Edit Ports</b> in the <b>NDE</b> ribbon to open the <b>Port properties</b> window. Make changes, then click <b>OK</b> .											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;portNumber&gt;</td> <td>Integer</td> <td>The port number of the changed terminal port (integers between 1 and <i>n</i>).</td> </tr> <tr> <td>&lt;impedance&gt;</td> <td>Double or String</td> <td>Double if impedance value is real; string format (e.g., 24+10i) for complex impedance.</td> </tr> </tbody> </table>			Name	Type	Description	<portNumber>	Integer	The port number of the changed terminal port (integers between 1 and <i>n</i> ).	<impedance>	Double or String	Double if impedance value is real; string format (e.g., 24+10i) for complex impedance.
Name	Type	Description										
<portNumber>	Integer	The port number of the changed terminal port (integers between 1 and <i>n</i> ).										
<impedance>	Double or String	Double if impedance value is real; string format (e.g., 24+10i) for complex impedance.										
<b>Return Value</b>	None.											

<b>Python Syntax</b>	SetPortImpedance()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.SetPortImpedance(2, "25+3i")</pre>

<b>VB Syntax</b>	SetPortImpedance
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") oPostProc.SetPortImpedance 2, "25+3i"</pre>

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## SetPostProcSettings

Applies postprocessing settings (oPostProc) to the specified network data.

**Note:** Making changes to the PostProcSettings will not change the network data. To make changes, call oData.SetPostProcSettings to change the network data.

<b>UI Access</b>	None.								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;PostProcSettings&gt;</td> <td>Object</td> <td>The settings that will be applied to the data.</td> </tr> </tbody> </table>			Name	Type	Description	<PostProcSettings>	Object	The settings that will be applied to the data.
Name	Type	Description							
<PostProcSettings>	Object	The settings that will be applied to the data.							
<b>Return Value</b>	None.								

**Python Syntax**

```
SetPostProcSettings()
```

**Python Example**

```
oNDE = oDesktop.GetTool("ndExplorer")
oData.SetPostProcSettings(oPostProc)
```

**VB Syntax**

```
SetPostProcSettings
```

**VB Example**

```
Set oNDE = oDesktop.GetTool("ndExplorer")
oData.SetPostProcSettings oPostProc
```

**Warning:** The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

## Smooth

Creates a new network data with values smoothed. The number of adjacent points smoothed is user-specified.

**UI Access**

From the **Network Data Explorer** tab, select **Transforms** in the **NDE** ribbon. Then select **Smooth** from the drop-down menu.

**Parameters**

Name	Type	Description
<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.
<order>	Integer	The number of adjacent points that are smoothed.

			<b>Note:</b> The default value is <b>10</b> .
	<enforceCausality>	Boolean	Causal data is calculated before smoothing.
<b>Note:</b> The default value is <b>False</b> .			
<b>Return Value</b>	Network IDispatch.		

<b>Python Syntax</b>	Smooth()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Smooth(oData1, 3, True)</pre>

<b>VB Syntax</b>	Smooth
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData2 = oNDE.Smooth oData1, 3, True</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## Stretch (SPISim)

Creates a new network data representing a matrix of transmission lines with the length of those transmission lines multiplied by a factor of  $n$ .

**Note:** Stretch opens and interacts with **SPISim** to complete.

<b>UI Access</b>	From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Stretch</b> from the drop-down menu.		
<b>Parameters</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
	<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.
<b>Return Value</b>	<p>Network IDispatch.</p> <p><b>Note:</b> Stretch returns <b>None</b> if the network data argument does not represent a matrix of transmission lines.</p>		

<b>Python Syntax</b>	Stretch()
<b>Python Example</b>	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Stretch(oData1, 3.1)</pre>

<b>VB Syntax</b>	Stretch
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData2 = oNDE.Stretch oData1, 3.1</pre>

**Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.**

## Terminate

Creates a new network data with specified ports terminated.

UI Access	From the <b>Network Data Explorer</b> tab, select <b>Transforms</b> in the <b>NDE</b> ribbon. Then select <b>Terminate</b> from the drop-down menu.		
Parameters	Name	Type	Description
	<oData>	Object	Data from this object is copied to a new network data and the copy is transformed.
	<portNumbers>	Array of Integers	The port numbers that are terminated (integers between 1 and $n$ ).
Return Value	<termImpedances>	Array of Complex Numbers	The impedance values used to terminate the specified ports.
	Network IDispatch.  <b>Note:</b> Terminate returns <b>None</b> if the port numbers are not valid or there are no impedance value for the port numbers.		

Python Syntax	Terminate()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Terminate(oData1, [2, 3], [23, "10+2i"])</pre>

<b>VB Syntax</b>	Terminate
<b>VB Example</b>	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData2 = oNDE.Terminate oData1, Array(2, 3), Array(23, "10+2i")</pre>

# 25 - ComplInstance Script Commands

ComplInstance commands should be executed by the **oDesign2** or **oPropHost2** object.

```
Set oDesign2 = ComplInstance.GetParentDesign  
Set oPropHost2 = ComplInstance.GetPropHost
```

## Callback Scripting Using ComplInstance Object

Callback scripts are scripts that can be set in the Property Dialog for individual properties by clicking the button in the Callback column and choosing a script that is saved with the project. Callback scripts can contain any legal script commands including general Ansys script function calls ( e.g., GetApplicationName() ). In addition, Callback scripts can also call functions on a special object named ComplInstance.

You can obtain an interface to a ComplInstance in a schematic or layout by calling `oEditor.GetComplInstanceFromRefDes(refDes)`. For more information see Layout Scripting and [Schematic Scripting](#). This interface is also available as a ComplInstance object in [ComplInstance event callbacks](#), such as placing a component in a layout or schematic.

### Definitions

**<propName>** = text string

**<value>** = double

**<valueText>** = text string

**<fileName>** = full path file name

**<choices>** = string containing menu choices separated by commas

**<initialChoice>** = string containing initial choice for menu; must be one of the <choices>

**<scriptName>** = string containing name of script stored in project

**<bool>** is 1 for true or 0 for false

<editorName> is either "Layout" or "SchematicEditor"

The topics for this section include:

[ComInstance Functions](#)

## ComInstance Functions

Following are commands that can be used to manipulate properties from a ComInstance script.

The topics for this section include:

[GetComponentName](#)

GetEditor

[GetInstanceID](#)

[GetInstanceName](#)

[GetParentDesign](#)

[GetPropHost](#)

[GetPropServerName](#)

### GetComponentName

Returns the name of the component corresponding to this ComInstance.

UI Access	NA		
Parameters	Name	Type	Description

	None		
<b>Return Value</b>	String name of component (e.g. MS_TRL) and stores it in "name"		

<b>Python Syntax</b>	GetComponentName()
<b>Python Example</b>	name = CompInstance.GetComponentName()

<b>VB Syntax</b>	GetComponentName()
<b>VB Example</b>	name = CompInstance.GetComponentName()

## GetInstanceID [Component Instance]

*Use:* Returns the instanceID of the ComplInstance.

*Command:* None

*Syntax:* GetInstanceID()

*Return Value:* String

*VB Example:* id = CompInstance.GetInstanceID();

Returns id of compInstance (e.g. 7) and stores it in "id".

Note that this is not the same number as the one used in the RefDes.

<b>Python Syntax</b>	GetInstanceID()
----------------------	-----------------

**Python Example**

```
id = CompInstance.GetInstanceID()
```

## **GetInstanceName [Component Instance]**

*Use:* Returns the instance name of the component corresponding to this ComplInstance.

*Command:* None

*Syntax:* GetInstanceName()

*Return Value:* String

*VB Example:* name = CompInstance.GetInstanceName();

Returns instanceName (e.g. A7) of complInstance and stores it in "name".

Note that the Instance Name is not the same as the RefDes.

## **GetParentDesign**

*Use:* Returns an interface to the complInstance's parent design.

*Command:* None

*Syntax:* GetParentDesign()

*Return Value:* Returns interface to design.

*Example:* Set oDesign2 = CompInstance.GetParentDesign();

Returns the interface to the design containing the complInstance.

This interface can be used to call Design functions. See: [Design Object Script Commands](#).

<b>Python Syntax</b>	GetParentDesign()
<b>Python Example</b>	<code>oDesign2 = CompInstance.GetParentDesign()</code>

## GetPropHost

*Use:* Returns an interface to the PropHost of the Complnstance, which gives access to its properties.

*Command:* None

*Syntax:* GetPropHost()

*VB Example:* Set oPropHost2 = CompInstance.GetPropHost();

Returns the interface to the properties of the complnstance.

This interface can be used to call PropHost functions; for more information see [Callback Scripting Using PropHost Object](#).

<b>Python Syntax</b>	GetPropHost()
<b>Python Example</b>	<code>oPropHost2 = CompInstance.GetPropHost()</code>

## GetPropServerName

*Use:* Returns the PropServerName of the Component corresponding to this Complnstance.

*Command:* None

*Syntax:* GetPropServerName()

*Return Value:* String

*VB Example:*

```
name = CompInstance.GetPropServerName();
```

Returns propserver name of compInstance (e.g., CompInst@MS\_TRL;7) and stores it in "name".

<b>Python Syntax</b>	GetPropServerName()
<b>Python Example</b>	name = CompInstance.GetPropServerName()

# 26 - Schematic Scripting

The Schematic scripting interface is a set of commands that match the data changing methods available in the UI of the Schematic, plus some selection and query methods. Examples of the commands available through the scripting interface are adding items, removing items, and modifying items on the Schematic. Identifying objects on the Schematic is done by a unique ID that is generated and returned by all methods that add objects, and by the FindElements and GetSelections methods. This ID can then be passed into commands to modify or remove Schematic objects.

- Since most Schematic objects can have sub-components (such as property displays or segments on a wire, etc), IDs can also be in the format of "TopLevelID:SubComponentIndex".
- The SubComponentIndex is a one-based index into the sub components of the item thus making the second segment of component CAP1 identified by "CAP1:2".
- A SubComponentIndex of zero will refer to the TopLevel item only.
- When an ID is mentioned later in this document, it will refer to either a simple string ID representing a top-level Schematic item or a "TopLevelID:SubComponentIndex" pair unless otherwise stated.
- If a specified ID or SubComponentIndex does not exist, then the function will ignore that entry and proceed with other specified IDs. Even if no valid IDs are specified, the functions will still return success.

**The topics for this section include:**

- [Method Format](#)
- [Editor Scripting IDs](#)
- [Create Method List](#)
- [General Method List](#)
- [Property Method List](#)
- [Information Method List](#)

## Method Format

In the following formats, [Opt=x] appearing after a parameter indicates that the parameter is optional and the default value is x.

When calling object-creation methods as functions, parenthesis are required in order to retrieve the name of the created object. Parenthesis are also required in JavaScript. When using VB/VBScript to call a method as a subroutine, however, parenthesis are not required.

All methods to create Schematic objects have the following form:

```
Create[type] (VARIANT parameters,  
// Array of type specific parameters  
VARIANT attributes,  
// attributes is in the format of:  
Array("NAME:Attributes", _  
"Page:=", int, _ // [Opt=1] Page number (one-based)  
// Note: Page 1 always exists  
"X:=", double, _ // X position of the object  
"Y:=", double, _ // Y position of the object  
"Angle:=", double, _ // Rotation angle (radians)  
"Flip:=", bool) // True if mirrored  
[out,retval] string id)
```

The algorithm used in the create methods is:

- 1) Create SchAdd[type]Command object with parameters passed in
-

## 2) Execute the command

All methods to modify Schematic objects have the following form:

```
[modification] (Array ("NAME:Selections",
"Page:=", Page number [optional: Default=1]
"Selections:=", IDs to modify see format below),
VARIANT params
// Array of modification specific parameters)
```

The algorithms in these types of methods will be:

- 1) Go through ids and get the SelectableObjs they correspond to
- 2) Select the SelectableObjs found in (1)
- 3) Select any sub components specified
- 4) Create the Sch[*modification*]Command
- 5) Execute the command

### ID Format

When IDs are passed to a function it can be in one of the following formats:

- Array of integers of top-level items only
- String of IDs separated by spaces or commas
- Array of strings with each string element being an ID

### Point Format

---

When a method takes an array of points, each element in the array is a string in the format of: "x y", "x,y", or "(x,y)".

## Editor Scripting IDs

Objects in the schematic are identified by text IDs. These IDs are used in scripts to perform actions on objects. These IDs are returned by all methods that add objects, and by the FindElements and GetSelections methods. The IDs are then passed into commands to modify or remove Schematic objects.

**Format of IDs for different schematic objects:**

**Components:** ComInst@<CompName>;<comInstID>;<optional schematicID>

**Geometric primitive:** SchObj@<schematicID>

**Global Port/ground ::** GPort@<portName>;<schematicID>

**Interface Port ::** IPort@<portName>;<schematicID>

**Page Port:** PagePort@<portName>;<schematicID>

**Wire :** Wire@<netName>;<schematicID>;<segment index list>

**Find this information in the Properties Window for the selected object as follows:**

<CompName> General tab/CompName

<comInstID> General tab/ID

<schematicID> Symbol tab/SchematicID

<portName> Param Values tab/PortName

<netName> General tab/NetName

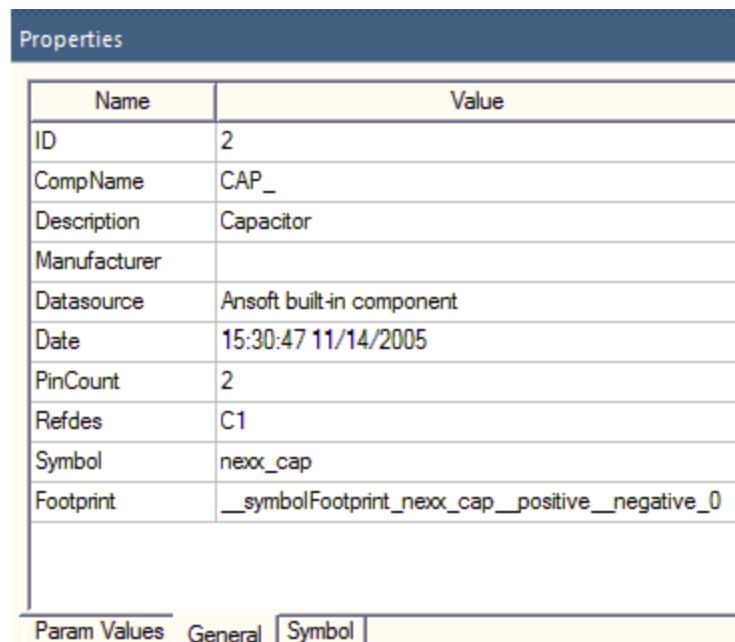
<segment index list> numbers separated by commas; Symbol tab/Segmentn - e.g. Segment0 refers to index "0"

In the Layout, the ID for a selected object is always the Name or Net property in the Footprint tab of the Properties Window

## Format for Components

```
Array ("CompInst@CAP_ ; 2 ; 4") )
```

The format for Components is ComplInst@<CompName>;<complInstID>;<optional schematicID>. In the documented example, <CompName> stands for the component name CAP\_, <complInstID> stands for its ID 2, and <optional schematicID> is the schematic ID 4. If you select the component, the **Properties** window gets updated and displays its details. For example, the selected object's component name and ID appear in the **General** tab of the **Properties** window as shown below.



The SchematicID is shown in the **Symbol** tab.

## Create Method List

This section lists the scripts that are available in the Schematic scripting interface to create and change data.

### Script Position Parameters

In Create Method scripts, X and Y position parameters are expressed in meters.

- If your layout grid units are metric, you may enter the X and Y positions directly into a script as parameters.
- If your layout grid is expressed in mils, you must first convert the component's positional coordinates to meters.

To convert an X or Y position to meters based on the minor grid size:

$$\text{X or Y position in meters} = \text{desired\_position} \times (0.00254 / \text{minor\_grid\_size})$$

where desired\_position and minor\_grid\_size have the same units. For example, to position an object at 500mm with a minor grid size of 20mm:

$$\text{X or Y position in meters} = 500 \times (0.00254 / 20)$$

This section lists the following commands:

[CreateArc \(Schematic Editor\)](#)

[CreateCircle \(Schematic Editor\)](#)

[CreateComponent \(Schematic Editor\)](#)

[CreateCurve \(Schematic Editor\)](#)

[CreateGlobalPort \(Schematic Editor\)](#)

[CreateGround \(Schematic Editor\)](#)

[CreateLine \(Schematic Editor\)](#)

[CreatePagePort \(Schematic Editor\)](#)

[CreatePort \(Schematic Editor\)](#)[CreatePolygon \(Schematic Editor\)](#)[CreateRectangle \(Schematic Editor\)](#)[CreateText \(Schematic Editor\)](#)[CreateWire \(Schematic Editor\)](#)

## CreateArc (Schematic Editor)

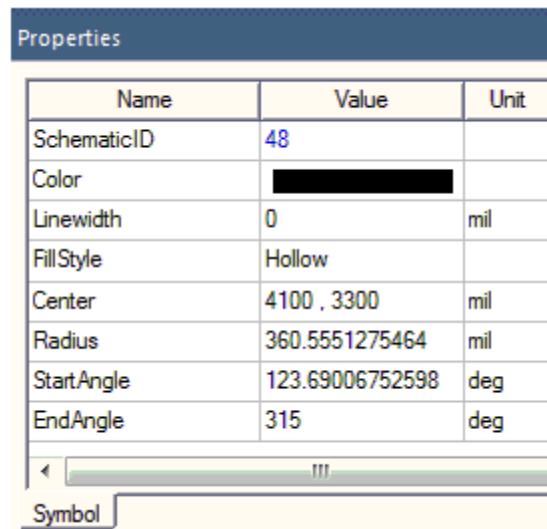
Create an arc

UI Access	Draw>Primitive>Arc		
Parameters	Name	Type	Description
	<ArcData>	Array	<pre>Array ("NAME:ArcData" _ "x:=", double, _ // X position of the object "y:=", double, _ // Y position of the object "Radius:=", double, _ // Radius of the circle "StartAng:=", double, _ // Start angle of the arc (radians) "EndAng:=", double, _ // End angle of the arc (radians) "LineWidth:=", double, // the width of the line, in meters "Color:=", int, // the RGB value of the arc color "Id:=", int), // [Opt=New id] Id for this item</pre>
	<Attributes>	Array	<pre>Array ("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)</pre>

<b>Return Value</b>	String Unique id
---------------------	---------------------

Arc is created with the format SchObj@<schematicID>

The Schematic ID can be found on the **Symbols** tab of the **Properties** window when you select the arc.



<b>Python Syntax</b>	CreateArc()
<b>Python Example</b>	<code>oEditor.CreateArc</code>

```
(["NAME:ArcData", _  
"X:=", -0.004318, "Y:=", -0.00127, _  
"Radius:=", 0.00297299377732279, _  
"StartAng:=", 1.9195673303788, _  
"EndAng:=", 3.32144615338227, _  
"Id:=", 10], _  
["NAME:Attributes", "Page:=", 1])
```

<b>VB Syntax</b>	CreateArc()
<b>VB Example</b>	<pre>oEditor.CreateArc  Array("NAME:ArcData", _ "X:=", -0.004318, "Y:=", -0.00127, _ "Radius:=", 0.00297299377732279, _ "StartAng:=", 1.9195673303788, _ "EndAng:=", 3.32144615338227, _ "Id:=", 10), _ Array("NAME:Attributes", "Page:=", 1)</pre>

**VB Example:**

```
Dim oAnsoftApp
```

```
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject("Project4")
Set oDesign = oProject.SetActiveDesign("Circuit1")
Set oEditor = oDesign.SetActiveEditor("SchematicEditor")
Dim ArcID
ArcID = oEditor.CreateArc(Array("NAME:ArcData", "X:=", 0.10414, "Y:=", 0.08382, "Radius:=", _
0.00915810023967853, "StartAng:=", 2.15879893034246, "EndAng:=", _
5.49778714378214, "LineWidth:=", 0, "Color:=", 0, "Id:=", 48), Array("NAME:Attributes", "Page:=", _
-
1))
MsgBox "Arc ID = " & ArcID
```

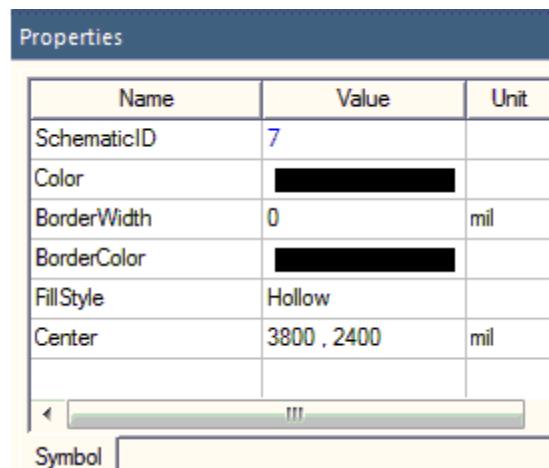
## CreateCircle (Schematic Editor)

Create a circle

UI Access	Draw>Primitive>Circle		
<b>Parameters</b>	Name	Type	Description
	<CircleData>	Array	<pre>Array ("NAME:CircleData" _       "x:=", double, _ // X position of the object       "y:=", double, _ // Y position of the object       "Radius:=", double, _ // Radius of the circle       "LineWidth:=", double, // the width of the circle border, in meters       "Bordercolor:=", int, // the RGB value of the border color       "Fill:=", int, // the fill pattern id, 0 = hollow, 1 = solid, 2 = NEDiagonal,       3 = OrthoCross, 4 = DiagCross, 5 = NWDiagonal, 6 = Horizontal, 7 = Vertical       "Color:=", int, // the RGB value of the circle fill color       "Id:=", int), // [Opt=New id] Id for this item</pre>
	<Attributes>	Array	<pre>Array ("NAME:Attributes", _       "Page:=", int) _ // [Opt=1] Page number (one-based)       [out,retval] string id)</pre>
<b>Return Value</b>	String Unique id		

Circle is created and the returned value has the format SchObj@<schematicID>

The schematic ID of the circle can be located in the **Properties** window on the **Symbols** tab.



<b>Python Syntax</b>	CreateCircle()
<b>Python Example</b>	<pre>oEditor.CreateCircle(["NAME:CircleData", _     "X:=", -0.004572, "Y:=", -0.000508, _     "Radius:=", 0.001778, "Id:=", 12], _     ["NAME:Attributes", "Page:=", 1])</pre>

<b>VB Syntax</b>	CreateCircle()
------------------	----------------

<b>VB Example</b>	<pre> oEditor.CreateCircle Array("NAME:CircleData", _ "X:=", -0.004572, "Y:=", -0.000508, _ "Radius:=", 0.001778, "Id:=", 12), _ Array("NAME:Attributes", "Page:=", 1) </pre>
-------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## CreateComponent (Schematic Editor)

Creates a component of a given type.

<b>UI Access</b>	N/A.		
<b>Parameters</b>	Name <code>&lt;params&gt;</code>	Type Array	Description Structured array.  <code>Array("NAME:ComponentProps",</code> <code>    "Name:=", &lt;String path to component in library&gt;,</code> <code>    "Id:=", &lt;String Component ID&gt;)</code>
	<code>&lt;attributes&gt;</code>	Array	Structured array.  <code>Array("NAME:Attributes",</code> <code>    "Page:=", &lt;Integer: page number&gt;,</code>  <b>Note:</b> For the page number, page 1 always exists.  <code>    "X:=", &lt;Double: X position of object&gt;,</code> <code>    "Y:=", &lt;Double: Y position of the object&gt;,</code>

			<pre>"Angle:=", &lt;Double: rotation angle in degrees&gt;, "Flip:=", &lt;boolean: True if mirrored; else False&gt;, "PinNumber:=", &lt;Integer: index of pin in symbol 0 to n-1&gt;</pre> <p><b>Note:</b></p> <p>PinNumber is optional and is set to -1 by default. PinNumber affects where the symbol is placed relative to the X and Y parameters. If PinNumber is set to -1, the symbol is placed with its center at (X,Y). If it is a pin index, the symbol is placed so that the pin is located at (X,Y). The PinNumber is not recorded.</p>
<b>Return Value</b>	<p>String containing unique ID in the format:</p> <pre>CompInst@&lt;CompName&gt;;&lt;compInstID&gt;;&lt;optional schematicID&gt;</pre> <p>For example, the return value CompInst@CAP; 4; 35 indicates that the component name is CAP, its ID is 4, and its Schematic ID is 35.</p> <p>You can also see these details in the <b>Properties</b> window when you select the created component.</p>		

<b>Python Syntax</b>	CreateComponent(<params>, <attributes>)
<b>Python Example</b>	<pre>oEditor.CreateComponent(     [         "NAME:ComponentProps",         "Name:=", "Maxwell Circuit Elements\Passive Elements:Res",</pre>

```

    "Id:=", "1"
],
[ "NAME:Attributes",
  "Page:=", 1,
  "X:=", -0.01524,
  "Y:=", 0.00508,
  "Angle:=", 0,
  "Flip:=", false
]
)

```

<b>VB Syntax</b>	CreateComponent <params>, <attributes>
<b>VB Example</b>	<pre> Set oProject = oDesktop.SetActiveProject("Project1") Set oDesign = oProject.SetActiveDesign("Circuit1") Set oEditor = oDesign.SetActiveEditor("SchematicEditor") Dim CompID  CompID = oEditor.CreateComponent Array("NAME:ComponentProps", "Name:=", "Maxwell Circuit Elements\Passive Elements:Res", "Id:=", "1"), Array("NAME:Attributes", "Page:=", 1, "X:=", -0.01524, "Y:=", 0.00508, "Angle:=", 0, "Flip:=", false) </pre>

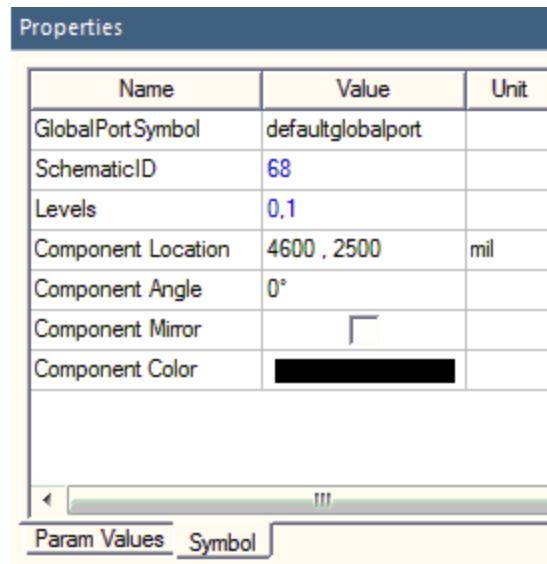
## CreateGlobalPort (Schematic Editor)

Create a global port and enable placement on the schematic

<b>UI Access</b>	Draw>Global Port		
<b>Parameters</b>	Name	Type	Description
	<portname>	String	Name for this port - leave empty to use default unique name
	<id>	Integer	ID number for the port - leave empty to use default (recommended)
	<pagenumber>	Integer	The (1-based) page number of the schematic page this port should be added to - leave empty to use default (1)
	<xval>	Double	X position of the object
	<yval>	Double	Y position of the object
	<angle>	Double	Rotation of the object, in radians. Rotations of multiples of 90 degrees are valid. Other values will be adjusted to the nearest valid angle.
	<degrees>	Double	Rotation of the object, in degrees. Rotations of multiples of 90 degrees are valid. Other values will be adjusted to the nearest valid angle
	<flip>	Boolean	True if mirrored
<b>Return Value</b>	String PortID, the identifier that can be used for this port in script operations involving this schematic		

Global port is created and the return value has the format GPort@<portName>;<schematicID>

The Schematic ID can be found in the **Properties** window on the **Symbols** tab when you select the global port.



<b>Python Syntax</b>	CreateGlobalPort(["NAME:GlobalPortProps", "Name:=", <portname>, "Id:=", <id>], ["NAME:Attributes", "Page:=", <pagenumber>, "X:=", <xval>, "Y:=", <yval>, "Angle:=", <angle>, [or "Degrees:=", <degrees>,] "Flip:=", <flip>])
<b>Python Example</b>	<pre> oEditor.CreateGlobalPort ( [ "NAME:GlobalPortProps", "Name:=", "G_0", "Id:=", 1 ],_ [ "NAME:Attributes", "Page:=", 1, "X:=", -0.02286, "Y:=",_ 0.00254, "Angle:=", 0, "Flip:=", false] </pre>

<b>VB Syntax</b>	CreateGlobalPort(Array("NAME:GlobalPortProps", "Name:=", <portname>, "Id:=", <id>), Array("NAME:Attributes", "Page:=", <pagenumber>, "X:=", <xval>, "Y:=", <yval>, "Angle:=", <angle>, [or "Degrees:=", <degrees>], "Flip:=", <flip>))
<b>VB Example</b>	<pre>oEditor.CreateGlobalPort Array ("NAME:GlobalPortProps", "Name:=", "G_0", "Id:=", 1),_ Array ("NAME:Attributes", "Page:=", 1, "X:=", -0.02286, "Y:=", 0.00254, "Angle:=", 0, "Flip:=", false)</pre>

**VB Example:**

```
Dim oAnsoftApp  
Dim oDesktop  
Dim oProject  
Dim oDesign  
Dim oEditor  
Dim oModule  
Set oAnsoftApp = CreateObject ("Ansoft.ElectronicsDesktop")  
Set oDesktop = oAnsoftApp.GetAppDesktop ()  
oDesktop.RestoreWindow  
Set oProject = oDesktop.SetActiveProject ("Project4")  
Set oDesign = oProject.SetActiveDesign ("Circuit1")  
Set oEditor = oDesign.SetActiveEditor ("SchematicEditor")
```

```

Dim GPort

GPort = oEditor.CreateGlobalPort(Array("NAME:GlobalPortProps", "Name:=", "G_1", "Id:=", 68),
Array("NAME:Attributes", "Page:=", _
1, "X:=", 0.11684, "Y:=", 0.0635, "Angle:=", 0, "Flip:=", false))

MsgBox "Port = " & GPort

```

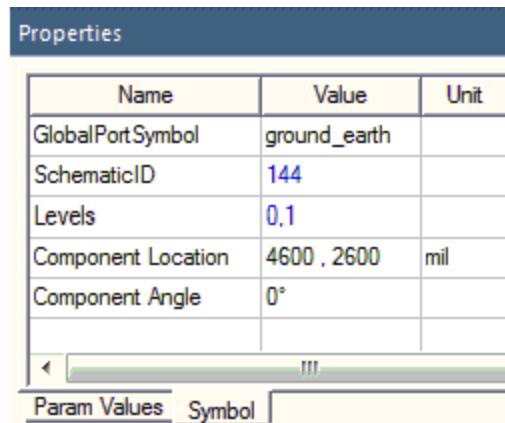
## CreateGround (Schematic Editor)

Create a ground

UI Access	Draw>Ground		
Parameters	Name <GroundProps>	Type Array	Description Array ("NAME:GroundProps", _ "Id:=", int), // [Opt=New id] Id for this item
	<Attributes>	Array	Array ("NAME:Attributes", _ "Page:=", int, _ // [Opt=1] Page number (one-based)  Note: Page 1 always exists  "X:=", double, _ // X position of the object  "Y:=", double, _ // Y position of the object  "Angle:=", double, _ // Rotation angle (radians)  "Flip:=", bool), // True if mirrored  [out,retval] string id)
Return Value	String Unique id		

Ground is created and the returned value has the format GPort@<portname>;<schematicID>

The Schematic ID for ground can be found on the **Symbols** tab of the **Properties** window when you select the object.



<b>Python Syntax</b>	CreateGround()
<b>Python Example</b>	<pre>oEditor.CreateGround (["NAME:GroundProps", "Id:=", 8],  ["NAME:Attributes", "Page:=", 1, "X:=",  -0.04064, "Y:=", -0.00254, "Angle:=",  0, "Flip:=", false])</pre>

<b>VB Syntax</b>	CreateGround()
<b>VB Example</b>	<pre> oEditor.CreateGround Array("NAME:GroundProps", "Id:=", 8), Array("NAME:Attributes", "Page:=", 1, "X:=", -0.04064, "Y:=", -0.00254, "Angle:=", 0, "Flip:=", false) </pre>

***VB Example:***

```

Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject("Project4")
Set oDesign = oProject.SetActiveDesign("Circuit1")
Set oEditor = oDesign.SetActiveEditor("SchematicEditor")

```

```
Gnd = oEditor.CreateGround(Array("NAME:GroundProps", "Id:=", 144), Array("NAME:Attributes",
"Page:=", _
1, "X:=", 0.11684, "Y:=", 0.06604, "Angle:=", 0, "Flip:=", false))

MsgBox "GndIndex =" & Gnd
```

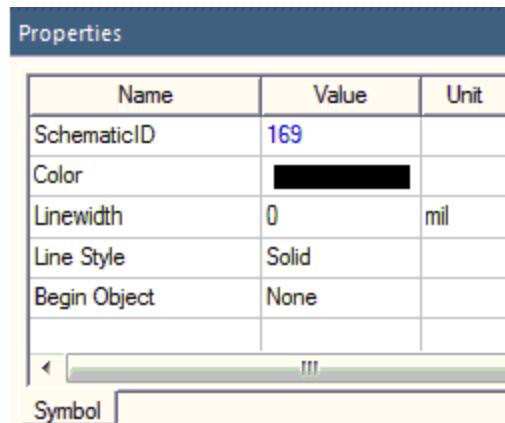
## CreateLine (Schematic Editor)

Creates a line.

UI Access	Draw>Primitive>Line		
Parameters	Name <LineData>	Type Array	Description Array ("NAME:LineData" _ "Points:=", Array of points, _ "LineWidth:=", double, // the width of the line, in meters "Color:=", int, // the RBG value of the line color "Id:=", int), // [Opt=New id] Id for this item
	<Attributes>	Array	Array ("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)
Return Value	String Unique id		

Line is created and the returned value has the format SchObj@<schematicID>

The Schematic ID can be found on the **Symbol** tab of the **Properties** window when you select the line.



<b>Python Syntax</b>	CreateLine()
<b>Python Example</b>	<pre>oEditor.CreateLine ( ["NAME:LineData",     "Points:=", [ "(-0.055652, 0.020669)", _     "(-0.036923, 0.011257)", "(-0.023144, 0.018049)"], _     "LineWidth:=", 0, "Color:=", 0, "Id:=", 22)], _     ["NAME:Attributes", "Page:=", 1])</pre>

<b>VB Syntax</b>	CreateLine()
<b>VB Example</b>	<pre>oEditor.CreateLine Array("NAME:LineData",_ "Points:=", Array("(-0.055652, 0.020669)", _  "(-0.036923, 0.011257)", "(-0.023144, 0.018049)"),_ "LineWidth:=", 0, "Color:=", 0, "Id:=", 22), _ Array("NAME:Attributes", "Page:=", 1)</pre>

**VB Example:**

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject("Project4")
Set oDesign = oProject.SetActiveDesign("Circuit1")
Set oEditor = oDesign.SetActiveEditor("SchematicEditor")
LineID = oEditor.CreateLine(Array("NAME:LineData", "Points:=", Array("(0.050800, 0.106680)", _
```

```

" (0.124460, 0.106680), "(0.124460, 0.106680)", "LineWidth:=", 0, "Color:=", _
0, "Id:=", 169), Array("NAME:Attributes", "Page:=", 1))

MsgBox "LineID = " & LineID

```

## CreatePagePort (Schematic Editor)

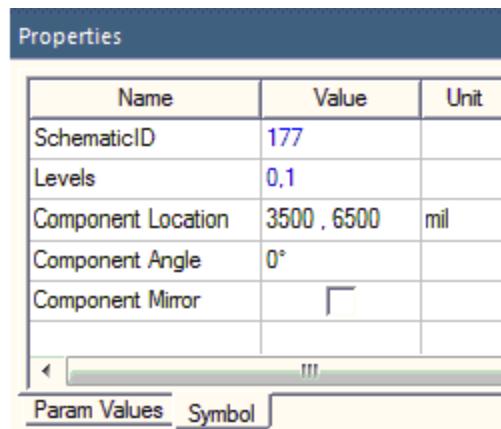
Create a PagePort

UI Access	Draw>Page Connector		
Parameters	Name	Type	Description
	<PagePortProps>	Array	<p>Array ("NAME:PagePortProps"),</p> <p>Array ("NAME:Attributes", _</p> <p>"Page:=", int, _ // [Opt=1] Page number (one-based)</p> <p>Note: Page 1 always exists</p> <p>"X:=", double, _ // X position of the object</p> <p>"Y:=", double, _ // Y position of the object</p> <p>"Angle:=", double, _ // Rotation angle (radians)</p> <p>"Flip:=", bool), // True if mirrored</p> <p>[out,retval] string id)</p>
Return Value	String Unique id		

The pageport is created and it has the following format:

PagePort@<portName>;<schematicID>

The schematic ID can be found in the **Properties** window on the **Symbols** tab when you select the pageport.



<b>Python Syntax</b>	CreatePagePort()
<b>Python Example</b>	<pre>oEditor.CreatePagePort ([{"NAME": "PagePortProps", "Name": "pageport_0", "Id": "12"], [{"NAME": "Attributes", "Page": "1", "X": "-0.0381", "Y": "-0.0127", "Angle": "0", "Flip": false}])</pre>

<b>VB Syntax</b>	CreatePagePort()
<b>VB Example</b>	<pre> oEditor.CreatePagePort  Array("NAME:PagePortProps", "Name:=", "pageport_0", "Id:=", _ 12), Array("NAME:Attributes", "Page:=", _ 1, "X:=", -0.0381, "Y:=", -0.0127, "Angle:=", _ 0, "Flip:=", false) </pre>

***VB Example:***

```

Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule

Set oAnsoftApp = CreateObject ("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject("Project4")
Set oDesign = oProject.SetActiveDesign("Circuit1")

```

```
Set oEditor = oDesign.SetActiveEditor("SchematicEditor")
PagePortID = oEditor.CreatePagePort(Array("NAME:PagePortProps", "Name:=", "pageport_0", "Id:=", _
177), Array("NAME:Attributes", "Page:=", 1, "X:=", 0.0889, "Y:=", 0.1651, "Angle:=", _
0, "Flip:=", false))
MsgBox "PagePort = " & PagePortID
```

## CreateIPort (Schematic Editor)

Create an interface port

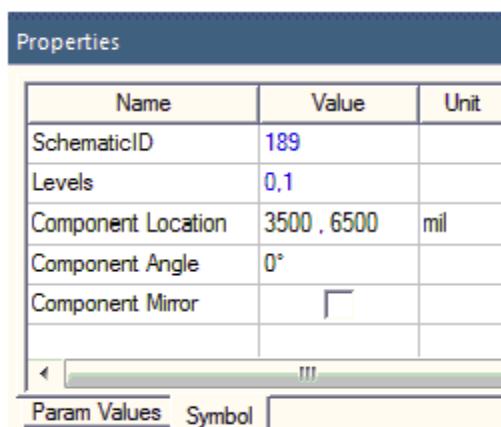
UI Access	Draw>Interface Port		
Parameters	Name <IPortProps>	Type Array	Description Array ("NAME:IPortProps"), "Name:=", string), // [Opt=default name] Name for this port "Id:=", int), // Port id number
	<Attributes>	Array	Array ("NAME:Attributes", _ "Page:=", int, _ // [Opt=1] Page number (one-based)

			<p><b>Note:</b></p> <p>Page 1 always exists</p> <p>"X:=", double, _ // X position of the object</p> <p>"Y:=", double, _ // Y position of the object</p> <p>"Angle:=", double, _ // Rotation angle (radians)</p> <p>"Flip:=", bool), // True if mirrored</p> <p>[out,retval] string id)</p>
<b>Return Value</b>	String Unique id		

Interface port is created and the returned value has the following format:

IPort@<portName>;<schematicID>

The interface port schematic ID can be found in the **Properties** window on the **Symbols** tab.



<b>Python Syntax</b>	CreateIPort()
<b>Python Example</b>	<pre>oEditor.CreateIPort ([ "NAME:IPortProps", "Name:=", _ "Port1", "Id:=", 4], [ "NAME:Attributes", "Page:=", _ 1, "X:=", -0.0381, "Y:=", 0.0127, "Angle:=", _ 0, "Flip:=", false])</pre>

<b>VB Syntax</b>	CreateIPort()
<b>VB Example</b>	<pre>oEditor.CreateIPort Array("NAME:IPortProps", "Name:=", _ "Port1", "Id:=", 4), Array("NAME:Attributes", "Page:=", _ 1, "X:=", -0.0381, "Y:=", 0.0127, "Angle:=", _ 0, "Flip:=", false)</pre>

*VB Example:*

```
Dim oAnsoftApp  
Dim oDesktop  
Dim oProject  
Dim oDesign  
Dim oEditor  
Dim oModule
```

```

Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject("Project4")
Set oDesign = oProject.SetActiveDesign("Circuit1")
Set oEditor = oDesign.SetActiveEditor("SchematicEditor")
IPortID = oEditor.CreateIPort(Array("NAME:IPortProps", "Name:=", "Port1", "Id:=", 189), Array
("NAME:Attributes", "Page:=", _
1, "X:=", 0.0889, "Y:=", 0.1651, "Angle:=", 0, "Flip:=", false))
MsgBox "IPort ID = " & IPortID

```

## CreatePolygon (Schematic Editor)

Create a polygon

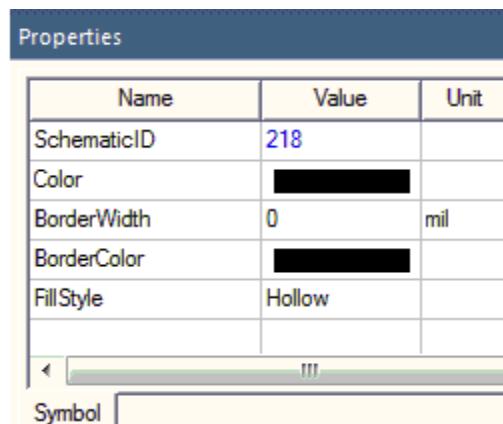
UI Access	Draw>Primitive>Polygon		
Parameters	Name	Type	Description
	<PolygonData>	Array	Array ("NAME:PolygonData" _ "Points:=", Array of points, _ "LineWidth:=", double, // the width of the Polygon border, in meters "Bordercolor:=", int, // the RGB value of the border color "Fill:=", int, // the fill pattern id, 0 = hollow, 1 = solid, 2 = NEDiagonal, 3 = OrthoCross, 4 = DiagCross, 5 = NWDiagonal, 6 = Horizontal, 7 = Vertical "Color:=", int, // the RGB value of the Polygon fill color "Id:=", int), // [Opt=New id] Id for this item

	<Attributes>	Array	Array ("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)
Return Value	String Unique id		

Polygon is created and the return value has the format

SchObj@<schematicID>

The Schematic ID can be found in the **Properties** window on the **Symbol** tab when you select the polygon.



<b>Python Syntax</b>	CreatePolygon()
<b>Python Example</b>	<pre> oEditor.CreatePolygon(     ["NAME:PolygonData", "Points:=", [ _          "(-0.058951, 0.006308)", "(-0.046142, 0.010480)", _          "(-0.046239, 0.001067)"), "LineWidth:=", _          0, "BorderColor:=", 0, "Fill:=", 0, "Color:=", _          0, "Id:=", 27], ["NAME:Attributes", "Page:=", 1]]) </pre>

<b>VB Syntax</b>	CreatePolygon()
<b>VB Example</b>	<pre> oEditor.CreatePolygon     Array("NAME:PolygonData", "Points:=", Array( _         "(-0.058951, 0.006308)", "(-0.046142, 0.010480)", _          "(-0.046239, 0.001067)"), "LineWidth:=", _          0, "BorderColor:=", 0, "Fill:=", 0, "Color:=", _          0, "Id:=", 27), Array("NAME:Attributes", "Page:=", 1) </pre>

**VB Example:**

```

Dim oAnsoftApp
Dim oDesktop

```

```
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject("Project4")
Set oDesign = oProject.SetActiveDesign("Circuit1")
Set oEditor = oDesign.SetActiveEditor("SchematicEditor")
polygonID = oEditor.CreatePolygon(Array("NAME:PolygonData", "Points:=", Array(_
"(0.101600, 0.058420)", "(0.101600, 0.060960)", "(0.101600, 0.058420)", _(
0.099060, 0.060960)", "(0.101600, 0.060960)", "(0.101600, 0.058420)", _(
0.101600, 0.060960)", "(0.101600, 0.060960)"), "LineWidth:=", 0, "BorderColor:=", _(
0, "Fill:=", 0, "Color:=", 0, "Id:=", 218), Array("NAME:Attributes", "Page:=", 1))
MsgBox "Polygon ID = " & polygonID
```

## CreateRectangle (Schematic Editor)

Create a rectangle

UI Access

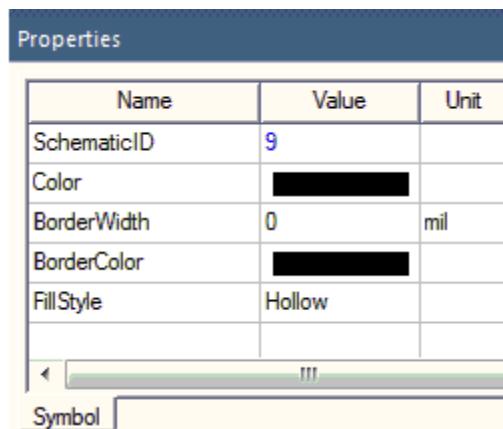
Draw>Primitive>Rectangle

	Name	Type	Description
<b>Parameters</b>	<RectData>	Array	<pre>Array ("NAME:RectData" _       "x1:=", double, _ // X position of the upper left       "y1:=", double, _ // Y position of the upper left       "x2:=", double, _ // X position of the lower right       "y2:=", double, _ // Y position of the lower right       "LineWidth:=", double, // the width of the Rectangle border, in meters       "Bordercolor:=", int, // the RBG value of the border color       "Fill:=", int, // the fill pattern id, 0 = hollow, 1 = solid, 2 = NEDiagonal,       3 = OrthoCross, 4 = DiagCross, 5 = NWDiagonal, 6 = Horizontal, 7 = Vertical       "Color:=", int, // the RBG value of the Rectangle fill color       "Id:=", int), // [Opt=New id] Id for this item</pre>
	<Attributes>	Array	<pre>Array ("NAME:Attributes",       "Page:=", int) // [Opt=1] Page number (one-based)       [out,retval] string id)</pre>
<b>Return Value</b>	String Unique id		

Rectangle is created and the return value has the format

SchObj@<schematicID>

The Schematic ID can be found in the **Properties** window on the **Symbol** tab when you select the rectangle.



<b>Python Syntax</b>	CreateRectangle()
<b>Python Example</b>	<pre>oEditor.CreateRectangle ([ "NAME:RectData", "X1:=", -0.0755449856733525, "Y1:=", _ 0.0335755491881566, "X2:=", -0.0611831900668577, "Y2:=", _ 0.0254242597898758, "LineWidth:=", _ 0, "BorderColor:=", 0, "Fill:=", 0, "Color:=", 0, "Id:=", 31], _ [ "NAME:Attributes", "Page:=", 1])</pre>

<b>VB Syntax</b>	CreateRectangle()
------------------	-------------------

**VB Example**

```
oEditor.CreateRectangle  
Array("NAME:RectData", "X1:=", -0.0755449856733525, "Y1:=", _  
0.0335755491881566, "X2:=", -0.0611831900668577, "Y2:=", _  
0.0254242597898758, "LineWidth:=", _  
0, "BorderColor:=", 0, "Fill:=", 0, "Color:=", 0, "Id:=", 31), _  
Array("NAME:Attributes", "Page:=", 1)
```

**VB Example:**

```
Dim oAnsoftApp  
Dim oDesktop  
Dim oProject  
Dim oDesign  
Dim oEditor  
Dim oModule  
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")  
Set oDesktop = oAnsoftApp.GetAppDesktop()  
oDesktop.RestoreWindow  
Set oProject = oDesktop.SetActiveProject("Project4")  
Set oDesign = oProject.SetActiveDesign("Circuit1")  
Set oEditor = oDesign.SetActiveEditor("SchematicEditor")
```

```
rectangleID = oEditor.CreateRectangle(Array("NAME:RectData", "X1:=", 0.1016, "Y1:=", 0.0635,
"X2:=", _
0.10414, "Y2:=", 0.06096, "LineWidth:=", 0, "BorderColor:=", 0, "Fill:=", 0, "Color:=", _
0, "Id:=", 9), Array("NAME:Attributes", "Page:=", 1))

MsgBox "Rectangle ID = " & rectangleID
```

## CreateText (Schematic Editor)

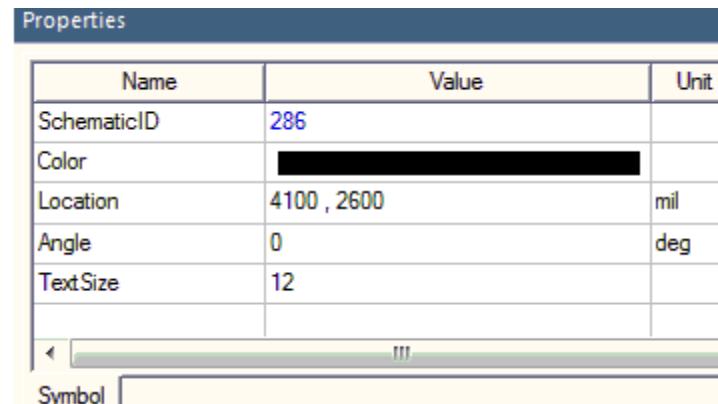
Create text

UI Access	Draw>Primitive>Text		
	Name	Type	Description
Parameters	<TextData>	Array	Array ("NAME:TextData" _ "x:=", double, _ // X position of the object "y:=", double, _ // Y position of the object "Text:=", string, _ // Text to display "Color:=", int, // the RBG value of the text color "Id:=", int), // Id for this item "ShowRect:=", bool, // true or false, whether to show the highlight rectangle for the text "X1:=", double, // the text rectangle left X value, in meters "Y1:=", double, // the text rectangle upper Y value, in meters "X2:=", double, // the text rectangle right X value, in meters "Y2:=", double, // the text rectangle lower Y value, in meters

		<pre>"RectLineWidth:=", double, // the width of the rectangle border, in meters "RectBordercolor:=", int, // the RBG value of the rectangle border color "RectFill:=", int, // the rectangle fill pattern id, 0 = hollow, 1 = solid, 2 = NEDiagonal, 3 = OrthoCross, 4 = DiagCross, 5 = NWDiagonal, 6 = Horizontal, 7 = Vertical "RectColor:=", int, // the RBG value of the rectangle fill color</pre>
	<Attributes>	Array ("NAME:Attributes", _ "Page:=", int) _ // [Opt=1] Page number (one-based) [out,retval] string id)
<b>Return Value</b>	String Unique id	

Text is created and the return value has the format: SchObj@<SchematicID>

The Schematic ID can be found in the **Properties** window on the **Symbols** tab when you select the text.



<b>Python Syntax</b>	CreateText()  oEditor.CreateText(  [ "NAME:TextData", "X:=", -0.0764183381088826, "Y:=", _ 0.040174212034384, "Size:=", 12, "Angle:=", _ 0, "Text:=", "Control Circuit", "Color:=", _ 0, "Id:=", 34, "ShowRect:=", false, "X1:=", _ -0.0793287547755609, "Y1:=", _ 0.0407033787010528, "X2:=", -0.0502245881087778, _ "Y2:=", 0.035411712034365, "RectLineWidth:=", _ 0, "RectBorderColor:=", 0, "RectFill:=", _ 0, "RectColor:=", 0], [ "NAME:Attributes", "Page:=", 1])
<b>Python Example</b>	

<b>VB Syntax</b>	CreateText()  oEditor.CreateText
<b>VB Example</b>	Array("NAME:TextData", "X:=", -0.0764183381088826, "Y:=", _ 0.040174212034384, "Size:=", 12, "Angle:=", _ 0, "Text:=", "Control Circuit", "Color:=", _

```

0, "Id:=", 34, "ShowRect:=", false, "X1:=", _
-0.0793287547755609, "Y1:=", _
0.0407033787010528, "X2:=", -0.0502245881087778,_
"Y2:=", 0.035411712034365, "RectLineWidth:=", _
0, "RectBorderColor:=", 0, "RectFill:=",_
0, "RectColor:=", 0), Array("NAME:Attributes", "Page:=", 1)

```

***VB Example:***

```

Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.SetActiveProject("Project4")
Set oDesign = oProject.SetActiveDesign("Circuit1")
Set oEditor = oDesign.SetActiveEditor("SchematicEditor")
textobj = oEditor.CreateText(Array("NAME:TextData", "X:=", 0.10414, "Y:=", 0.06604, "Size:=", _

```

```
12, "Angle:=", 0, "Text:=", "CreateDefaultTectxt" & Chr(13) & Chr(10) & "", "Color:=", _  
0, "Id:=", 286, "ShowRect:=", false, "X1:=", 0.100141851851836, "Y1:=", _  
0.0670983333333376, "X2:=", 0.140123333333477, "Y2:=", 0.0565149999999619, "RectLineWidth:=", _  
0, "RectBorderColor:=", 0, "RectFill:=", 0, "RectColor:=", 0), Array("NAME:Attributes", "Page:=", _  
-  
1)  
  
MsgBox "text = " & textobj
```

## CreateWire (Schematic Editor)

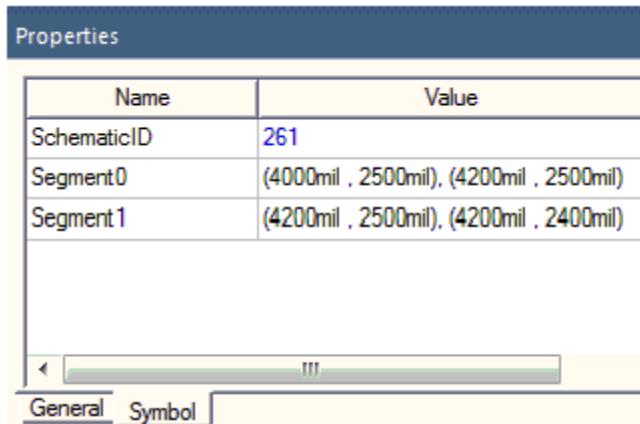
Create a wire

UI Access	Draw>Wire		
Parameters	Name <WireData>	Type Array	Description Array ("NAME:WireData" _ "Name:=", string), // [Opt=default name] Name for this wire "Id:=", int), // Wire idnum "Points:=", Points on wire)
	<Attributes>	Array	Array ("NAME:Attributes", _ "Page:=", int, _ // [Opt=1] Page number (one-based) Note: Page 1 always exists [out,retval] string id)
Return Value	String		

	Unique id
--	-----------

Wire is created and its return value has the format `Wire@<netName>;<schematicID>;<segment index list>`

The Schematic ID can be found in the **Symbol** tab of the **Properties** window when you select the wire. The wire net name appears in the **General** tab. The parameter segment index list indicates the number of segments i.e. the segment count.



<b>Python Syntax</b>	<code>CreateWire()</code>
<b>Python Example</b>	<pre> oEditor.CreateWire(     ["NAME:WireData", "Name:=", "", "Id:=", 41, "Points:=", [         "(-0.033020, 0.015240)", "(-0.017780, 0.015240)",         "(-0.017780, 0.012700)"]], ["NAME:Attributes", "Page:=", 1]) </pre>

<b>VB Syntax</b>	CreateWire()
<b>VB Example</b>	<pre>oEditor.CreateWire Array("NAME:WireData", "Name:=", "", "Id:=", 41, "Points:=", Array( _     "(-0.033020, 0.015240)", "(-0.017780, 0.015240)",     "(-0.017780, 0.012700)")), Array("NAME:Attributes", "Page:=", 1)</pre>

***VB Example:***

```
Dim oAnsoftApp  
Dim oDesktop  
Dim oProject  
Dim oDesign  
Dim oEditor  
Dim oModule  
  
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")  
Set oDesktop = oAnsoftApp.GetAppDesktop()  
oDesktop.RestoreWindow  
Set oProject = oDesktop.SetActiveProject("Project4")  
Set oDesign = oProject.SetActiveDesign("Circuit1")  
Set oEditor = oDesign.SetActiveEditor("SchematicEditor")
```

```
WireID = oEditor.CreateWire(Array("NAME:WireData", "Name:=", "", "Id:=", 261, "Points:=", Array(  
-  
"(0.101600, 0.063500)", "(0.106680, 0.063500)", "(0.106680, 0.060960)")), Array("NAME:At-  
tributes", "Page:=", _  
1))  
MsgBox "Wire ID = " & WireID
```

## General Method List

This section presents the general script methods that are available.

The general method script commands are listed here.

[Activate \(Schematic Editor\)](#)

[AddPinGrounds \(Schematic Editor\)](#)

[AddPinIPorts \(Schematic Editor\)](#)

[AddPinPageConnectors \(Schematic Editor\)](#)

[AlignHorizontal \(Schematic Editor\)](#)

[AlignVertical \(Schematic Editor\)](#)

[BringToFront \(Schematic Editor\)](#)

[Close Editor \(Schematic Editor\)](#)

[Copy \(Schematic Editor\)](#)

[CopyData \[Schematic Editor\]](#)

[CopySubdesign \[Schematic Editor\]](#)

[CreatePage \(Schematic Editor\)](#)

[Cut \(Schematic Editor\)](#)

[DeactivateOpen \(Schematic Editor\)](#)  
[DeactivateShort \(Schematic Editor\)](#)  
[Delete \(Schematic Editor\)](#)  
[DeletePage \(Schematic Editor\)](#)  
[ElectricRuleCheck \(Schematic Editor\)](#)  
[ExportImage \(Schematic Editor\)](#)  
[FindElements \(Schematic Editor\)](#)  
[FitToBorder \[Schematic Editor\]](#)  
[GridSetup \(Schematic Editor\)](#)  
[FlipHorizontal \(Schematic Editor\)](#)  
[FlipVertical \(Schematic Editor\)](#)  
[Move \(Schematic Editor\)](#)  
[NameNets \(Schematic Editor\)](#)  
[PageBorders \(Schematic Editor\)](#)  
[Pan \(Schematic Editor\)](#)  
[Paste \(Schematic Editor\)](#)  
[PasteData \[Schematic Editor\]](#)  
[PasteDesign \(Schematic Editor\)](#)  
[PushExcitations \(Schematic Editor\)](#)  
[Rotate \(Schematic Editor\)](#)

[ShowVariableBlock \(Schematic Editor\)](#)

[SelectAll \(Schematic Editor\)](#)

[SelectPage \(Schematic Editor\)](#)

[SendToBack\(Schematic Editor\)](#)

[SetPageData \[Schematic\]](#)

[SortComponents \(Schematic Editor\)](#)

[ZoomArea \(Schematic Editor\)](#)

[ZoomIn \(Schematic Editor\)](#)

[ZoomOut \(Schematic Editor\)](#)

[ZoomPrevious \(Schematic Editor\)](#)

[ZoomToFit \(Schematic Editor\)](#)

[Wire \(Schematic Editor\)](#)

## **Activate (Schematic Editor)**

Enable items previously disabled

<b>UI Access</b>	Edit>Activate								
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>Name</td><td>Type</td><td>Description</td></tr></tbody></table>			Name	Type	Description	Name	Type	Description
Name	Type	Description							
Name	Type	Description							
<b>Return Value</b>	Value								

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	Activate(["NAME:Selections","Selections:=", [<components>]])
<b>Python Example</b>	oEditor.Activate(["NAME:Selections","Selections:=", ["ComplInst@STEP;1;1"]])

<b>VB Syntax</b>	Activate Array("NAME:Selections", "Selections:=", Array("<components>"))
<b>VB Example</b>	oEditor.Activate Array("NAME:Selections", "Selections:=", Array("CompInst@STEP;1;1"))

## AddPinGrounds (Schematic Editor)

Adds grounds at all unconnected pins

```
AddPinGrounds (
    Array("NAME:Selections", _ // PagePort Name
    "Selections:=", _ // Net Name
    Array("CompInst@CAP_ ;2;4"))
```

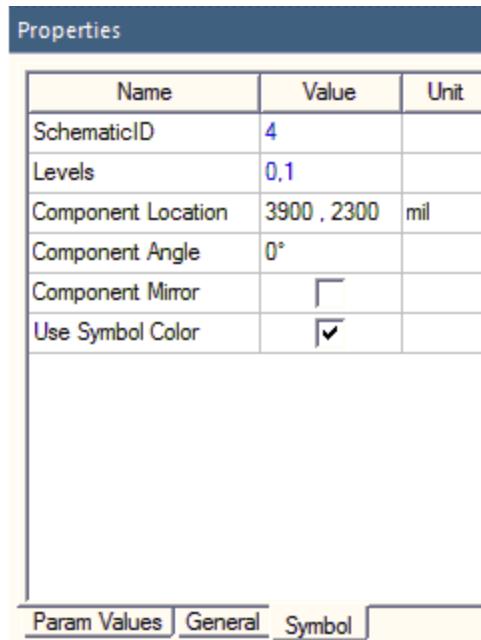
**Note:**

The format for Components is ComInst@<CompName>;<comInstID>;<optional schematicID>. In the documented example, <CompName> stands for the component name CAP\_, <comInstID> stands for its ID 2, and <optional schematicID> is the schematic ID 4. If you select the component, the **Properties** window gets updated and displays its details. For example, the selected object's component name and ID appear in the **General** tab of the **Properties** window as shown below.

Properties	
Name	Value
ID	2
CompName	CAP_
Description	Capacitor
Manufacturer	
Datasource	Ansoft built-in component
Date	15:30:47 11/14/2005
PinCount	2
Refdes	C1
Symbol	nexx_cap
Footprint	__symbolFootprint_nexx_cap_positive_negative_0

Param Values General Symbol

To view the SchematicID click the **Symbol** tab.



<b>Python Syntax</b>	AddPinPageConnectors ()
<b>Python Example</b>	<pre>oEditor.AddPinPageconnectors["NAME:Selections", "Selections:=", [_ "CompInst@C;1;1"]]</pre>

## AddPinIPorts (Schematic Editor)

Adds Interface Ports at unconnected pins of all selected components. Each port has a unique name.

<b>UI Access</b>	Draw>Add at unconnected pins>Interface Ports		
<b>Parameters</b>	Name <IntPortName,NetName>	Type Array	Description Array("NAME:Selections", _ // Interface Port Name "Selections:=", _ // Net Name
<b>Return Value</b>	None		

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	AddPinIPorts ()
<b>Python Example</b>	<pre>oEditor.AddPinIPorts ["NAME:Selections", "Selections:=", ["CompInst@R;2;37"]]</pre>

<b>VB Syntax</b>	AddPinIPorts ()
<b>VB Example</b>	<pre>oEditor.AddPinIPorts Array("NAME:Selections", "Selections:=", Array("CompInst@R;2;37"))</pre>

## AddPinPageConnectors (Schematic Editor)

Adds Page Ports at unconnected pins of all selected components

<b>UI Access</b>	Draw>Add at unconnected pins>Grounds						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>Name</td><td>Type</td><td>Description</td></tr></table>	Name	Type	Description	Name	Type	Description
Name	Type	Description					
Name	Type	Description					
<b>Return Value</b>	None						

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	AddPinPageConnectors ()
<b>Python Example</b>	<pre>oEditor.AddPinPageconnectors [ "NAME:Selections", "Selections:=", [ _ "CompInst@C;1;1" ]]</pre>

<b>VB Syntax</b>	AddPinPageConnectors ()
<b>VB Example</b>	<pre>oEditor.AddPinPageconnectors Array("NAME:Selections", "Selections:=", Array( _ "CompInst@C;1;1" ))</pre>

## AlignHorizontal (Schematic Editor)

Align items horizontally

UI Access	Draw>Align Horizontal		
Parameters	Name	Type	Description
	<Selections>	Array	Array ("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify)
	<AlignParameters>	Array	Array ("NAME:AlignParameters", _ "Disconnect:=", bool _ // [Opt=0] Should wires disconnect "Rubberband:=", bool) _ // [Opt=1] Should wires staircase Note: Alignment occurs relative to the first item in ids
Return Value	Value		

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	AlignHorizontal()
Python Example	<pre>oEditor.AlignHorizontal ([ "NAME:Selections", "Selections:=", [ _ "CompInst@R;2;4", "CompInst@C;1;1" ]], [ "NAME:AlignParameters", "Disconnect:=", _ false, "Rubberband:=", false])</pre>

<b>VB Syntax</b>	AlignHorizontal()
<b>VB Example</b>	<pre>oEditor.AlignHorizontal Array("NAME:Selections", "Selections:=", Array( _ "CompInst@R;2;4", "CompInst@C;1;1")), Array("NAME:AlignParameters", "Disconnect:=", _ false, "Rubberband:=", false)</pre>

## AlignVertical (Schematic Editor)

Align items vertically

<b>UI Access</b>	Draw>Align Vertical		
<b>Parameters</b>	Name	Type	Description
	<Selections>	Array	<pre>Array("NAME:Selections", _ "Page:=", page number, _// [Opt=1] Page number "Selections:=", IDs to modify)</pre>
	<Align Parameters>	Array	<pre>Array("NAME:AlignParameters", _ "Disconnect:=", bool _// [Opt=0] Should wires disconnect "Rubberband:=", bool) _// [Opt=1] Should wires staircase Note: Alignment occurs relative to the first item in ids</pre>
<b>Return Value</b>	None		

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	AlignVertical()
<b>Python Example</b>	<pre>oEditor.AlignVertical (["NAME:Selections", "Selections:=", ["CompInst@R;2;4", _ "CompInst@C;1;1"]], ["NAME:AlignParameters", "Disconnect:=", false, "Rubberband:=", _ false])</pre>

<b>VB Syntax</b>	AlignVertical()
<b>VB Example</b>	<pre>oEditor.AlignVertical Array("NAME:Selections", "Selections:=", Array("CompInst@R;2;4", _ "CompInst@C;1;1")), Array("NAME:AlignParameters", "Disconnect:=", false, "Rub- berband:=", _ false)</pre>

## BringToFront (Schematic Editor)

Bring the selected object to the front of the view

<b>UI Access</b>	Draw>Bring to Front
------------------	---------------------

	Name	Type	Description
<b>Parameters</b>	<Selections>	Array	<p>["NAME: Selections", "Selections:=", [&lt;Selected Components&gt;]]</p> <p>&lt;Selected Components&gt;</p> <p>ComplInst@DefName; ID, schematic ID , ComplInst@DefName; ID, schematic ID</p> <p>ComplInst@R;1,2</p>
<b>Return Value</b>	None		

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	BringToFront( )
<b>Python Example</b>	<pre>oEditor.BringToFront( ["NAME:Selections", "Selections:=", ["SchObj@1"]])</pre>

<b>VB Syntax</b>	BringToFront( )
<b>VB Example</b>	<pre>oEditor.BringToFront Array("NAME:Selections", "Selections:=", Array("SchObj@1"))</pre>

## CloseEditor (Schematic Editor)

Close an Editor

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>None</td> <td></td> <td></td> </tr> </tbody> </table>	Name	Type	Description	None		
Name	Type	Description					
None							
<b>Return Value</b>	None						

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	CloseEditor()
<b>Python Example</b>	<code>oDefinitionEditor.CloseEditor()</code>

<b>VB Syntax</b>	CloseEditor
<b>VB Example</b>	<code>oDefinitionEditor.CloseEditor</code>

## Copy (Schematic Editor)

GeneralCopy

```
// Copy items for pasting
```

```
Copy()  
  
Array("NAME:Selections", _  
"Page:=", page number, _ // [Opt=1] Page number  
"Selections:=", IDs to modify))
```

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	Copy()
<b>Python Example</b>	<pre>oEditor.Copy ([ "NAME:Selections", "Selections:=", [ "CompInst@R;2;37", _ "CompInst@C;1;1" ] ])</pre>

## CopyData [Schematic Editor]

**Use:** Copy graphic object or component properties for pasting.

**Command:** **Edit> Copy Data** (or Right-click on graphic object or component and select **Copy Data**)

**Syntax:** CopyData Array("NAME:Selections", "Selections:=", Array(<selections>))

**Return Value:** None.

**Parameters:** <selections>

Type: Comma-separated list of strings

Identifiers for each selected item to be copied, in the form CompInst@<id>, where id is the compname;ID;schematicID for each element.

```
Example: Set oEditor = oDesign.SetActiveEditor("SchematicEditor")
oEditor.CopyData Array("NAME:Selections", "Selections:=", Array("CompInst@Cap;1;1"))
```

## **CopySubdesign [Schematic Editor]**

Copy components that represent subdesigns, as well as their represented designs, for pasting. The designs, when pasted, will be independent, with unique names based on the original design.

<b>UI Access</b>	(Right-click on hierarchical component) <b>Copy as New Design</b>		
<b>Parameters</b>	Name <selections>	Type Comma-separated list of strings	Description Identifiers for each selected item to be copied, in the form CompInst@<id>, where id is the compname;ID;schematicID for each element.
<b>Return Value</b>	None		

<b>Python Syntax</b>	CopySubdesign (["NAME:Selections", "Selections:=", [<selections>]])
<b>Python Example</b>	<pre>oEditor.CopySubdesign ([     "NAME:Selections", "Selections:=", [         "CompInst@Simplorer2;23;17"]])</pre>

<b>VB Syntax</b>	CopySubdesign Array("NAME:Selections", "Selections:=", Array(<selections>))
<b>VB Example</b>	<pre> oEditor.CopySubdesign Array(     "NAME:Selections", "Selections:=",     Array ("CompInst@Simplorer2;23;17")) </pre>

## CreatePage (Schematic Editor)

Create a page at the end of the existing pages

<b>UI Access</b>	Schematic>New Page									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;PageName&gt;</td> <td>String</td> <td>Name of the page that has been created</td> </tr> <tr> <td>&lt;PageNum&gt;</td> <td>Integer</td> <td>Page Number</td> </tr> </tbody> </table>	Name	Type	Description	<PageName>	String	Name of the page that has been created	<PageNum>	Integer	Page Number
Name	Type	Description								
<PageName>	String	Name of the page that has been created								
<PageNum>	Integer	Page Number								
<b>Return Value</b>	None									

### Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	CreatePage(string name, // Page name [out,retval] int num) // Page number
<b>Python Example</b>	<pre> oEditor.CreatePage ("TestPage1") </pre>

<b>VB Syntax</b>	CreatePage( string name, // Page name [out,retval] int num) // Page number
<b>VB Example</b>	oEditor.CreatePage "TestPage1"

## Cut (Schematic Editor)

Cut Page Selection

<b>UI Access</b>	Edit>Cut		
<b>Parameters</b>	Name <Selections>	Type Array	Description Array ("NAME:Selections", _ "Page:=", page number, _// [Opt=1] Page number "Selections:=", IDs to modify))
<b>Return Value</b>	None		

### Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	Cut()
<b>Python Example</b>	oEditor.Cut ( ["NAME:Selections", "Selections:=", ["CompInst@R;10;93"] ] )

<b>VB Syntax</b>	Cut()
------------------	-------

**VB Example**

```
oEditor.Cut Array("NAME:Selections",
"Selections:=", Array("CompInst@R;10;93"))
```

## DeactivateOpen (Schematic Editor)

Disable items and replace with an open circuit

UI Access	Edit>Deactivate (Open)		
Parameters	Name <Selections>	Type Array	Description Array("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify))
Return Value	None		

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

Python Syntax	DeactivateOpen()
Python Example	<pre>oEditor.DeactivateOpen ([ "NAME:Selections", "Selections:=", [ "CompInst@R;2;37"]])</pre>

<b>VB Syntax</b>	DeactivateOpen()
<b>VB Example</b>	<pre>oEditor.DeactivateOpen Array("NAME:Selections", "Selections:=", Array( "CompInst@R;2;37" ))</pre>

## DeactivateShort (Schematic Editor)

Disable items and replace with a closed circuit

<b>UI Access</b>	Edit>Deactivate (Short)		
<b>Parameters</b>	Name	Type	Description
	<Selections>	Array	<p>Array ("NAME:Selections", _</p> <p>"Page:=", page number, _ // [Opt=1] Page number</p> <p>"Selections:=", IDs to modify))</p>
<b>Return Value</b>	None		

### Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	DeactivateShort ()
<b>Python Example</b>	<pre>oEditor.DeactivateShort(["NAME:Selections", "Selections:=", [ "CompInst@R;2;37"]])</pre>

<b>VB Syntax</b>	DeactivateShort()
<b>VB Example</b>	<pre>oEditor.DeactivateShort Array("NAME:Selections", "Selections:=", Array( _     "CompInst@R;2;37"))</pre>

## Delete (Schematic Editor)

Delete items

<b>UI Access</b>	Edit>Delete		
<b>Parameters</b>	Name <Selections>	Type Array	Description Array ("NAME:Selections", _ "Page:=", page number, _// [Opt=1] Page number "Selections:=", IDs to modify))
<b>Return Value</b>	None		

### Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	Delete()
<b>Python Example</b>	<pre>oEditor.Delete ([ "NAME:Selections", "Selections:=",     ["CompInst@R;10;93"] ] )</pre>

<b>VB Syntax</b>	Delete()
<b>VB Example</b>	<pre>oEditor.Delete Array("NAME:Selections", "Selections:=", Array("CompInst@R;10;93"))</pre>

## DeletePage (Schematic Editor)

To reduce the number of pages in the schematic

<b>UI Access</b>	Schematic>Remove Page		
<b>Parameters</b>	Name	Type	Description
<b>Return Value</b>	None		

### Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	DeletePage(<pagenum>, <pagenum>*)
<b>Python Example</b>	<pre>num = oEditor.DeletePage([1, 4])</pre>

<b>VB Syntax</b>	DeletePage Array(<pagenum>[, <pagenum>]*)
<b>VB Example</b>	<pre>Dim oEditor Set num = oEditor.DeletePage Array(1, 4)</pre>

## ElectricRuleCheck (Schematic Editor)

Check Electric Rule

UI Access	Schematic>Electric Rule Check		
Parameters	Name <ElectricRules>	Type Array	Description Array(string, ...), bool  Array  Array of strings containing names of electric rules to perform.  Valid names are:  "PinRule": Checks for components with unconnected pins  "OutputPinRule": Checks for nets with more than one output pin  bool  True if this should check all subcircuits  False if this should just check this circuit
Return Value	None		

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	ElectricRuleCheck()
<b>Python Example</b>	<pre>oEditor.ElectricRuleCheck (["PinRule",     "OutputPinRule", "OverlapCompRule"], True)</pre>

<b>VB Syntax</b>	ElectricRuleCheck()
<b>VB Example</b>	<pre>oEditor.ElectricRuleCheck Array("PinRule",     "OutputPinRule", "OverlapCompRule"), True</pre>

## ExportImage (Schematic Editor)

To export a picture for a specified page of the current design to a file. The image size can also be specified. The filename extension determines the type of image exported.

<b>UI Access</b>	None.		
<b>Parameters</b>	Name	Type	Description
	<filename>	String	The name of the file, with format-specific extension. Extensions supported are: bmp, gif, jpg, jpeg, png, tif, tiff.
	<pagenum>	Integer	The page number of the current schematic. Page numbers start at 1. If the number is out of range, the current schematic page will be printed.
	<dx>	Integer	The width of the image. If <code>dx</code> is less than 160, 160 will be used for the width.
	<dy>	Integer	The height of the image. If <code>dy</code> is less than 160, 160 will be used for the height.
<b>Return Value</b>	None		

### Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	ExportImage (<filename>, <pagenum>, <dx>, <dy>)
<b>Python Example</b>	<code>oEditor.ExportImage ("c:\mysch.png", 1, 800, 400)</code>

<b>VB Syntax</b>	ExportImage (<filename>, <pagenum>, <dx>, <dy>)
<b>VB Example</b>	<code>oEditor.ExportImage "c:\mysch.png", 1, 800, 400</code>

## ExportNetlist

Exports a netlist solution.

<b>UI Access</b>	None		
<b>Parameters</b>	Name	Type	Description
	<solution>	String	The Circuit solution in the design. For Maxwell Circuit, the solution must be an empty string, "".
<b>Return Value</b>	None.		

<b>Python Syntax</b>	ExportNetlist(<setupName>,<fileName>)
<b>Python Example</b>	<code>oDesign.ExportNetlist("", "SolutionData.sph")</code>

<b>VB Syntax</b>	ExportNetlist <solution> <fileName>
<b>VB Example</b>	<code>oDesign.ExportNetlist "" SolutionData.sph</code>

## FindElements (Schematic Editor)

Select elements based on properties

UI Access	Edit>Find Elements		
<b>Parameters</b>	Name	Type	Description
	<propname>	String	The name of the property, or "*" to search through all properties
	<propvalue>	String	The value of the property, or a substring
	<criterion>	Integer	0 = Contains 1 = Does not contain 2 = Exact match
	<types>	Integer	2 - components only 4 - graphic objects only 8 - wires only 16 - ports only Add the numbers to look through more types - 30 is a common number, searching through everything
	<match>	Boolean	True = match all of the properties specified False = match any of the properties specified
	<case>	Boolean	True = case-sensitive False = not case-sensitive
	<sub>	Boolean	True = search subcircuits as well as current schematic

		False = only search current schematic True = search within current selections False = search schematic(s)
<b>Return Value</b>	Array of strings, the names of the elements selected	

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	FindElements(["NAME:SearchProps", "Prop:=", [<propname>, <propvalue>, <criterion>][, [<propname>, <propvalue>, <criterion>]]*], ["NAME:Parameters", "Filter:=", <types>, "MatchAll:=", <match>, "MatchCase:=", <case>, "SearchSubCkt:=", <sub>, "SearchSelectionOnly:=", <only>])
<b>Python Example</b>	<pre> oEditor.FindElements (     [         "NAME:SearchProps",         "Prop:=", ["CompName", "", 0]     ],     [         "NAME:Parameters",         "Filter:=" , 30,         "MatchAll:=" , False,     ] ) </pre>

```

        "MatchCase:=" , False,
        "SearchSubCkt:=" , True,
        "SearchSelectionOnly:=" , False
    ]
)

```

<b>VB Syntax</b>	FindElements(Array("NAME:SearchProps", "Prop:=", Array(<propname>, <propvalue>, <criterion>)[, Array(<propname>, <propvalue>, <criterion>)]*), Array("NAME:Parameters", "Filter:=", <types>, "MatchAll:=", <match>, "MatchCase:=", <case>, "SearchSubCkt:=", <sub>, "SearchSelectionOnly:=", <only>)
<b>VB Example</b>	<pre> eltarray = oEditor.FindElements Array("NAME:SearchProps", "Prop:=", Array("InstanceName", "R", 0)), Array("NAME:Parameters", "Filter:=", _2, "MatchAll:=", true, "MatchCase:=", false, "SearchSubCkt:=", false, "SearchSelectionOnly:=", false) </pre>

## FitToBorder [Schematic Editor]

**Use:** To reset the schematic view to just include the page border, if any.

<b>UI Access</b>	View>Fit Border								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>None</td> <td></td> <td></td> </tr> </tbody> </table>			Name	Type	Description	None		
Name	Type	Description							
None									

<b>Return Value</b>	None
---------------------	------

<b>Python Syntax</b>	FitToBorder()
<b>Python Example</b>	<code>oEditor.FitToBorder()</code>

<b>VB Syntax</b>	FitToBorder
<b>VB Example</b>	<code>oEditor.FitToBorder</code>

## GridSetup (Schematic Editor)

Changes the settings on the schematic grid

<b>UI Access</b>	Schematic>Grid Setup		
	<b>Name</b>	<b>Type</b>	<b>Description</b>
<b>Parameters</b>	<Options>	Array	<code>Array ("NAME:Options", _</code> <code>"MajorGrid:=", string, _ // Major grid size with units</code> <code>"Divisions:=", int, _ // Number of minor grid divisions</code> <code>"MajorColor:=", int, _ // RGB Color of major grid lines</code> <code>"MinorColor:=", int, _ // RGB Color of minor grid lines</code> <code>"ShowGrid:=", bool, _ // Should the grid be shown?</code> <code>"SnapToGrid:=", bool, _ // Should objects snap to grid?</code>

		"BackgroundColor:=", int, _ // RGB Color of the background "SaveAsDefault:=", bool))
<b>Return Value</b>	None	

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	GridSetup()
<b>Python Example</b>	<pre>oEditor.GridSetup([     "NAME:Options", _      "MajorGrid:=", "10mm", _      "Divisions:=", 10, _      "MajorColor:=", 0x00ff0000, _ # Red     "MinorColor:=", 0x0000ff00, _ # Green     "ShowGrid:=", true, _      "SnapToGrid:=", true, _      "BackgroundColor:=", 0xffffffff, _ # White     "SaveAsDefault:=", false])</pre>

VB Syntax	GridSetup()
<b>VB Example</b>	<pre>oEditor.GridSetup  Array(     "NAME:Options", _     "MajorGrid:=", "10mm", _     "Divisions:=", 10, _     "MajorColor:=", 0x00ff0000, _ // Red     "MinorColor:=", 0x0000ff00, _ // Green     "ShowGrid:=", true, _     "SnapToGrid:=", true, _     "BackgroundColor:=", 0x00ffffff, _ // White     "SaveAsDefault:=", false)</pre>

For example:

```
oEditor.GridSetup Array(
    "NAME:Options", _
    "MajorGrid:=", "10mm", _
    "Divisions:=", 10, _
    "MajorColor:=", 0x00ff0000, _ // Red
    "MinorColor:=", 0x0000ff00, _ // Green
    "ShowGrid:=", true, _
```

```
"SnapToGrid:=", true, _
"BackgroundColor:=", 0x00ffffff, _ // White
"SaveAsDefault:=", false)
```

## FlipHorizontal (Schematic Editor)

Flip items about the horizontal (x) axis

<b>UI Access</b>	NA		
<b>Parameters</b>	Name	Type	Description
	<Selections>	Array	Array ("NAME: Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify))
<b>Return Value</b>	None		

### Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	FlipHorizontal(
<b>Python Example</b>	oEditor.FlipHorizontal(

```
[ "NAME:Selections", "Selections:=", [ _  
"CompInst@R;2;4", "CompInst@C;1;1" ] ],  
[ "NAME:FlipParameters",  
"Disconnect:=", true, "Rubberband:=", false])
```

<b>VB Syntax</b>	FlipHorizontal(
	oEditor.FlipHorizontal Array("NAME:Selections", "Selections:=", Array( _ "CompInst@R;2;4", "CompInst@C;1;1" )), Array("NAME:FlipParameters", "Disconnect:=", true, "Rubberband:=", false))
<b>VB Example</b>	

## FlipVertical (Schematic Editor)

Flip items about the vertical (y) axis

<b>UI Access</b>	NA		
<b>Parameters</b>	Name <Selections>	Type Array	Description Array("NAME:Selections", _ "Page:=", page number, _// [Opt=1] Page number "Selections:=", IDs to modify))  <FlipParameters> Array Array("NAME:FlipParameters", _

		"Disconnect:=", bool _ // [Opt=0] Should wires disconnect "Rubberband:=", bool) _ // [Opt=1] Should wires staircase
<b>Return Value</b>	None	

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	FlipVertical(
<b>Python Example</b>	<pre>oEditor.FlipVertical ([ "NAME:Selections", "Selections:=", [ "CompInst@R;2;4", "CompInst@C;1;1" ]], [ "NAME:FlipParameters", "Disconnect:=", true, "Rubberband:=", false])</pre>

<b>VB Syntax</b>	FlipVertical(
<b>VB Example</b>	<pre>oEditor.FlipVertical Array("NAME:Selections", "Selections:=", Array("CompInst@R;2;4", "CompInst@C;1;1")), Array("NAME:FlipParameters", "Disconnect:=", true, "Rubberband:=", false)</pre>

## Move (Schematic Editor)

Move items

<b>UI Access</b>	NA		
<b>Parameters</b>	Name	Type	Description
	<Selections>	Array	Array("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify))
<b>Return Value</b>	None		

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	Move()
<b>Python Example</b>	<pre>oEditor.Move ( ["NAME:elements", "circle_0", "rect_2"],</pre>

	[0.0165613577023499, -0.001])
--	-------------------------------

<b>VB Syntax</b>	Move()
<b>VB Example</b>	<pre>oEditor.Move Array("NAME:elements", "circle_0", "rect_2"), Array(0.0165613577023499, -0.001)</pre>

## NameNets (Schematic Editor)

Change names of nets on the schematic

<b>UI Access</b>	Schematic>Auto-Name Wires								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>None</td> <td></td> <td></td> </tr> </tbody> </table>			Name	Type	Description	None		
Name	Type	Description							
None									
<b>Return Value</b>	None								

### Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	<pre>NameNets(     bool display, // Show Net names or not     ["NAME:NetNames", _      "string:=", _ #Old name of the net (e.g. net_1)      string, _ #New name of the net (e.g. my_net)      ...]) # Repeat for additional nets to change</pre>
<b>Python Example</b>	<pre>oEditor.NameNets (True, ["NAME:NetNames",     "Wire@net_2;41:1:=", "A[0]"])</pre>

<b>VB Syntax</b>	<pre>NameNets(     bool display, // Show Net names or not     Array("NAME:NetNames", _     "string:=", _ // Old name of the net (e.g. net_1)     string, _ // New name of the net (e.g. my_net)     ...)) // Repeat for additional nets to change</pre>
<b>VB Example</b>	<pre>oEditor.NameNets true, Array ("NAME:NetNames",     "Wire@net_2;41:1:=", "A[0]")</pre>

## PageBorders (Schematic Editor)

### Note:

This command has been replaced with the [SetPageData](#) command. Legacy scripts using this command will play back for the first page of a schematic only.

Sets up visible page borders on the schematic

```
PageBorders(  
    Array("NAME:Options", _  
        "PageSize:=", _  
        Array("x:=", string, _ // Width of page border with units  
              "y:=", string), _ // Height of page border with units  
        "PageMargins:=", _  
        Array("x:=", string, _ // Margin Width with units  
              "y:=", string), _ // Margin Height with units  
        "ZonesHoriz:=", int, _ // Number of Horizontal zones  
        "ZonesVert:=", int)) // Number of Vertical zones
```

### Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

## Pan (Schematic Editor)

Pan Display

<b>UI Access</b>	View>Pan		
<b>Parameters</b>	Name	Type	Description
<b>Return Value</b>	None		

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	Pan([ double, _ // Move the visible area by X meters double]) // Move the visible area by Y meters
<b>Python Example</b>	<code>oDefinitionEditor.Pan ([-0.00922653869663931, 0.00686839904222147])</code>

<b>VB Syntax</b>	Pan(Array( double, _ // Move the visible area by X meters double)) // Move the visible area by Y meters
<b>VB Example</b>	<code>oDefinitionEditor.Pan</code>

	Array (-0.00922653869663931, 0.00686839904222147)
--	---------------------------------------------------

## Paste (Schematic Editor)

Paste copied items

<b>UI Access</b>	Edit>Paste		
<b>Parameters</b>	Name	Type	Description
	None		
<b>Return Value</b>	None		

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	Paste( VARIANT attrs)# specified attributes
<b>Python Example</b>	<pre>oEditor.Paste ([ "NAME:Attributes",     "Page:=", 1, "X:=", -0.059131200000001, "Y:=",_     0.0307847999999999, "Angle:=", 0, "Flip:=",false])</pre>

<b>VB Syntax</b>	Paste(VARIANT attrs); // specified attributes
<b>VB Example</b>	<pre>oEditor.Paste Array ("NAME:Attributes",     "Page:=", 1, "X:=", -0.059131200000001, "Y:=",_</pre>

	0.030784799999999, "Angle:=", 0, "Flip:=", false)
--	---------------------------------------------------

## PasteData [Schematic Editor]

**Use:** Paste graphic object or component properties to another object or component.

**Command:** Edit> Paste Data (or Right-click on graphic object or component and select Paste Data)

**Syntax:** PasteData Array("NAME:Selections", "Selections:=", Array(<selections>), "SelectList:=", Array(<select\_prop\_display>, <select\_parameters>), "Exclude:=", Array(<ExcludeList>))

**Return Value:** None.

**Parameters:** Selections

Type: Comma separated component instance list where user wants to paste data

Identifiers for each selected item to be copied, in the form CompInst@<id>, where id is the compname;ID;schematicID for each element.

SelectList

Type: Comma separated Boolean list where:

<select\_prop\_display> true to paste property displays

<select\_parameters> true to paste parameters

<ExcludeList>

Type: Comma separated list of property names to exclude from being pasted.

**Example:** Set oEditor = oDesign.SetActiveEditor("SchematicEditor")

```
oEditor.PasteData Array("NAME:Selections", "Selections:=", Array("CompInst@Cap;2;3"),  
"SelectList:=", Array(true, true), "Exclude:=", Array("IC"))
```

## PasteDesign (Schematic Editor)

Paste a design that has already been copied to the clipboard into schematic as a subdesign.

*Syntax:*

```
PasteDesign(  
    pasteOption, // see below for values  
    Array("NAME:Attributes", _  
        "Page:=", int, _ // [Opt=1] Page number (one-based)  
        // Note: Page 1 always exists  
        "X:=", double, _ // X position of the object  
        "Y:=", double, _ // Y position of the object  
        "Angle:=", double, _ // Rotation angle (radians)  
        "Flip:=", bool) // True if mirrored  
)
```

*Return Value:* Component is created and the returned value has the format ComInst@<CompName>;<compInstID>;<optional schematicID>

For instance a return value ComInst@CAP\_4;35 indicates that the component name is CAP\_, its ID is 5, and its Schematic ID is 35. These details can be found in the Properties window when you select the created component. The component name and ID can be found in the General tab and the Schematic ID can be found in the Symbols tab.

*Parameters:* pasteOption should be one of the following:

- 0 to link to the existing design that was copied
- 1 to create a new copy of the design that was copied but keep the original layers of the design being copied
- 2 to create a new copy of the design and merge the layers of the design being copied into the design receiving the copy

*VB Example:*

```
Dim oAnsoftApp  
Dim oDesktop  
Dim oProject  
Dim oDesign  
Dim oEditor  
Dim oModule  
  
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")  
Set oDesktop = oAnsoftApp.GetAppDesktop()  
oDesktop.RestoreWindow  
  
Set oProject = oDesktop.SetActiveProject("SimpleCircuitPaste")  
oProject.CopyDesign "D"  
  
Set oDesign = oProject.SetActiveDesign("A")  
Set oEditor = oDesign.SetActiveEditor("SchematicEditor")  
oEditor.PasteDesign 1, Array("NAME:Attributes", "Page:=", 1, "X:=", 0.0762, "Y:=", _  
0.0635, "Angle:=", 0, "Flip:=", false)
```

## PushExcitations

Allows access to computed excitations for transient and linear frequency solutions. The script command can be accessed from three locations.

UI Access	<ul style="list-style-type: none"> <li>• Layout Editor</li> <li>• Schematic Editor</li> <li>• Select a Nexxim solution in a 3D Layout design, right click, and choose <b>Push Excitations</b>.</li> </ul>									
Parameters	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="445 572 836 719">&lt;Reference Designation&gt;</td> <td data-bbox="836 572 931 719">String</td> <td data-bbox="931 572 1896 719">           "&lt;refdes&gt;"            This argument is empty when excitations are pushed from a Nexxim solution.         </td> </tr> <tr> <td data-bbox="445 719 836 1356">&lt;PushExcitations Parameters&gt;</td> <td data-bbox="836 719 931 1356">Array</td> <td data-bbox="931 719 1896 1356">           This parameter changes depending on whether the excitations comes from a transient or linear frequency solution. The keyword "transient:=" indicates to AEDT which solution generated the excitations. If "transient:=" is present, "CalcThevenin =" and its value are ignored.            For a linear frequency solution, use this array:  <pre>Array ("NAME:options",       "CalcThevenin =", &lt;true false&gt;,       "Sol:=", "&lt;solution name&gt;")</pre>           If CalcThevenin is true, Thevenin's equivalent is calculated. Parameters for the linear frequency solution do not include a Freq argument, so all frequencies from the solution are used.            For a transient solution, use:  <pre>Array ("NAME:options",</pre> </td> </tr> </tbody> </table>	Name	Type	Description	<Reference Designation>	String	"<refdes>" This argument is empty when excitations are pushed from a Nexxim solution.	<PushExcitations Parameters>	Array	This parameter changes depending on whether the excitations comes from a transient or linear frequency solution. The keyword "transient:=" indicates to AEDT which solution generated the excitations. If "transient:=" is present, "CalcThevenin =" and its value are ignored. For a linear frequency solution, use this array: <pre>Array ("NAME:options",       "CalcThevenin =", &lt;true false&gt;,       "Sol:=", "&lt;solution name&gt;")</pre> If CalcThevenin is true, Thevenin's equivalent is calculated. Parameters for the linear frequency solution do not include a Freq argument, so all frequencies from the solution are used. For a transient solution, use: <pre>Array ("NAME:options",</pre>
Name	Type	Description								
<Reference Designation>	String	"<refdes>" This argument is empty when excitations are pushed from a Nexxim solution.								
<PushExcitations Parameters>	Array	This parameter changes depending on whether the excitations comes from a transient or linear frequency solution. The keyword "transient:=" indicates to AEDT which solution generated the excitations. If "transient:=" is present, "CalcThevenin =" and its value are ignored. For a linear frequency solution, use this array: <pre>Array ("NAME:options",       "CalcThevenin =", &lt;true false&gt;,       "Sol:=", "&lt;solution name&gt;")</pre> If CalcThevenin is true, Thevenin's equivalent is calculated. Parameters for the linear frequency solution do not include a Freq argument, so all frequencies from the solution are used. For a transient solution, use: <pre>Array ("NAME:options",</pre>								

			<pre>"transient:=", Array(     "start:=", &lt;start time&gt;,     "stop:=", &lt;stop time&gt;,     "maxHarmonics:=", &lt;max harmonics&gt;,     "winType:=", &lt;&gt;window&gt;,     ["widthPct:=", &lt;width percentage&gt;,]     ["kaiser:=", &lt;Kaiser value&gt;,]     ["correctCoherentGain:=", true]),     "Sol:=", "&lt;solution name&gt;") winType can have the following values: <ul style="list-style-type: none"> <li>• Rectangular</li> <li>• Bartlett</li> <li>• Blackman</li> <li>• Hamming</li> <li>• Hanning</li> <li>• Kaiser</li> <li>• Welch</li> <li>• Weber</li> <li>• Lanzcos</li> </ul></pre>
<b>Return Value</b>	None		

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>VB Syntax</b>	<pre>PushExcitations "&lt;refdes&gt;", Array("NAME:options", "transient:=", ["CalcThevenin =", &lt;true&gt;], Array("start:=", &lt;start time&gt;, "stop:=", &lt;stop time&gt;, "maxHarmonics:=", &lt;max harmonics&gt;, "winType:=", &lt;&gt;window&gt;, ["widthPct:=", &lt;width percentage&gt;], ["kaiser:=", &lt;Kaier value&gt;], ["correctCoherentGain:=", true]), "Sol:=", "&lt;solution name&gt;")</pre>
<b>VB Example</b>	<p>For a transient solution:</p> <pre>Set oEditor = oDesign.SetActiveEditor("Layout") oEditor.PushExcitations "U3", Array("NAME:options", _ "transient:=", Array("start:=", 0, "stop:=", 5E-005, _ "maxHarmonics:=", 100, "winType:=", "Rectangular", _ "widthPct:=", 100, "kaiser:=", 0, "correctCoherentGain:=",_ true), "Sol:=", "Transient")</pre> <pre>Set oDesign = oProject.SetActiveEditor("Design1") oDesign.PushExcitations "", Array("NAME:options", _ "transient:=", Array("start:=", 0, "stop:=", 1E-08, _ "maxHarmonics:=", 100, "winType:=", "Hamming", _ "widthPct:=", 100, "kaiser:=", 0, "correctCoherentGain:=",_</pre>

```
true), "Sol:=", "Transient Setup 1")
```

For a linear frequency solution:

```
Set oEditor = oDesign.SetActiveEditor("SchematicEditor")
oEditor.PushExcitations "S1", Array("NAME:options", _
"CalcThevenin:=", false, "Sol:=", "LinearFrequency")
```

## Rotate (Schematic Editor)

Rotate items

UI Access	Draw>Rotate											
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;Selections&gt;</td> <td>Array</td> <td>           Array("NAME:Selections", _            "Page:=", page number, _ // [Opt=1] Page number            "Selections:=", IDs to modify)         </td> </tr> <tr> <td>&lt;RotateParameters&gt;</td> <td>Array</td> <td>           Array("NAME:RotateParameters", _            "Disconnect:=", bool _ // [Opt=0] Should wires disconnect            "Rubberband:=", bool _ // [Opt=1] Should wires staircase            "Angle:=", double) // [Opt=2] Angle to rotate            Note: Rotation occurs around center of ids         </td> </tr> </tbody> </table>	Name	Type	Description	<Selections>	Array	Array("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify)	<RotateParameters>	Array	Array("NAME:RotateParameters", _ "Disconnect:=", bool _ // [Opt=0] Should wires disconnect "Rubberband:=", bool _ // [Opt=1] Should wires staircase "Angle:=", double) // [Opt=2] Angle to rotate Note: Rotation occurs around center of ids		
Name	Type	Description										
<Selections>	Array	Array("NAME:Selections", _ "Page:=", page number, _ // [Opt=1] Page number "Selections:=", IDs to modify)										
<RotateParameters>	Array	Array("NAME:RotateParameters", _ "Disconnect:=", bool _ // [Opt=0] Should wires disconnect "Rubberband:=", bool _ // [Opt=1] Should wires staircase "Angle:=", double) // [Opt=2] Angle to rotate Note: Rotation occurs around center of ids										
Return Value	None											

**Note:**

When you record a script, the schematic editor's Rotate command shows the rotation angle for some older scripts in radians, while newer scripts show rotation angle in degrees.

```
oEditor.Rotate Array("NAME:Selections", "Selections:=", _  
    Array("ComplInst@RES_1;1:1")), Array("NAME:RotateParameters", _  
    "Angle:=", 1.5707963267949, "Disconnect:=", false, "Rubberband:=", false)
```

If "Degrees=xxx" is not shown, the rotation must be scaled.

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	Rotate()
<b>Python Example</b>	<pre>oEditor.Rotate ([ "NAME:Selections", "Selections:=",     ["CompInst@R;2;4", "CompInst@C;1;1"]],     [ "NAME:RotateParameters", "Degrees:=", 90, "Disconnect:=", _         false, "Rubberband:=", false])</pre>

<b>VB Syntax</b>	Rotate()
<b>VB Example</b>	<pre>oEditor.Rotate Array("NAME:Selections", "Selections:=", Array("CompInst@R;2;4", "CompInst@C;1;1")), Array("NAME:RotateParameters", "Degrees:=", 90, "Disconnect:=", false, "Rubberband:=", false)</pre>

## SelectAll (Schematic Editor)

Select all elements on the given page.

<b>UI Access</b>	Edit>Select All						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>None</td><td></td><td></td></tr></tbody></table>	Name	Type	Description	None		
Name	Type	Description					
None							
<b>Return Value</b>	None						

### Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	<pre>SelectAll(int pageNum) # Page number</pre> <p>The first page number is 1. If an invalid page number is passed in (i.e. 0, -1, etc), SelectAll will use the active page on the currently active view. If there is no active view, the first page, page 1, is used.</p>
<b>Python Example</b>	<pre>oEditor.SelectAll(1)</pre>

<b>VB Syntax</b>	SelectAll(int pageNum) // Page number  The first page number is 1. If an invalid page number is passed in (i.e. 0, -1, etc), SelectAll will use the active page on the currently active view. If there is no active view, the first page, page 1, is used.
<b>VB Example</b>	oEditor.SelectAll 1

## SelectPage (Schematic Editor)

Select a page in the UI

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>None</td> <td></td> <td></td> </tr> </tbody> </table>	Name	Type	Description	None		
Name	Type	Description					
None							
<b>Return Value</b>	None						

### Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	SelectPage(int page) # Page number
<b>Python Example</b>	oEditor.SelectPage (1)

<b>VB Syntax</b>	SelectPage(int page) // Page number
------------------	-------------------------------------

**VB Example**

```
oEditor.SelectPage 1
```

**SetPageData [Schematic Editor]****Note:**

This command replaces the [PageBorders](#) command.

To add or remove page border, title block and page property data from the specified page. Title blocks will not be shown if the page border type is "None". Any number of title blocks may be specified, as well as any number of page properties. Some of the page properties are special and their values will be updated automatically as appropriate, such as Project, ProjectPath, Design, Number and Date.

UI Access	Schematic>Page Borders and Title Blocks...		
Parameters	Name	Type	Description
	<bordertype>	String	One of: "None", "Outline", "ANSI", "ISO", "DIN"
	<sizeX>	Double (with units)	Size of X axis
	<sizeY>	Double (with units)	Size of Y axis
	<marginX>	Double (with units)	Margin of X axis
	<marginY>	Double (with units)	Margin of Y axis
	<zonesH>	Integer	
	<zonesV>	Integer	
	<symbolName>	String	The name of the title block symbol.
	<libName>	String	The name of the library (with path from library directory root) that contains the title block symbol

	<table border="1"> <tr><td>&lt;Hsnap&gt;</td><td>String</td><td>One of: "Left", "Right", "Center", "None"</td></tr> <tr><td>&lt;Vsnap&gt;</td><td>String</td><td>One of "Top", "Bottom", "Center", "None"</td></tr> <tr><td>&lt;rotation&gt;</td><td>String</td><td>One of "0", "90", "270"</td></tr> <tr><td>&lt;propName&gt;</td><td>String</td><td>The name of a page property followed by ":="</td></tr> <tr><td>&lt;propVal&gt;</td><td>String</td><td>The value of a page property</td></tr> </table>	<Hsnap>	String	One of: "Left", "Right", "Center", "None"	<Vsnap>	String	One of "Top", "Bottom", "Center", "None"	<rotation>	String	One of "0", "90", "270"	<propName>	String	The name of a page property followed by ":="	<propVal>	String	The value of a page property
<Hsnap>	String	One of: "Left", "Right", "Center", "None"														
<Vsnap>	String	One of "Top", "Bottom", "Center", "None"														
<rotation>	String	One of "0", "90", "270"														
<propName>	String	The name of a page property followed by ":="														
<propVal>	String	The value of a page property														
<b>Return Value</b>	None															

<b>Python Syntax</b>	<pre>SetPageData (&lt;pagenum&gt;,             [ "NAME:Border", "BorderType:=", &lt;bordertype&gt;, "PageSize:=", [ "x:=", &lt;sizeX&gt;,             "y:=", &lt;sizeY&gt;],             "PageMargins:=", [ "x:=", &lt;marginX&gt;,             "y:=", &lt;marginY&gt;], "ZonesHoriz:=", &lt;zonesH&gt;,             "ZonesVert:=", &lt;zonesV&gt;],             [ "NAME:TitleBlocks"[], "TitleBlock:=",             [&lt;symbolName&gt;, &lt;libName&gt;, &lt;Hsnap&gt;, &lt;Vsnap&gt;,             &lt;rotation&gt;, -1)]*],             [ "NAME:PageProps"[], &lt;propName&gt;, &lt;propVal&gt;]*])</pre>
<b>Python Example</b>	<pre>oEditor.SetPageData (1, [ "NAME:Border",             "BorderType:=", "ANSI", "PageSize:=", [ "x:=", "11in", "y:=", "8.5in"],             "PageMargins:=", [ "x:=", "0.38in", "y:=", "0.25in"], "ZonesHoriz:=",             2, "ZonesVert:=", 2], [ "NAME:TitleBlocks", "TitleBlock:=", [ "TitleBlk1",             "TitleBlocks", "Right", "Bottom", "0", -1]], [ "NAME:PageProps",</pre>

```
"Title:=", "Design Title", "Author:=", "I. M. User", "Number:=", "1",
"Project:=", "Project8", "ProjectPath:=", "e:/projects/",
"Design:=", "TwinBuilder1", "Date:=", "9:53:11 PM Jan 27, 2016"])
```

<b>VB Syntax</b>	<pre>SetPageData (&lt;pagenum&gt;, Array("NAME:Border", "BorderType:=", &lt;bordertype&gt;, "PageSize:=", Array("x:=", &lt;sizeX&gt;, "y:=", &lt;sizeY&gt;), "PageMargins:=", Array("x:=", &lt;marginX&gt;, "y:=", &lt;marginY&gt;), "ZonesHoriz:=", &lt;zonesH&gt;, "ZonesVert:=", &lt;zonesV&gt;), Array("NAME:TitleBlocks"[, "TitleBlock:=", Array(&lt;symbolName&gt;, &lt;libName&gt;, &lt;Hsnap&gt;, &lt;Vsnap&gt;, &lt;rotation&gt;, -1) ]*), Array("NAME:PageProps"[, &lt;propName&gt;, &lt;propVal&gt;]*)</pre>
<b>VB Example</b>	<pre>Dim oEditor  oEditor.SetPageData 1, Array("NAME:Border", "BorderType:=", "ANSI", "PageSize:=", Array("x:=", "11in", "y:=", "8.5in"), "PageMargins:=", Array("x:=", "0.38in", "y:=", "0.25in"), "ZonesHoriz:=", 2, "ZonesVert:=", 2), Array("NAME:TitleBlocks", "TitleBlock:=", Array("TitleBlk1",</pre>

```
"TitleBlocks", "Right", "Bottom", "0", -1)), Array("NAME:PageProps",
"Title:=", "Design Title", "Author:=", "I. M. User", "Number:=", "1",
"Project:=", "Project8", "ProjectPath:=", "e:/projects/",
"Design:=", "TwinBuilder1", "Date:=", "9:53:11 PM Jan 27, 2016")
```

## SendToBack

Send the selected object to the back of the view

<b>UI Access</b>	Draw>Send To Back								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;Object&gt;</td> <td>String</td> <td>Object to send to the back.</td> </tr> </tbody> </table>			Name	Type	Description	<Object>	String	Object to send to the back.
Name	Type	Description							
<Object>	String	Object to send to the back.							
<b>Return Value</b>	None								

### Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#)

<b>Python Syntax</b>	SendToBack (["NAME:Selections", "Selections:=", [<Object>, <Object>, ...]])
<b>Python Example</b>	<pre>oDefinitionEditor.SendToBack(["NAME:Selections", "Selections:=", ["SchObj@10"]])</pre>

<b>VB Syntax</b>	SendToBack Array("NAME:Selections", "Selections:=", Array (<Object>, <Object>, ...))
------------------	--------------------------------------------------------------------------------------

**VB Example**

```
oDefinitionEditor.SendToBack Array("NAME:Selections",
"Selections:=", Array( "SchObj@10"))
```

## ShowVariableBlock (Schematic Editor)

*Use:* Control display of Variable Text Block in Schematic Editor.

*Command:* Schematic Ribbon > Settings > Design Variables

*Syntax:* ShowVariableBlock("True") // Display Text Block  
ShowVariableBlock("False") // Hide Text Block

*Return Value:* None.

*Parameters:* True // Display Text Block  
False // Hide Text Block

**Python Syntax**

```
ShowVariableBlock()
```

**Python Example**

```
oEditor ShowVariableBlock
[
    ("True")
]
```

## SortComponents (Schematic Editor)

Sorts all components, or a block of components, using the specified method

**UI Access**

Schematic>Sort Components

	Name	Type	Description
<b>Parameters</b>	<Type>	Array	<pre>Array(type, _ // string specifying sort type       method)) // string specifying sort method type       // string argument specifying sort type: "All Components" or "Blocks"       // An empty string corresponds to "All Components" method       // string argument specifying sort method: "By Name", "Left to Right", or "Signal       Flow"</pre>
<b>Return Value</b>	None		

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	SortComponents ()
<b>Python Example</b>	<code>oEditor.SortComponents ()</code>

<b>VB Syntax</b>	SortComponents ()
<b>VB Example</b>	<code>oEditor.SortComponents</code>

**Wire (Schematic Editor)**

*Use:* Wire specified pairs of pins in two selected component instances.

*Command:* Select two component instances in the Schematic Editor, then right click on one and choose **Wire**.

**Syntax:**

```
Wire (string firstComponent, _  
      string firstListOfPins, _ // list of pins separated by space  
      string secondComponent, _  
      string secondListOfPins ) //second list of pins separated by space
```

**Return Value:** None

*VB Example:* oEditor.Wire "CompInst@etch\_sline\_s20\_z70\_1;10;76", "A1 A2", "Com-  
pInst@CX0805MRNPO9BB220;1;1", "B1 B2"

## ZoomArea (Schematic Editor)

Zoom Area

UI Access	View>Zoom Area		
Parameters	Name <Size>	Type Array	Description <pre>Array(double, _       //Base X value of the area to zoom into double),       _ // Base Y value of the area to zoom into       Array(double, _ // X size of the area to zoom       double)) // Y Size of the area to zoom  NOTE: All values in meters</pre>
Return Value	None		

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	ZoomArea()
<b>Python Example</b>	<pre>oEditor.ZoomArea ([0.00307569800375911, 0.021101611633739], [-0.050433530091516, -0.0220260723256825])</pre>

<b>Python Syntax</b>	ZoomArea()
<b>Python Example</b>	<pre>oEditor.ZoomArea ([0.00307569800375911, 0.021101611633739], [-0.050433530091516, -0.0220260723256825])</pre>

## ZoomIn (Schematic Editor)

Zoom into the schematic

<b>UI Access</b>	View>Zoom In								
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>None</td> <td></td> <td></td> </tr> </tbody> </table>			Name	Type	Description	None		
Name	Type	Description							
None									
<b>Return Value</b>	None								

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	ZoomIn()
<b>Python Example</b>	<code>oEditor.ZoomIn()</code>

<b>VB Syntax</b>	ZoomIn()
<b>VB Example</b>	<code>oEditor.ZoomIn</code>

## ZoomOut (Schematic Editor)

Zoom out from the schematic

<b>UI Access</b>	View>Zoom Out		
<b>Parameters</b>	Name	Type	Description
	None		
<b>Return Value</b>	None		

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	ZoomOut()
<b>Python Example</b>	<code>oEditor.ZoomOut()</code>

<b>VB Syntax</b>	ZoomOut()
<b>VB Example</b>	<code>oEditor.ZoomOut</code>

## ZoomPrevious (Schematic Editor)

Restore the zoom to the previous settings

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>None</td><td></td><td></td></tr></tbody></table>	Name	Type	Description	None		
Name	Type	Description					
None							
<b>Return Value</b>	None						

### Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	ZoomPrevious()
<b>Python Example</b>	<code>oEditor.ZoomPrevious ()</code>

<b>VB Syntax</b>	ZoomPrevious()
<b>VB Example</b>	<code>oEditor.ZoomPrevious</code>

## ZoomToFit (Schematic Editor)

Set the Zoom to Fit

<b>UI Access</b>	View>Fit Drawing		
<b>Parameters</b>	Name	Type	Description
	None		
<b>Return Value</b>	None		

**Note:**

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

<b>Python Syntax</b>	ZoomToFit()
<b>Python Example</b>	<code>oEditor.ZoomToFit()</code>

<b>VB Syntax</b>	ZoomToFit()
<b>VB Example</b>	<code>oEditor.ZoomToFit</code>

## Property Method List

This section presents the property script methods that are available.

[ChangeProperty \(Schematic Editor\)](#)

[GetEvaluatedPropertyValue \(Schematic Editor\)](#)

[GetProperties \(Schematic Editor\)](#)

[GetPropertyAttribute \[Schematic Editor\]](#)

[GetPropertyValue \(Schematic Editor\)](#)

[SetPropertyValue \(Schematic Editor\)](#)

### ChangeProperty (Schematic Editor)

Change a property

UI Access	NA						
Parameters	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> </table>	Name	Type	Description	Name	Type	Description
Name	Type	Description					
Name	Type	Description					
Return Value	Value						

Python Syntax	ChangeProperty()
Python Example	<pre> oEditor.ChangeProperty([     "NAME:AllTabs",     ["NAME:ComponentTab", ["NAME:PropServers", _          "Wire@net_1;14:1"], ["NAME:ChangedProps",          ["NAME:NetName", "ExtraText:=",         _</pre>

	"net_10", "Name:=", "net_10", "SplitWires:=", false]])
--	--------------------------------------------------------

VB Syntax	ChangeProperty()
<b>VB Example</b>	<pre>oEditor.ChangeProperty Array("NAME:AllTabs",                            Array("NAME:ComponentTab", Array("NAME:PropServers",                                "Wire@net_1;14:1"), Array("NAME:ChangedProps",                                Array("NAME:NetName", "ExtraText:=",                                      "net_10", "Name:=", "net_10", "SplitWires:=", false))))</pre>

## GetEvaluatedPropertyValue (Schematic Editor)

Get value.

*Command:* None.

*Syntax:* GetEvaluatedPropertyValue<tabDescription, componentID, propName>

*Return Value:* Evaluated value of variable property in double format.

*Parameters:* tabDescription = name of property tab where property is found

componentID = id of component instance where property is found,

in the format: "ComplInst@<name of component type>;<id of complInstance>"

propName = name of the variable property

*VB Example:*

```
dim info
```

```
evalValue = oEditor.GetEvaluatedPropertyValue("PassedParameterTab", "CompInst@CAP_3", "C")
```

Notes:

1. This function is only available with the schematic editor.
2. Calling this function on non-numeric properties (e.g. Text properties) returns 0.

## GetProperties (Schematic Editor)

To get the names of properties for a specified tab and element.

<b>UI Access</b>	NA		
<b>Parameters</b>	Name	Type	Description
	<tab>	String	One of six choices, "PassedParameterTab", "ComponentTab", "Component", "BaseElementTab", "Quantities", "Signals"
<b>Return Value</b>	Array of strings, the property names		

<b>Python Syntax</b>	GetProperties(<tab>, <objectId>)
<b>Python Example</b>	<pre>comparray = oEditor.GetAllComponents()  for i in comparray:      instval = oEditor.GetPropertyValues         ("PassedParameterTab", i, "InstanceName")      proparray = oEditor.GetProperties ("Quantities", i)      for j in proparray:</pre>

```
direction = oEditor.GetPropertyAttribute  
("Quantities",i, j, "Direction")  
  
AddInfoMessage(str(j) + " = (direction) " +  
str(direction))
```

VB Syntax	GetProperties(<tab>, <objectId>)
VB Example	<pre>comparray = oEditor.GetAllComponents for each i in comparray      instval = oEditorGetPropertyValue ("PassedParameterTab",i,"InstanceName")     proparray = oEditor.GetProperties ("Quantities",i)      for each j in proparray         direction = oEditor.GetPropertyAttribute         ("Quantities",i, j, "Direction")          MsgBox j + " = (direction) " + direction         next     next</pre>

## GetPropertyAttribute [Schematic Editor]

To find the value of an attribute of a specified property.

<b>UI Access</b>	NA															
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;tab&gt;</td> <td>String</td> <td>One of six choices, "PassedParameterTab", "ComponentTab", "Component", "BaseElementTab", "Quantities", "Signals"</td> </tr> <tr> <td>&lt;objectId&gt;</td> <td>String</td> <td>Instance ID of object with the property</td> </tr> <tr> <td>&lt;propName&gt;</td> <td>String</td> <td>The name of the property</td> </tr> <tr> <td>&lt;attribute&gt;</td> <td>String</td> <td>Currently, only the "Direction" attribute is supported. Return values will be one of "In", "Out" or "InOut"</td> </tr> </table>	Name	Type	Description	<tab>	String	One of six choices, "PassedParameterTab", "ComponentTab", "Component", "BaseElementTab", "Quantities", "Signals"	<objectId>	String	Instance ID of object with the property	<propName>	String	The name of the property	<attribute>	String	Currently, only the "Direction" attribute is supported. Return values will be one of "In", "Out" or "InOut"
Name	Type	Description														
<tab>	String	One of six choices, "PassedParameterTab", "ComponentTab", "Component", "BaseElementTab", "Quantities", "Signals"														
<objectId>	String	Instance ID of object with the property														
<propName>	String	The name of the property														
<attribute>	String	Currently, only the "Direction" attribute is supported. Return values will be one of "In", "Out" or "InOut"														
<b>Return Value</b>	A string, the attribute's value.															

<b>Python Syntax</b>	GetPropertyAttribute(<tab>, <objectId>, <propName>, <attribute>)
<b>Python Example</b>	<pre> comparray = oEditor.GetAllComponents()  for i in comparray:      instval = oEditor.GetPropertyValues         ("PassedParameterTab", i, "InstanceName")      proparray = oEditor.GetProperties ("Quantities", i)      for j in proparray:          direction = oEditor.GetPropertyAttribute             ("Quantities", i, j, "Direction")          AddInfoMessage(str(j) + " = (direction) " + </pre>

	str(direction))
--	-----------------

<b>VB Syntax</b>	GetPropertyAttribute(<tab>, <objectId>, <propName>, <attribute>)
<b>VB Example</b>	<pre> comparray = oEditor.GetAllComponents  for each i in compararray      instval = oEditor.GetPropertyValues ("PassedParameterTab", i, "InstanceName")      proparray = oEditor.GetProperties ("Quantities", i)      for each j in proparray          direction = oEditor.GetPropertyAttribute("Quantities", i, j, "Direction")          MsgBox j + " = (direction) " + direction      next  next </pre>

## GetPropertyValue (Schematic Editor)

Get a property

<b>UI Access</b>	NA			
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> </table>	Name	Type	Description
Name	Type	Description		

	<tab>	String	Tab with the property
	<item>	String	Name of the object
	<propname>	String	Name of the property
	<value>	String	Value of the property
<b>Return Value</b>	None		

<b>Python Syntax</b>	GetPropertyValue(<tab>,<item>,<propname>,<value>)
<b>Python Example</b>	None

<b>VB Syntax</b>	GetPropertyValue(<tab>,<item>,<propname>,<value>)
<b>VB Example</b>	Please refer to the examples that can be found in C:\Program Files\...[Ansys Installation]\...\Examples\Twin Builder\Applications\Scripting\Ex6_MathMod

## SetPropertyValue (Schematic Editor)

Set a property.

<b>UI Access</b>	NA															
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;tab&gt;</td> <td>String</td> <td>Tab with the property</td> </tr> <tr> <td>&lt;item&gt;</td> <td>String</td> <td>Name of the objects</td> </tr> <tr> <td>&lt;propname&gt;</td> <td>String</td> <td>Name of the property</td> </tr> <tr> <td>&lt;value&gt;</td> <td>String</td> <td>The new value of the property</td> </tr> </tbody> </table>	Name	Type	Description	<tab>	String	Tab with the property	<item>	String	Name of the objects	<propname>	String	Name of the property	<value>	String	The new value of the property
Name	Type	Description														
<tab>	String	Tab with the property														
<item>	String	Name of the objects														
<propname>	String	Name of the property														
<value>	String	The new value of the property														
<b>Return Value</b>	None															

<b>Python Syntax</b>	SetPropertyValue(<params>)
<b>Python Example</b>	None

<b>VB Syntax</b>	SetPropertyValue(<params>)
<b>VB Example</b>	Please refer to the examples that can be found in C:\Program Files\...[Ansys Installation]\Examples\Twin Builder\Applications\Scripting\Ex6_MathMod

## Information Method List

This section presents the information methods that are available.

[GetComlInstanceFromRefDes \(Schematic Editor\)](#)

[GetComponentInfo \(Schematic Editor\)](#)

[GetComponentPins \(Schematic Editor\)](#)

[GetComponentPinInfo \(Schematic Editor\)](#)

[GetComponentPinLocation \[Schematic Editor\]](#)

[GetEditorName \(Schematic Editor\)](#)

[GetNetConnections \(Schematic Editor\)](#)

[GetNumPages \[Schematic Editor\]](#)

[GetPortInfo \(Schematic Editor\)](#)

[GetSelections \(Schematic Editor\)](#)

## [GetSignals \(Schematic Editor\)](#)

### **GetAllPorts**

Get all ports in a design.

<b>UI Access</b>	NA		
<b>Parameters</b>	Name	Type	Description
	None		
<b>Return Value</b>	Ports in the design		

<b>Python Syntax</b>	GetAllPorts()
<b>Python Example</b>	<code>oEditor.GetAllPorts ()</code>

<b>VB Syntax</b>	GetAllPorts
<b>VB Example</b>	<pre>Set oEditor = oDesign.SetActiveEditor("SchematicEditor") oEditor.GetAllPorts ()</pre>

## **GetComplInstanceFromRefDes (Schematic Editor)**

Informational

<b>UI Access</b>	NA		
<b>Parameters</b>	Name	Type	Description

	Component	String	Name of the Component
<b>Return Value</b>	Returns IDispatch for ComplInstance		

<b>Python Syntax</b>	GetComplInstanceFromRefDes(string) # string is refDes for the component
<b>Python Example</b>	GetCompInstanceFromRefDes ("R1")

<b>VB Syntax</b>	GetComplInstanceFromRefDes(string) // string is refDes for the component
<b>VB Example</b>	GetCompInstanceFromRefDes ("R1")

## GetComponentInfo (Schematic Editor)

Informational

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <tr> <td>Name</td> <td>Type</td> <td>Description</td> </tr> <tr> <td>&lt;complID&gt;</td> <td>String</td> <td>The Schematic component's ID, obtained from GetSelections or through FindElements</td> </tr> </table>	Name	Type	Description	<complID>	String	The Schematic component's ID, obtained from GetSelections or through FindElements
Name	Type	Description					
<complID>	String	The Schematic component's ID, obtained from GetSelections or through FindElements					
<b>Return Value</b>	Array of Strings, as follows: "Page=number"						

```

"ComponentName=string"
"GateCount=number"
"LocationX=number"
"LocationY=number"
"BBoxLLx=number"
"BBoxLLy=number"
"BBoxURx=number"
"BBoxURy=number"
"Angle=number" (in degrees)
"Flip=true or false"

```

<b>Python Syntax</b>	GetComponentInfo(<compID>)
<b>Python Example</b>	<pre> For i in selectionArray:     compArray = oEditor.GetComponentInfo(i)     For Each k in compArray:         AddInfoMessage(str(k) +                        "Got with GetComponentInfo") </pre>

<b>VB Syntax</b>	GetComponentInfo<compID>
------------------	--------------------------

**VB Example**

```
selectionArray = oEditor.GetSelections  
  
For Each i in selectionArray  
  
    compArray = oEditor.GetComponentInfo(i)  
  
    For Each k in compArray  
  
        MsgBox k, 0, "GetComponentInfo"  
  
    Next  
  
Next
```

**GetComponentPins (Schematic Editor)**

Informational

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;compID&gt;</td><td>String</td><td>The Schematic component's ID, obtained from GetSelections or through FindElements.</td></tr></tbody></table>	Name	Type	Description	<compID>	String	The Schematic component's ID, obtained from GetSelections or through FindElements.
Name	Type	Description					
<compID>	String	The Schematic component's ID, obtained from GetSelections or through FindElements.					
<b>Return Value</b>	Array of strings, which are the names of all the component's pins						

**Python Syntax**

```
GetComponentPins(<compID>)
```

**Python Example**

```
selectionArray = oEditor.GetSelections()  
  
for i in selectionArray  
  
    pinArray = oEditor.GetComponentPins(i)
```

	<pre>for j in pinArray     AddInfoMessage("Pin: " + str(j))</pre>
--	-------------------------------------------------------------------

<b>VB Syntax</b>	<code>GetComponentPins&lt;compID&gt;</code>
<b>VB Example</b>	<pre>selectionArray = oEditor.GetSelections for Each i in selectionArray     pinArray = oEditor.GetComponentPins(i)     for each j in pinArray         MsgBox j, 0, "GetComponentPins"     Next Next</pre>

### GetComponentPinInfo (Schematic Editor)

Informational - gets details about a pin on a schematic component.

<b>UI Access</b>	NA									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;compID&gt;</td> <td>String</td> <td>The Schematic component's ID</td> </tr> <tr> <td>&lt;pinName&gt;</td> <td>String</td> <td>The pin name</td> </tr> </tbody> </table>	Name	Type	Description	<compID>	String	The Schematic component's ID	<pinName>	String	The pin name
Name	Type	Description								
<compID>	String	The Schematic component's ID								
<pinName>	String	The pin name								
<b>Return Value</b>	<p>String values, as follows:</p> <p>"X=number"</p> <p>"Y=number"</p>									

	"Angle=number" (in degrees) "Flip=true or false" "WireId=netID or empty string" (netID is the identifier of the schematic net, and can be used in other informational methods)
--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Python Syntax</b>	GetComponentPinInfo( <complD>, <pinName>)
<b>Python Example</b>	<pre>info = oEditor.GetComponentPinInfo ("1", "N1")</pre>

<b>VB Syntax</b>	GetComponentPinInfo <complD>, <pinName>
<b>VB Example</b>	<pre>info = oEditor.GetComponentPinInfo ("1", "N1")</pre>

## GetComponentPinLocation [Schematic Editor]

Informational - gets the X or Y location of a pin on a schematic component. Invoke once with TRUE for "XorY" to get the X value of the pin's location in the schematic, and once with FALSE to get the Y value.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;complD&gt;</td><td>String</td><td>The Schematic component's ID</td></tr></table>	Name	Type	Description	<complD>	String	The Schematic component's ID
Name	Type	Description					
<complD>	String	The Schematic component's ID					

	<table border="1"> <tr> <td>&lt;pinName&gt;</td><td>String</td><td>The pin name</td></tr> <tr> <td>&lt;XorY&gt;</td><td>Boolean</td><td>TRUE gets the X value; FALSE gets the Y value.</td></tr> </table>	<pinName>	String	The pin name	<XorY>	Boolean	TRUE gets the X value; FALSE gets the Y value.
<pinName>	String	The pin name					
<XorY>	Boolean	TRUE gets the X value; FALSE gets the Y value.					
<b>Return Value</b>	<p>String values, as follows:</p> <p>"X=<i>number</i>"</p> <p>"Y=<i>number</i>"</p>						

<b>Python Syntax</b>	GetComponentPinLocation <complID>, <pinname>, <XorY>
<b>Python Example</b>	<pre>varx = oEditor.GetComponentPinLocation (compid, "N1", TRUE)</pre>

<b>VB Syntax</b>	GetComponentPinLocation <complID>, <pinname>, <XorY>
<b>VB Example</b>	<pre>varx = oEditor.GetComponentPinLocation (compid, "N1", TRUE)</pre>

## GetEditorName (Schematic Editor)

Get the name of the schematic editor.

<b>UI Access</b>	N/A
<b>Parameters</b>	None
<b>Return Value</b>	String containing the name of the schematic editor. "SchematicEditor"

<b>Python Syntax</b>	GetEditorName()
<b>Python Example</b>	<code>oEditor.GetEditorName ()</code>

<b>VB Syntax</b>	GetEditorName( [out, retval] string) // name of the editor
<b>VB Example</b>	<pre>dim info       info = oEditor.GetEditorName</pre>

## GetNetConnections (Schematic Editor)

*Use:* Informational.

*Command:* None.

*Syntax:* GetNetConnections(<netName>)

*Return Value:* Array of strings containing the connections for the net identified by the

netName argument. Strings in the array are in one of four formats:

"IPort portID x y pinName"

"GPort portID x y pinName"

"PagePort portID x y pinName"

"Component componentName compID x y pinName"

where *IPort*, *GPort*, *PagePort*, and *Component* are key words

that indicate what the rest of the string specifies:

*IPort* indicates that *portID* specifies the name of the interface port,

*GPort* indicates that *portID* specifies the name of the global port,

*PagePort* indicates that *portID* specifies the name of the page port,  
*Component* indicates that *compID* specifies the component instance identifier,  
*x* and *y* are the connection point,  
*pinName* is the name of the connected pin,  
*componentName* is the name of the component.

*Parameters:* <netName>

Type: String

The name of the net. Specifying "0" as the netName will return all objects connected to ground including all ground ports.

VB Example: `netArray = oEditor.GetNetConnections("net_1")`

## GetNumPages [Schematic Editor]

To determine how many pages there are in the schematic.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>None</td> <td></td> <td></td> </tr> </tbody> </table>	Name	Type	Description	None		
Name	Type	Description					
None							
<b>Return Value</b>	Integer, the number of pages in the schematic.						

<b>Python Syntax</b>	<code>GetNumPages()</code>
<b>Python Example</b>	<code>num = oEditor. GetNumPages ()</code>

--	--

<b>VB Syntax</b>	GetNumPages
<b>VB Example</b>	Set num = oEditor. GetNumPages

## GetPortInfo (Schematic Editor)

Informational – get details about a port on a schematic component.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;portID&gt;</td><td>String</td><td>The Schematic port's ID, obtained from GetSelections or through FindElements.</td></tr></tbody></table>	Name	Type	Description	<portID>	String	The Schematic port's ID, obtained from GetSelections or through FindElements.
Name	Type	Description					
<portID>	String	The Schematic port's ID, obtained from GetSelections or through FindElements.					
<b>Return Value</b>	<p>Array of strings, as follows:</p> <p>"Name=string" // Name of the port.</p> <p>"X=val" // Connection point X location.</p> <p>"Y=val" // Connection point Y location.</p> <p>"Angle=val" // Component angle.</p> <p>"Flip=true or false" // Whether the component is flipped or not.</p> <p>"WireId=string" // The identifier of the schematic net, can be used in other informational methods.</p>						

<b>Python Syntax</b>	GetPortInfo(<portID>)
<b>Python Example</b>	<pre>selectionArray = oEditor.GetSelections()  for i in selectionArray:     info = oEditor.GetPortInfo(i)     for j in info:         AddInfoMessage(str("GetPortInfo: " + j))</pre>

<b>VB Syntax</b>	GetPortInfo<portID>
<b>VB Example</b>	<pre>selectionArray = oEditor.GetSelections  For Each i in selectionArray     info = oEditor.GetPortInfo(i)     for each j in info         MsgBox j, mbokonly, "GetPortInfo"     Next</pre> <p style="text-align: right;">Next</p>

## GetSelections (Schematic Editor)

Returns an array of currently selected objects.

<b>UI Access</b>	N/A
<b>Parameters</b>	None.

<b>Return Value</b>	Array containing object IDs
---------------------	-----------------------------

<b>Python Syntax</b>	GetSelections()
<b>Python Example</b>	<code>oEditor.GetSelections()</code>

<b>VB Syntax</b>	GetSelections
<b>VB Example</b>	<code>oEditor.GetSelections</code>

## GetSignals (Schematic Editor)

Informational

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"><tr><td>Name</td><td>Type</td><td>Description</td></tr><tr><td>&lt;wirename&gt;</td><td>String</td><td>The wire's name, as seen in the UI, obtained in script through parsing return data from GetWireInfo.</td></tr></table>	Name	Type	Description	<wirename>	String	The wire's name, as seen in the UI, obtained in script through parsing return data from GetWireInfo.
Name	Type	Description					
<wirename>	String	The wire's name, as seen in the UI, obtained in script through parsing return data from GetWireInfo.					
<b>Return Value</b>	Array of strings which contains the signal names in the <i>wirename</i> argument. The <i>wirename</i> "Net1" would return an array with a single string "Net1". The <i>wirename</i> "WW[0-3],A,B" would return an array with six strings "WW[0]", "WW[1]", "WW[2]", "WW[3]", "A", :B".						

<b>Python Syntax</b>	GetSignals(<wirename>)
<b>Python Example</b>	<code>selectionArray = oEditor.GetSelections()</code>

```
for i in selectionArray:

    netArray3 = oEditor.GetWireInfo(i)

    for n in netArray3:

        wname = n[0:9]

        if (wname == "WireName:"):

            nn = len(n)

            wn = n[nn-10:]

            AddInfoMessage("Wirename, parsed from ID :" + str(wn))

            sigs = oEditor.GetSignals(wn)

            for ll in sigs:

                AddInfoMessage("Element in array

from GetSignals :" + str(ll))
```

<b>VB Syntax</b>	GetSignals(<wirename>)
<b>VB Example</b>	selectionArray = oEditor.GetSelections  For Each i in selectionArray      netArray3 = oEditor.GetWireInfo(i)

```
for each n in netArray3

    wname = Left(n, 9)

    If (wname = "WireName:") Then

        nn = Len(n)

        wn = Right(n, nn - 10)

        MsgBox wn, 0, "Wirename, parsed from ID"

        sigs = oEditor.GetSignals(wn)

        for each ll in sigs

            MsgBox ll, 0, "Element in array from GetSignals"

        Next

    End If

    Next

    Next
```

---

## GetWireConnections (Schematic Editor)

Informational.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>None</td> <td></td> <td></td> </tr> </tbody> </table>	Name	Type	Description	None		
Name	Type	Description					
None							
<b>Return Value</b>	<p>Array of strings containing the connections for the wire identified by the wireID argument.</p> <p>The strings in the array are in one of four formats:</p> <p>"InterfacePort=portname portID x,y"</p> <p>"GlobalPort=portname portID x,y"</p> <p>"PagePort=portname portID x,y"</p> <p>"ComponentPin=compld pinname x,y"</p> <p>where x,y is the connection point.</p>						

<b>Python Syntax</b>	GetWireConnections(<wireID>)
<b>Python Example</b>	<pre>selectionArray = oEditor.GetSelections()  for i in selectionArray:     AddInfoMessage("Selected element" + i)      netArray = oEditor.GetWireConnections(i)      for m in netArray:         AddInfoMessage("GetWireConnections" + str(m))</pre>

<b>VB Syntax</b>	GetWireConnections(<wireID>)
------------------	------------------------------

**VB Example**

```
selectionArray = oEditor.GetSelections  
  
For Each i in selectionArray  
  
    MsgBox i, 0, "Selected element"  
  
    netArray = oEditor.GetWireConnections(i)  
  
    for each m in netArray  
  
        MsgBox m, mbokonly, "GetWireConnections"  
  
    Next  
  
Next
```

**GetWireInfo (Schematic Editor)**

Informational.

<b>UI Access</b>	NA		
<b>Parameters</b>	Name <wire id>	Type String	Description The Schematic wire's ID, obtained from GetSelections or through FindElements.

	<p>Array of strings containing the wire info in the wireID argument.</p> <p>The strings in the array are:</p> <ul style="list-style-type: none"> <li>"Page=number"</li> <li>"WireName=string"</li> <li>"SegmentCount=number"</li> <li>"IPortCount=number"</li> <li>"GPortCount=number"</li> <li>"PagePortCount=number"</li> <li>"PinCount=number"</li> <li>"BBoxLLx=val"</li> <li>"BBoxLLy=val"</li> <li>"BBoxURx=val"</li> <li>"BBoxURy=val"</li> </ul>
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Python Syntax</b>	GetWireInfo(<wireID>)
<b>Python Example</b>	<pre>selectionArray = oEditor.GetSelections()  for i in selectionArray:     AddInfoMessage("Selected element" + str(i))      netArray2 = oEditor.GetWireInfo(i)      for n in netArray2:</pre>

	AddInfoMessage ("GetWireInfo" + str(n))
--	-----------------------------------------

<b>VB Syntax</b>	GetWireInfo(<wireID>)
<b>VB Example</b>	<pre> selectionArray = oEditor.GetSelections  For Each i in selectionArray      MsgBox i, 0, "Selected element"      netArray3 = oEditor.GetWireInfo(i)      for each n in netArray3          MsgBox n, 0, "GetWireInfo"      Next  Next </pre>

## GetWireSegments (Schematic Editor)

Informational.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;wireID&gt;</td> <td>String</td> <td>The Schematic wire's ID, obtained from GetSelections or through FindElements</td> </tr> </tbody> </table>	Name	Type	Description	<wireID>	String	The Schematic wire's ID, obtained from GetSelections or through FindElements
Name	Type	Description					
<wireID>	String	The Schematic wire's ID, obtained from GetSelections or through FindElements					

<b>Return Value</b>	<p>Array of strings containing the wire segments in the wireID argument. The strings in the array are:</p> <p>"Segment=val,val val,val number" the "val" quantities are segment endpoints x1,y1 x2,y2 and the "number" is a schematic segment ID</p>
---------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Python Syntax</b>	<pre>GetWireSegments (&lt;wireID&gt;)</pre>
<b>Python Example</b>	<pre>selectionArray = oEditor.GetSelections()  for i in selectionArray:     AddInfoMessage("Selected element" + str(i))      netArray2 = oEditor.GetWireSegments(i)      for n in netArray2:         AddInfoMessage("GetWireSegments" + str(n))</pre>

<b>VB Syntax</b>	<pre>GetWireSegments (&lt;wireID&gt;)</pre>
<b>VB Example</b>	<pre>selectionArray = oEditor.GetSelections  For Each i in selectionArray     MsgBox i, 0, "Selected element"      netArray2 = oEditor.GetWireSegments(i)      for each n in netArray2</pre>

```
MsgBox n, 0, "GetWireSegments"
```

Next

Next

# 27 - Definition Manager Script Commands

The definition manager controls the use of materials and scripts in a project. It also provides access to the managers for symbols, footprints, padstacks, and components in a project.

```
Set oProject = oDesktop.SetActiveProject("Project1")
Set oDefinitionManager = oProject.GetDefinitionManager()
```

**The topics for this section include:**

[AddDefinitionFromBlock](#)

[AddMaterial](#)

[CloneMaterial](#)

[DoesMaterialExist](#)

[EditMaterial](#)

[ExportMaterial](#)

[RemoveMaterial](#)

[RemoveUnusedDefinitions](#)

**Related Topics:**

[Component Manager Script Commands](#)

[Material Manager Script Commands](#)

[Model Manager Script Commands](#)

[Network Data Explorer Manager Script Commands](#)

[Script and Library Scripts](#)

[Symbol Manager Script Commands](#)

## AddMaterial (Maxwell)

**Use:** Adds a local material.

**Command:** Add Material command in the material editor .

**Syntax:** AddMaterial Array("NAME:<MaterialName>","<MatProperty>, <MatProperty>, ...)

**Return Value:** None

**Parameters:** <MatProperty> (simple material)

"<PropertyName>:=", <value>

<MatProperty> (anisotropic material)

Array ("NAME:<PropertyName>","

"property\_type:=", "AnisoProperty",

"unit:=", <string>", "component1:=", <value>,

```
"component2:=", <value>, "component3:=", <value>)

<PropertyName>
Type: <string>

Should be one of the following:
"permittivity", "permeability", "conductivity", "dielectric_loss_tangent", "magnetic_loss_tan-
gent", "magnetic_coercivity", "core_loss_type", "mass_density", "stacking_type", "mag-
netostriction", "inverse_magnetostriction", "youngs_modulus", "poissons_ratio"

property_type
Type: <string>

Should be one of the following:
"AnisoProperty", "nonlinear"

unit
Type: <string>

Possible values: delta_H: "Oe" saturation_mag: "Gauss", "uGauss", "Tesla", "uTesla"
other properties: ""(empty string)

<MatProperty> (nonlinear material)

Array("NAME:permeability",
"property_type:=", "nonlinear",
"BTTypeForSingleCurve:=", <string>,
"Hunit:=", <string>, "Bunit:=", <string>,
"IsTemperatureDependent:=", <bool>,
Array("NAME:BHCoordinates",
```

```
Array("NAME:Coordinate", "X:=", <value>, "Y:=", <value>, ...
BTypeForSingleCurve
Type: <string>
```

Should be one of the following: "Intrinsic", "Normal" ( defaults to Normal if nothing is specified). An error message displays in the Message Window if the string does not match either "Normal" or "Intrinsic"; advising the user of this and informing the user that this property has been set to "Normal".

Hunit

```
Type: <string>
Possible values: "Oe", "A_per_meter", "kA_per_meter"
```

Bunit

```
Type: <string>
Possible values: "tesla", "kTesla" "uGauss"
```

IsTemperatureDependent

```
Type: <bool>
possible values: True or False
<MatProperty> (vector material)
Array("NAME:<PropertyName>",
      "property_type:=", "VectorProperty", "Magnitude:=", <value>, "DirCompl:=", <value>, "DirCom-
      p2:=", <value>, "DirComp3:=", <value>))
<MatProperty> (choice material)
Array("NAME:<PropertyName>",


```

```
"property_type:=", "ChoiceProperty", "Choice:=", <string>")
```

**Example:** oProject.AddMaterial Array("NAME:Material2",  
"dielectric\_loss\_tangent:=", "44",\_  
Array("NAME:saturation\_mag",\_  
"property\_type:=", "AnisoProperty",\_  
"unit:=", "Gauss",\_ "component1:=", "11", \_  
"component2:=", "22", \_ "component3:=", "33"), \_  
"delta\_H:=", "44Oe")

**Example:** oDefinitionManager.AddMaterial Array("NAME:Material5",\_  
"CoordinateSystemType:=", \_"Cartesian",\_  
Array("NAME:AttachedData"),\_  
Array("NAME:magnetic\_coercivity", "property\_type:=", \_  
"VectorProperty", "Magnitude:=", "-1A\_per\_meter",\_  
"DirComp1:=", "1", "DirComp2:=", \_"2", "DirComp3:=", \_  
"3"))

**Example:** oDefinitionManager.AddMaterial Array("NAME:Material4",\_  
"CoordinateSystemType:=", \_"Cartesian",\_  
Array("NAME:AttachedData"),\_  
Array("NAME:stacking\_type", "property\_type:=", \_  
"ChoiceProperty", "Choice:=", "Lamination"),\_  
"stacking\_factor:=", "0",\_

```
Array("NAME:stacking_direction", "property_type:=", _  
"ChoiceProperty", "Choice:=", "") )
```

NOTE: Litz Wire properties are valid only for 2D/3D Transient and 2D/3D Eddy Current designs.

**Example (Square Litz Wire):** oDefinitionManager.AddMaterial Array("NAME:LitzMaterialSquare", \_

```
...
```

```
Array("NAME:stacking_type", "property_type:=", "ChoiceProperty", "Choice:=", "Litz Wire"),
```

```
...
```

```
Array("NAME:wire_type", "property_type:=", _
```

```
"ChoiceProperty", "Choice:=", "Square"), "strand_number:=", "16", "wire_width:=", _  
"2mm")
```

**Example (Round Litz Wire):** oDefinitionManager.AddMaterial Array("NAME:LitzMaterialRound", \_

```
...
```

```
Array("NAME:stacking_type", "property_type:=", "ChoiceProperty", "Choice:=", "Litz Wire"),
```

```
...
```

```
Array("NAME:wire_type", "property_type:=", _
```

```
"ChoiceProperty", "Choice:=", "Round"), "strand_number:=", "16", "wire_diameter:=", _  
"2mm")
```

**Example (Rectangular Litz Wire):** oDefinitionManager.AddMaterial Array("NAME:LitzMa-  
terialRectangular", \_

```
...
```

```
Array("NAME:stacking_type", "property_type:=", "ChoiceProperty", "Choice:=", "Litz Wire"),  
...  
Array("NAME:wire_type", "property_type:=", _  
"ChoiceProperty", "Choice:=", "Rectangular"), "strand_number:=", "5", "wire_width:=", _  
"5.5mm", "wire_thickness:=", "5.6",  
    Array("NAME:wire_thickness_direction", "property_type:=", _  
"ChoiceProperty", "Choice:=", "V(1)")  
    Array("NAME:wire_width_direction", "property_type:=", _  
"ChoiceProperty", "Choice:=", "V(2)")  
)
```

**Note:** Young's Modulus and Poisson's Ratio properties are valid only for 2D & 3D Magnetostatic and Transient.

***Python Example (Young's Modulus and Poisson's Ratio):***

```
oDefinitionManager.AddMaterial("copper",  
[  
...  
["NAME:youngs_modulus",  
"property_type:=", "AnisoProperty",  
"unit:=", "",
```

```
"component1:=" , "1",
"component2:=" , "1.5",
"component3:=" , "3"

] ,
[

"NAME:poissons_ratio",
"property_type:=" , "AnisoProperty",
"unit:=" , "",
"component1:=" , "4",
"component2:=" , "5",
"component3:=" , "6"

] ,
...
])
```

**VB Example (Young's Modulus and Poisson's Ratio):**

```
oDefinitionManager.AddMaterial "copper",
Array("NAME:copper",
...
Array("NAME:youngs_modulus",
"property_type:=" , "AnisoProperty",
```

```

"unit:="", "",
"component1:=", "1",
"component2:=", "2",
"component3:=", "3"),
Array("NAME:poissons_ratio",
"property_type:=", "AnisoProperty",
"unit:="", "",
"component1:=", "4",
"component2:=", "5",
"component3:=", "6"),
...
)

```

## CloneMaterial

Clones a local material.

<b>UI Access</b>	N/A									
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th><th>Type</th><th>Description</th></tr> </thead> <tbody> <tr> <td>&lt;matName&gt;</td><td>String</td><td>Name of existing material.</td></tr> <tr> <td>&lt;newName&gt;</td><td>String</td><td>Name for newly cloned material.</td></tr> </tbody> </table>	Name	Type	Description	<matName>	String	Name of existing material.	<newName>	String	Name for newly cloned material.
Name	Type	Description								
<matName>	String	Name of existing material.								
<newName>	String	Name for newly cloned material.								
<b>Return Value</b>	<p>Boolean:</p> <ul style="list-style-type: none"> <li>• 1 - Material is cloned.</li> <li>• 0 - Existing material not found or a conflict with the new material name.</li> </ul>									

### Python Syntax

CloneMaterial (<matName>, <newName>)

<b>Python Example</b>	<code>oDefinitionManager.CloneMaterial("copper1", "copper3")</code>
-----------------------	---------------------------------------------------------------------

<b>VB Syntax</b>	<code>CloneMaterial &lt;matName&gt;, &lt;newName&gt;</code>
------------------	-------------------------------------------------------------

<b>VB Example</b>	<code>oDefinitionManager.CloneMaterial "copper1", "copper3"</code>
-------------------	--------------------------------------------------------------------

## ComputeCoreLossCoefficients

Creates a current source.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	<code>&lt;CoreLossModelType&gt;</code>	string	One of: <b>Electrical Steel</b> or <b>Power Ferrite</b>
	<code>&lt;MassDensity&gt;</code>	double	Mass density of the core loss model type
<code>&lt;AttachedData&gt;</code>			Either <code>&lt;CoreLossData&gt;</code> or <code>&lt;CoreLossMultiCurveData&gt;</code>
<b>Return Value</b>	Array of 3 double values in SI unit: <ul style="list-style-type: none"><li>• For Electrical Steel, returns the values for [Kh, Kc, Ke]</li><li>• For Power Ferrite, returns the values for [Cm, X, Y]</li></ul>		

<b>Python Syntax</b>	<code>ComputeCoreLossCoefficients (&lt;CoreLossModelType&gt;, &lt;MassDensity&gt;, &lt;AttachedData&gt;)</code>
----------------------	-----------------------------------------------------------------------------------------------------------------

<b>Python Example</b>	<pre>DefinitionManager.ComputeCoreLossCoefficients("Electrical Steel", 7000, [     "NAME:CoefficientSetupData",     "property_data:=", "coreloss_data",     "coefficient_setup:=", "w_per_cubic_mete",     "Frequency:=", "60Hz",     "Thickness:=", "0.5mm",     "Conductivity:=", "0",     [ "NAME:Coordinates",       [ "NAME:DimUnits", "", ],       [         [ "NAME:Points", 0, 0, 3, 6, 4, 18         ]       ]     ]   ])</pre>
-----------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>VB Example</b>	<p>Single Curve:</p> <pre>oDefinitionManager.ComputeCoreLossCoefficients "Electrical Steel", 7670.5, Array("NAME:CoefficientSetupData", "property_data:=", _ "coreloss_data", "coefficient_setup:=", "w_per_cubic_meter", "Frequency:=", _ "60Hz", "Thickness:=", "0mm", "Conductivity:=", "0", Array("NAME:Coordinates", Array("NAME:DimUnits", _, _, _), Array("NAME:Points", 0, 0, 3, 6, 4, 12)))</pre>
-------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>VB Example</b>	<p>Multi Curves:</p> <pre>oDefinitionManager.ComputeCoreLossCoefficients "Electrical Steel", 7670.5, Array</pre>
-------------------	------------------------------------------------------------------------------------------------------------------

```
("NAME:CoreLossMultiCurveData", "property_data:=", _  
    "coreloss_multi_curve_data", "coreloss_unit:=", "w_per_cubic_meter", Array("NAME:AllCurves", Array  
        ("NAME:OneCurve", "Frequency:=", _  
            "50Hz", Array("NAME:Coordinates", Array("NAME:DimUnits", "", ""), Array("NAME:Points", 0, 0, _  
                3, 6, 4, 12))), Array("NAME:OneCurve", "Frequency:=", "75Hz", Array("NAME:Coordinates", Array  
                ("NAME:DimUnits", _, _  
                    "", ""), Array("NAME:Points", 0, 0, 3, 9, 4, 16))))
```

## DoesMaterialExist

Checks for the presence of a material in the library by name

<b>UI Access</b>	None.		
<b>Parameters</b>	Name <i>&lt;MaterialName&gt;</i>	Type <i>&lt;String&gt;</i>	Description Name of the material to search for in the material database
<b>Return Value</b>	Boolean: <ul style="list-style-type: none"><li>• <b>True</b> – specified material exists.</li><li>• <b>False</b> – specified material does not exist.</li></ul>		

<b>Python Syntax</b>	DoesMaterialExist( <i>&lt;MaterialName&gt;</i> )
<b>Python Example</b>	<code>oDefinitionManager.DoesMaterialExist("modified_epoxy")</code>

<b>VB Syntax</b>	DoesMaterialExist <MaterialName>
<b>VB Example</b>	<code>oDefinitionManager.DoesMaterialExist "modified_epoxy"</code>

## EditMaterial (Maxwell)

**Use:** Modifies an existing material.

**Command:** View/Edit Materials command in the material editor.

**Syntax:** EditMaterial <OriginalName>, Array("NAME:<NewName>",  
     <MatProperty>, <MatProperty>, ...)

**Return Value:** None

**Parameters:** <OriginalName>

Type: <string>

Name of the material before editing.

<NewName>

Type: <string>

New name for the material.

Refer to [AddMaterial](#) for detailed information on material properties.

```
Example: Set oProject = oDesktop.SetActiveProject("example_magnetostatic")
Set oDefinitionManager = oProject.GetDefinitionManager()
oDefinitionManager.EditMaterial "arm_steel", Array("NAME:arm_steel", "CoordinateSystemType:=",
"Cartesian", Array("NAME:AttachedData"), Array("NAME:ModifierData"), Array("NAME:permeability",
"property_type:=",
```

```
"nonlinear", "BTypeForSingleCurve:=", "normal", "HUnit:=", "A_per_meter", "BUnit:=", "tesla",
"IIsTemperatureDependent:=", "True", Array("NAME:BHCoordinates", Array("NAME:Coordinate", "X:=",
0, "Y:=", 0), Array("NAME:Coordinate", "X:=", 4000, "Y:=", 1.413), Array("NAME:Coordinate",
"X:=",
8010, "Y:=", 1.594), Array("NAME:Coordinate", "X:=", 16010, "Y:=", 1.751), Array("NAME:Coordin-
ate", "X:=",
24020, "Y:=", 1.839), Array("NAME:Coordinate", "X:=", 32030, "Y:=", 1.896), Array("NAME:Coordin-
ate", "X:=",
40030, "Y:=", 1.936), Array("NAME:Coordinate", "X:=", 48040, "Y:=", 1.967), Array("NAME:Coordin-
ate", "X:=",
64050, "Y:=", 2.008), Array("NAME:Coordinate", "X:=", 80070, "Y:=", 2.042), Array("NAME:Coordin-
ate", "X:=",
96080, "Y:=", 2.073), Array("NAME:Coordinate", "X:=", 112100, "Y:=", 2.101), Array("NAME:Coordin-
ate", "X:=",
128110, "Y:=", 2.127), Array("NAME:Coordinate", "X:=", 144120, "Y:=", 2.151), Array("NAME:Coordin-
ate", "X:=",
176150, "Y:=", 2.197), Array("NAME:Coordinate", "X:=", 208180, "Y:=", 2.24), Array("NAME:Coordin-
ate", "X:=",
272230, "Y:=", 2.325), Array("NAME:Coordinate", "X:=", 304260, "Y:=", 2.37), Array("NAME:Coordin-
ate", "X:=",
336290, "Y:=", 2.42), Array("NAME:Coordinate", "X:=", 396000, "Y:=", 2.5))), "conductivity:=",
"2000000", Array("NAME:magnetic_coercivity", "property_type:=", "VectorProperty", "Magnitude:=",
"0A_per_meter", "DirComp1:=", "1", "DirComp2:=", "0", "DirComp3:=", "0"), Array("NAME:core_loss_
type", "property_type:=",
```

```
"ChoiceProperty", "Choice:=", "Hysteresis Model"), "core_loss_hci:=", "100000", "core_loss_
br:=",
"88", "core_loss_hkc:=", "333", "core_loss_equiv_cut_depth:=", "0.002meter")
```

**Example:** Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")

```
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow

Set oProject = oDesktop.SetActiveProject("pmsg-1_test")
Set oDefinitionManager = oProject.GetDefinitionManager()

oDefinitionManager.EditMaterial "M19-24G_3DSF0.950", Array("NAME:M19-24G_3DSF0.950", "Coordi-
nateSystemType:",
"Cartesian", "BulkOrSurfaceType:=", 1, Array("NAME:PhysicsTypes", "set:=", Array(
"Electromagnetic")), Array("NAME:permeability", "property_type:=", "nonlinear", "BType:=",
"normal", "HUnit:=", "A_per_meter", "BUnit:=", "tesla", Array("NAME:BHCoordinates", Array
("NAME:Coordinate", "X:=",
0, "Y:=", 0), Array("NAME:Coordinate", "X:=", 22.28, "Y:=", 0.05), Array("NAME:Coordinate",
"X:=",
25.46, "Y:=", 0.1), Array("NAME:Coordinate", "X:=", 31.83, "Y:=", 0.15), Array("NAME:Coordi-
nate", "X:=",
47.74, "Y:=", 0.36), Array("NAME:Coordinate", "X:=", 63.66, "Y:=", 0.54), Array("NAME:Coordi-
nate", "X:=",
79.57, "Y:=", 0.65), Array("NAME:Coordinate", "X:=", 159.15, "Y:=", 0.99), Array("NAME:Coordi-
nate", "X:=",
318.3, "Y:=", 1.2), Array("NAME:Coordinate", "X:=", 477.46, "Y:=", 1.28), Array("NAME:Coordi-
nate", "X:=",
```

```
636.61, "Y:=", 1.33), Array("NAME:Coordinate", "X:=", 795.77, "Y:=", 1.36), Array("NAME:Coordinate", "X:=",
1591.5, "Y:=", 1.44), Array("NAME:Coordinate", "X:=", 3183, "Y:=", 1.52), Array("NAME:Coordinate", "X:=",
4774.6, "Y:=", 1.58), Array("NAME:Coordinate", "X:=", 6366.1, "Y:=", 1.63), Array("NAME:Coordinate", "X:=",
7957.7, "Y:=", 1.67), Array("NAME:Coordinate", "X:=", 15915, "Y:=", 1.8), Array("NAME:Coordinate", "X:=",
31830, "Y:=", 1.9), Array("NAME:Coordinate", "X:=", 111407, "Y:=", 2), Array("NAME:Coordinate",
"X:=",
190984, "Y:=", 2.1), Array("NAME:Coordinate", "X:=", 350138, "Y:=", 2.3), Array("NAME:Coordinate", "X:=",
509292, "Y:=", 2.5), Array("NAME:Coordinate", "X:=", 2100830, "Y:=", 4.5))), "conductivity:=",
"0", Array("NAME:core_loss_type", "property_type:=", "ChoiceProperty", "Choice:=",
"Electrical Steel"), "core_loss_kh:=", "0", "core_loss_kc:=", "4.32556", "core_loss_ke:=",
"0", "mass_density:=", "7650", Array("NAME:stacking_type", "property_type:=",
"ChoiceProperty", "Choice:=", "Lamination"), "stacking_factor:=", "0.95", Array("NAME:stacking_
direction", "property_type:=",
"ChoiceProperty", "Choice:=", "V(3)"), "core_loss_equiv_cut_depth:=",
"0.002meter")
```

**Example (Nonlinear Conductivity):**

```
oDefinitionManager>EditMaterial("nonlinear_copper",
[
```

```
...
[
    "NAME:conductivity",
    "property_type:=" , "nonlinear",
    [
        "NAME:JECoordinates",
        [
            "NAME:DimUnits",
            "V_per_meter",
            "A_per_m2"
        ],
        [
            "NAME:Point",
            0,
            0
        ],
        [
            "NAME:Point",
            1,
            2
        ],

```

```
[  
"NAME:Point",  
3,  
4  
]  
]  
]  
])
```

***Example (Litz wire):***

**Note:** Litz Wire properties are valid only for 2D/3D Transient and 2D/3D Eddy Current designs.

```
oDefinitionManager>EditMaterial Array("NAME:LitzMaterialRound",_  
...  
Array("NAME:wire_type", "property_type:=", _  
"ChoiceProperty", "Choice:=", "Round"), "strand_number:=", "16", "wire_diameter:=", _  
"2mm")
```

***Example: (Thermal Modifier***

**Note:** Example using a Thermal Modifier for Electrical Steel Core Loss properties (applicable to both Electrical Steel and Power Ferrite core loss models in 3D Eddy Current and 3D Transient designs)

```
Set oProject = oDesktop.SetActiveProject("3D_eddy_temperature_dependent_coreloss")
Set oDefinitionManager = oProject.GetDefinitionManager()
oDefinitionManager.EditMaterial "copper", Array("NAME:copper", "CoordinateSystemType:=",
"Cartesian", "BulkOrSurfaceType:=", 1, Array("NAME:PhysicsTypes", "set:=", Array(
"Electromagnetic", "Thermal", "Structural")), Array("NAME:AttachedData", Array("NAME:MatAp-
pearanceData", "property_data:=",
"appearance_data")),
Array("NAME:ModifierData", Array("NAME:ThermalModifierData", "modifier_data:=",
"thermal_modifier_data", Array("NAME:all_thermal_modifiers", Array("NAME:one_thermal_mod-
ifier", "Property:=",
"core_loss_kh", "Index:::", 0, "prop_modifier:=", "thermal_modifier", "use_free_form:=",
true, "free_form_value:=", "if(Temp > 1000cel, 1, if(Temp < -273.15cel, 1, 1))), Array
("NAME:one_thermal_modifier", "Property:=",
"core_loss_kc", "Index:::", 0, "prop_modifier:=", "thermal_modifier", "use_free_form:=",
true, "free_form_value:=", "if(Temp > 1000cel, 1, if(Temp < -273.15cel, 1, 1)), Array
("NAME:one_thermal_modifier", "Property:=",
"core_loss_ke", "Index:::", 0, "prop_modifier:=", "thermal_modifier", "use_free_form:=",
true, "free_form_value:=", "if(Temp > 1000cel, 1, if(Temp < -273.15cel, 1, 1)), Array
("NAME:one_thermal_modifier", "Property:=",
"core_loss_kdc", "Index:::", 0, "prop_modifier:=", "thermal_modifier", "use_free_form:=",
```

```
true, "free_form_value:=", "if(Temp > 1000cel, 1, if(Temp < -273.15cel, 1, 1))), Array  
("NAME:one_thermal_modifier", "Property::=",  
"core_loss_equiv_cut_depth", "Index::=", 0, "prop_modifier:=",  
"thermal_modifier", "use_free_form:=", true, "free_form_value:=",  
"if(Temp > 1000cel, 1, if(Temp < -273.15cel, 1, 1))"), "permeability:=",  
"0.999991", "conductivity:=", "27840000", "thermal_conductivity:=", "400", Array("NAME:core_  
loss_type", "property_type:=",  
"ChoiceProperty", "Choice:=", "Electrical Steel"), "core_loss_kh:=", "0", "core_loss_kc:=",  
"0", "core_loss_ke:=", "0", "core_loss_kdc:=", "0", "mass_density:=", "8933", "specific_  
heat:=",  
"385", "youngs_modulus:=", "120000000000", "poissons_ratio:=", "0.38", "thermal_expansion_coeffi-  
cient:=",  
"1.77e-05", "core_loss_equiv_cut_depth:=", "0.001meter")
```

**Note:** Young's Modulus and Poisson's Ratio properties are valid only for 2D & 3D Magnetostatic and Transient.

**Python Example (Young's Modulus and Poisson's Ratio):**

```
oDefinitionManager>EditMaterial("copper",  
[  
...  
["NAME:youngs_modulus",
```

```
"property_type:=" , "AnisoProperty",
"unit:=" ,
"component1:=" , "1",
"component2:=" , "1.5",
"component3:=" , "3"

] ,
[

"NAME:poissons_ratio",
"property_type:=" , "AnisoProperty",
"unit:=" ,
"component1:=" , "4",
"component2:=" , "5",
"component3:=" , "6"

] ,
...
])


```

**VB Example (Young's Modulus and Poisson's Ratio):**

```
oDefinitionManager>EditMaterial "copper" ,
Array("NAME:copper",
...

```

```
Array("NAME:youngs_modulus",
"property_type:=", "AnisoProperty",
"unit:=", "",
"component1:=", "1",
"component2:=", "2",
"component3:=", "3"),

Array("NAME:poissons_ratio",
"property_type:=", "AnisoProperty",
"unit:=", "",
"component1:=", "4",
"component2:=", "5",
"component3:=", "6"),

...)
```

**Example: B-P Curve Core Loss Model**

**Note:** B-P Curve properties for Core Loss model are valid only for 2D/3D Transient and 2D/3D Eddy Current designs.

```
Set oProject = oDesktop.SetActiveProject("ThermalCoreLoss_2D_Eddy")

Set oDefinitionManager = oProject.GetDefinitionManager()

oDefinitionManager.EditMaterial "steel_1008_CL",
Array("NAME:steel_1008_CL", "CoordinateSystemType:=",
"Cartesian", "BulkOrSurfaceType:=", 1, Array("NAME:PhysicsTypes", "set:=", Array( "Elec-
tromagnetic", "Thermal", "Structural")), Array("NAME:AttachedData", Array("NAME:MatAp-
pearanceData", "property_data:=", "appearance_data", "Red:=", 161, "Green:=", 161, "Blue:=",
161)),
Array("NAME:ModifierData", Array("NAME:ThermalModifierData", "modifier_data:=", "thermal_
```

```
modifier_data",
Array("NAME:all_thermal_modifiers"))), Array("NAME:permeability", "property_type:=", "non-
linear", "BTypeForSingleCurve:=", "normal", "HUnit:=", "A_per_meter", "BUnit:=", "tesla",
"IIsTemperatureDependent:=", false,
Array("NAME:BHCoordinates",
Array("NAME:DimUnits", "", ""),
Array("NAME:Point", 0, 0),
Array("NAME:Point", 159.2, 0.2402),
Array("NAME:Point", 318.3, 0.8654),
Array("NAME:Point", 477.5, 1.1106),
Array("NAME:Point", 636.6, 1.2458),
Array("NAME:Point", 795.8, 1.331),
Array("NAME:Point", 1591.5, 1.5),
Array("NAME:Temperatures")), "conductivity:=", "2000000", "magnetic_loss_tangent:=", "0",
Array("NAME:magnetic_coercivity",
"property_type:=", "VectorProperty",
"Magnitude:=", "0A_per_meter",
"DirComp1:=", "1", "DirComp2:=", "0", "DirComp3:=", "0"),
"thermal_conductivity:=", "45",

Array("NAME:core_loss_type",
"property_type:=", "ChoiceProperty",
"Choice:=", "B-P Curve"),
"core_loss_kdc:=", "0",
"mass_density:=", "7872",
"specific_heat:=", "481",
"youngs_modulus:=", "200000000000",
"poissons_ratio:=", "0.25",
"thermal_expansion_coefficient:=", "1.26e-05",
"core_loss_equiv_cut_depth:=", "0.001meter",
Array("NAME:core_loss_curves",
"property_type:=", "nonlinear",
"PUnit:=", "kw/m^3",
"BUnit:=", "tesla",
"Frequency:=", "60Hz",
```

```
"Thickness:=", "0.5mm",
"IsTemperatureDependent:=", true,
Array("NAME:BPCoordinates",
Array("NAME:DimUnits", "", ""),
Array("NAME:Temperatures",
Array("NAME:Temperature", "TempRef:=", "10cel",
Array("NAME:BPCoordinates",
Array("NAME:DimUnits", "", ""),
Array("NAME:Point", 0, 0),
Array("NAME:Point", 1, 2),
Array("NAME:Point", 3, 4)),
Array("NAME:Temperature", "TempRef:=", "20cel",
Array("NAME:BPCoordinates",
Array("NAME:DimUnits", "", ""),
Array("NAME:Point", 0, 0),
Array("NAME:Point", 11, 21),
Array("NAME:Point", 31, 41))))
```

<b>Python Syntax</b>	EditMaterial (<OriginalName>, "NAME:<NewName>", <MatProperty>, <MatProperty>, ...)
<b>Python Example</b>	# Example using a Thermal Modifier for Electrical Steel Core Loss properties #(applicable to Electrical Steel and Power Ferrite core loss models #in 3D Eddy Current and 3D Transient designs)

```
oDefinitionManager>EditMaterial("copper",
[
    "NAME:copper",
    "CoordinateSystemType:=", "Cartesian",
    "BulkOrSurfaceType:=" , 1,
    [
        "NAME:PhysicsTypes",
        "set:=" , ["Electromagnetic","Thermal","Structural"]
    ],
    [
        "NAME:AttachedData",
        [
            "NAME:MatAppearanceData",
            "property_data:=" , "appearance_data"
        ]
    ],
    [
        "NAME:ModifierData",
        [
            "NAME:ThermalModifierData",
            "modifier_data:=" , "thermal_modifier_data",
            [
                "NAME:all_thermal_modifiers",
                [
                    "NAME:one_thermal_modifier",
                    "Property:::" , "core_loss_kh",
                    "Index:::" , 0,
                    "prop_modifier:=" , "thermal_modifier",
                    "use_free_form:=" , True,
                    "free_form_value:=" , "if(Temp > 1000cel, 1,
                ],
                [
                    [
                        "NAME:two_thermal_modifier",
                        "Property:::" , "core_loss_kh",
                        "Index:::" , 1,
                        "prop_modifier:=" , "thermal_modifier",
                        "use_free_form:=" , True,
                        "free_form_value:=" , "if(Temp > 1000cel, 1,
                    ],
                    [
                        [
                            "NAME:three_thermal_modifier",
                            "Property:::" , "core_loss_kh",
                            "Index:::" , 2,
                            "prop_modifier:=" , "thermal_modifier",
                            "use_free_form:=" , True,
                            "free_form_value:=" , "if(Temp > 1000cel, 1,
                        ],
                        [
                            [
                                "NAME:four_thermal_modifier",
                                "Property:::" , "core_loss_kh",
                                "Index:::" , 3,
                                "prop_modifier:=" , "thermal_modifier",
                                "use_free_form:=" , True,
                                "free_form_value:=" , "if(Temp > 1000cel, 1,
                            ],
                            [
                                [
                                    "NAME:five_thermal_modifier",
                                    "Property:::" , "core_loss_kh",
                                    "Index:::" , 4,
                                    "prop_modifier:=" , "thermal_modifier",
                                    "use_free_form:=" , True,
                                    "free_form_value:=" , "if(Temp > 1000cel, 1,
                                ],
                                [
                                    [
  "NAME:six_thermal_modifier",
  "Property:::" , "core_loss_kh",
  "Index:::" , 5,
  "prop_modifier:=" , "thermal_modifier",
  "use_free_form:=" , True,
  "free_form_value:=" , "if(Temp > 1000cel, 1,
                                    ],
                                    [
  [
  "NAME:seven_thermal_modifier",
  "Property:::" , "core_loss_kh",
  "Index:::" , 6,
  "prop_modifier:=" , "thermal_modifier",
  "use_free_form:=" , True,
  "free_form_value:=" , "if(Temp > 1000cel, 1,
  ],
  [
  [
  "NAME:eight_thermal_modifier",
  "Property:::" , "core_loss_kh",
  "Index:::" , 7,
  "prop_modifier:=" , "thermal_modifier",
  "use_free_form:=" , True,
  "free_form_value:=" , "if(Temp > 1000cel, 1,
  ],
  [
  [
  "NAME:nine_thermal_modifier",
  "Property:::" , "core_loss_kh",
  "Index:::" , 8,
  "prop_modifier:=" , "thermal_modifier",
  "use_free_form:=" , True,
  "free_form_value:=" , "if(Temp > 1000cel, 1,
  ],
  [
  [
  "NAME:ten_thermal_modifier",
  "Property:::" , "core_loss_kh",
  "Index:::" , 9,
  "prop_modifier:=" , "thermal_modifier",
  "use_free_form:=" , True,
  "free_form_value:=" , "if(Temp > 1000cel, 1,
  ],
  [
  [
  "NAME:eleven_thermal_modifier",
  "Property:::" , "core_loss_kh",
  "Index:::" , 10,
  "prop_modifier:=" , "thermal_modifier",
  "use_free_form:=" , True,
  "free_form_value:=" , "if(Temp > 1000cel, 1,
  ],
  [
  [
  "NAME:twelve_thermal_modifier",
  "Property:::" , "core_loss_kh",
  "Index:::" , 11,
  "prop_modifier:=" , "thermal_modifier",
  "use_free_form:=" , True,
  "free_form_value:=" , "if(Temp > 1000cel, 1,
  ],
  ],
  ],
  ],
  ],
  ],
                                    ],
                                ],
                            ],
                        ],
                    ],
                ],
            ],
        ],
    ],
],
[
    [
        [
            [
                [
                    [
                        [
                            [
                                [
                                    [
  [
  [
  [
  [
  [
  [
  [
  [
  [
  [
  [
  [
  [
  [
  [
   [
  [
  [
   [
..
```

```
lue:=" , "if(Temp > 1000cel, 1, if(Temp -273.15cel, 1, 1))"
    ],
    [
        "NAME:one_thermal_modifier",
        "Property:::" , "core_loss_ke",
        "Index:::" , 0,
        "prop_modifier:::" , "thermal_modifier",
        "use_free_form:::" , True,
        "free_form_value:::" , "if(Temp > 1000cel, 1,
    ],
    [
        "NAME:one_thermal_modifier",
        "Property:::" , "core_loss_kdc",
        "Index:::" , 0,
        "prop_modifier:::" , "thermal_modifier",
        "use_free_form:::" , True,
        "free_form_value:::" , "if(Temp > 1000cel, 1,
    ],
    [
        "NAME:one_thermal_modifier",
        "Property:::" , "core_loss_equiv_cut_c",
        "Index:::" , 0,
        "prop_modifier:::" , "thermal_modifier",
        "use_free_form:::" , True,
        "free_form_value:::" , "if(Temp > 1000cel, 1,
    ]
],
],
],
"permeability:::" , "0.999991",
"conductivity:::" , "27840000",
"thermal_conductivity:::" , "400",
```


## GetProjectMaterialNames

Returns the material names belonging to an active Project.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name	Type	Description
	None		
<b>Return Value</b>	String names of the materials in the active project.		

<b>Python Syntax</b>	GetProjectMaterialNames()
<b>Python Example</b>	<pre> oProject = oDesktop.GetActiveProject() oDefinitionManager = oProject.GetDefinitionManager() materials = oDefinitionManager.GetProjectMaterialNames() AddWarningMessage(str(materials)) </pre>

## ExportMaterial

Exports a local material to a library.

<b>UI Access</b>	<b>Export to Library</b> command in the material editor.		
<b>Parameters</b>	Name	Type	Description
	<ExportData>	Array	["NAME:<LibraryName>","<MaterialName>, <MaterialName>, ..."]
	<LibraryName>	String	Name of the exported library.
	<MaterialName>	String	Name of the material to be exported.
<b>Return Value</b>	<LibraryLocation>	String	Location to save the library. Only "PersonalLib" and "UserLib" are allowed.
	None.		

<b>Python Syntax</b>	ExportMaterial (<ExportData>, <LibraryLocation>)
<b>Python Example</b>	<pre>oDefinitionManager.ExportMaterial ([ "NAME:mylib",_ "Material1", "Material2", "Material3"], "PersonalLib")</pre>

<b>VB Syntax</b>	ExportMaterial <ExportData>, <LibraryLocation>
<b>VB Example</b>	<pre>oDefinitionManager.ExportMaterial Array("NAME:mylib",_ "Material1", "Material2", "Material3"), "PersonalLib"</pre>

## RemoveMaterial

Removes a material from a library.

<b>UI Access</b>	<b>Remove Material(s)</b> command in the material editor		
<b>Parameters</b>	Name	Type	Description
	<MaterialName>	String	Name of the material to be removed.
	<IsProjectMaterial>	Boolean	If True, Maxwell assumes the material is a project material. The last two parameters will be ignored. If False, the material is not a project material.
	<LibraryName>	String	Name of the user or personal library where the material resides.
	<LibraryLocation>	String	Location of library. Valid options:"UserLib" or "PersonalLib".
<b>Return Value</b>	None.		

<b>Python Syntax</b>	RemoveMaterial (<MaterialName>, <IsProjectMaterial>, <LibraryName>, <LibraryLocation>)
<b>Python Example</b>	<pre>oDefinitionManager.RemoveMaterial ([     "Material1", false, "mo0907", "UserLib"])</pre>

<b>VB Syntax</b>	RemoveMaterial <MaterialName>, <IsProjectMaterial>, <LibraryName>, <LibraryLocation>
<b>VB Example</b>	<pre>oDefinitionManager.RemoveMaterial     "Material1", false, "mo0907", "UserLib"</pre>

## RemoveUnusedDefinitions

Removes any unused project definitions.

<b>UI Access</b>	Tools > Project Tools > Remove Unused Definitions.						
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;Definitions&gt;</td> <td>Array</td> <td>Definitions to be removed, such as materials and surface materials.</td> </tr> </tbody> </table>	Name	Type	Description	<Definitions>	Array	Definitions to be removed, such as materials and surface materials.
Name	Type	Description					
<Definitions>	Array	Definitions to be removed, such as materials and surface materials.					
<b>Return Value</b>	None.						

<b>Python Syntax</b>	RemoveUnusedDefinitions(<Definitions>)
<b>Python Example</b>	<pre> oProject.RemoveUnusedDefinitions (     [         [             "NAME:Materials",             "Al-Extruded"         ],         [             "NAME:SurfaceMaterials",             "Steel-oxidised-surface"         ]     ] ) </pre>

<b>VB</b>	RemoveUnusedDefinitions <Definitions>
-----------	---------------------------------------

Syntax	
<b>VB Example</b>	<pre>oProject.RemoveUnusedDefinitions Array(Array("NAME:Materials", "Al-Extruded"), Array("NAME:SurfaceMaterials", "Steel-oxidised-surface"))</pre>

## Component Manager Script Commands

The component manager provides access to components in a project. The manager object is accessed via the definition manager.

```
Set oDefinitionManager = oProject.GetDefinitionManager()
Set oComponentManager = oDefinitionManager.GetManager("Component")
```

**The topics for this section include:**

- [Add](#)
- [AddDynamicNPortData](#)
- [AddNPortData](#)
- [AddSolverOnDemandModel](#)
- [ClearSolutionCache](#)
- [Edit](#)
- [EditSolverOnDemandModel](#)
- [EditWithComps](#)
- [Export](#)
- [GetNPortData](#)
- [GetSolverOnDemandData](#)
- [GetSolverOnDemandModelList](#)

[Remove](#)

## RemoveSolverOnDemandMode

## UpdateDynamicLink

## Add [component manager]

## Use: Add a component

*Command:* Tools > Edit Configured Libraries > Components > Add Component

Syntax: Add Array("NAME:<ComponentName>",

```
"Info:=", <ComponentInfo>,  
"RefBase:=", <string>, // reference designator  
"NumParts:=", <int>, // parts per component  
"OriginalComponent:=", <string>  
"Terminal:=", <TerminalInfo>,  
"Terminal:=", <TerminalInfo>, ...  
// The remaining parameters are optional  
Array("NAME:Parameters", // any combo of the following  
"VariableProp:=", <VariableInfo>,  
"CheckboxProp:=", <CheckBoxInfo>,  
"ButtonProp:=", <ButtonInfo>,  
"TextProp:=", <TextInfo>,  
"NumberProp:=", <NumberInfo>,
```

```
"SeparatorProp:=", <SeparatorInfo>,
"ValueProp:=", <ValueInfo>,
"MenuProp:=", <MenuItem>),
Array("NAME:Properties", // any combo of the following
"CheckboxProp:=", <CheckBoxInfo>,
"TextProp:=", <TextInfo>,
"NumberProp:=", <NumberInfo>,
"SeparatorProp:=", <SeparatorInfo>,
"ValueProp:=", <ValueInfo>,
"MenuProp:=", <MenuItem>),
"VPointProp:=", <VPointInfo>,
"PointProp:=", <PointInfo>),
Array("Quantities",
"QuantityProp:=", <QuantityPropInfo>...),
Array("NAME:CosimDefinitions",
<CosimDefInfo>,
<CosimDefInfo>...)
```

*Return Value:<string>*

```
// composite name of the component.
// If the name requested conflicts with the name of an existing
// component, the requested name is altered to be unique.
```

```
// The name returned reflects any change made to be unique.
```

*Parameters:*<ComponentName>:

```
<string> // simple name of the component
```

```
<ComponentInfo>:
```

```
Array("Type:=", <TypeInfo>,
      "NumTerminals:=", <int>,
      "DataSource:=", <string>,
      "ModifiedOn:=", <ModifiedOnInfo>,
      "Manufacturer:=", "<string>",
      "Symbol:=", <string>,
      "Footprint:=", <string>,
      "Description:=", <string>,
      "InfoTopic:=", <string>,
      "InfoHelpFile:=", <string>,
      "IconFile:=", <string>,
      "LibraryName:=", "",
      "OriginalLocation:=", "Project", // Project Location
      "Author:=", <string>,
```

```
"OriginalAuthor:=", <string>,
"CreationDate:=", <int>)
```

#### <TypeInfo>:

An integer that is the or-ing of bits for each product listed below. The default setting is 0xffffffff (4294967295) which indicates valid for all products. In the component editing dialog, checking different boxes in the "Specify products for which this component is valid" grid control sets specific flags that correspond to the following hex/decimal settings:

Nexxim -- 100 binary, 4 decimal, 0x4  
SIwaveDeNovo -- 1000 binary, 8 decimal, 0x8  
Simplorer -- 10000 binary, 16 decimal, 0x10  
MaxwellCircuit -- 100000 binary, 32 decimal, 0x20

#### <ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

#### <TerminalInfo>:

```
Array(<string>, // symbol pin
      <string> // footprint pin
      <string>, // gate name
      <bool>, // shared
      <int>, // equivalence number
      <int>, // what to do if unconnected: flag as error:0, ignore:1
```

```
<string> // description  
<Nature>)
```

```
<Nature>:  
<string> // content varies as follows
```

Nexxim/Circuit:

```
"Electrical" // the only choice
```

Simplorer:

```
// several choices  
"Electrical", "Magnetic", "Fluidic", "Translational",  
"Translational_V", "Rotational", "Rotational_V",  
""Radian", "Thermal", or <VHDLPackageName>
```

```
<VHDLPackageName>:  
<string> // in the form <Library>.<Package>
```

```
<Library>:  
<string> // name of the VHDL library
```

```
<Package>:  
  <string> // name of the VHDL package  
  
<VariableInfo>:  
  Array(<string>, // name  
        <FlagLetters>,  
        <string>, // description  
        "CB:=", <string>, // optional - script for call back  
        <string>) // value: number, variable, or expression  
  
<FlagLetters>:  
  <string> // "D" - has description parameter,  
  // "RD" - readonly & has description parameter,  
  // or "RHD" - readonly, hidden, & has description parameter  
  
<CheckBoxInfo>:  
  Array(<string>, // name  
        <FlagLetters>,  
        <string>, // description  
        "CB:=", <string>, // optional - script for call back
```

```
<bool>) // value: true or false

<ButtonInfo>:
    Array(<string>, // name
          <FlagLetters>,
          <string>, // description
          <string>, // button title
          <string>, // extra text
          <ClientID>,
          "ButtonPropClientData:= ", <ClientdataArray>

<ClientID>:
    <int> // specifies Button Prop Client
    // 0 - unknown, ButtonPropClientData
    // array will be empty
    // 1 - Netlist Prop Client
    // 2 - not used
    // 3 - File Name Prop Client

<ClientdataArray>:
```

varies with <ClientID>

<ClientId> is 0 or 1: empty array

Array()

<ClientID> is 3:

Array("InternalFormatText:", "<prefix><RelativePath>")

<prefix>:

<string> // "<Project>", "<PersonalLib>", "<UserLib>", or "<SysLib>"

<RelativePath>:

<string> // relative path to file from <prefix>

<TextInfo>:

Array(<string>, // name

<FlagLetters>,

<string>, // description

"CB:=", <string>, // optional - script for call back

<string>) // value: a text string

```
<NumberInfo>:  
    Array(<string>, // name  
          <FlagLetters>,  
          <string>, // description  
          "CB:=", <string>, // optional - script for call back  
          <real>, // value: a number  
          <string>) // units  
  
<SeparatorInfo>:  
    Array(<string>, // name  
          <FlagLetters>,  
          <string>, // description  
          "CB:=", <string>, // optional - script for call back  
          <string>) // value: a text string  
  
<ValueInfo>:  
    Array(<string>, // name  
          <FlagLetters>,  
          <string>, // description  
          "CB:=", <string>, // optional - script for call back  
          <string>) // value: a number, variable or expression
```

```
<MenuPropInfo>
  Array(<string>, // name
        <FlagLetters>,
        <string>, // description
        <string>, // menu choices - separated by commas
        <int>) // 0 based index of current menu choice

<VPointInfo>
  Array(<string>, // name
        <FlagLetters>,
        <string>, // description
        "CB:=", <string>, // optional - script for call back
        <string>, // x value: number with length units
        <string>) // y value: number with length units

<PointInfo>
  Array(<string>, // name
        <FlagLetters>,
        <string>, // description
        "CB:=", <string>, // optional - script for call back
        <real>, // x value
```

```
<real> // y value

<QuantityPropInfo>:
    Array(<string>, // name
          <FlagLetters>,
          <string>, // description
          <string>, // value
          <TypeString>,
          <TypeStringDependentInfo>)

<TypeString>:
    <string> // "Across", "Through", or "Free"

<TypeStringDependentInfo>:
    <TypeString> is "Free" :
        <string>, // direction: "In", "Out", "InOut", or "DontCare"
        // Following <string> is not present if direction is "DontCare"
        <string> // when to calculate: "BeforeAnalogSolver",
        // "BeforeStateGraph", "AfterStateGraph", or "DontCareWhen"
```

```
<TypeString> is "Across" or "Through":
```

```
<int>, // terminal 1  
<int> // terminal 2
```

```
<CosimDefInfo>:
```

```
Array("NAME:CosimDefinition",  
    "CosimulatorType:=", <int>,  
    "CosimDefName:=", <string> // "HFSS 3D Layout", "Circuit",  
    // "Custom", or "Netlist"  
    "IsDefinition:=", <bool>,  
    final array member(s) vary with CosimDefName)
```

```
final array members for HFSS 3D Layout:
```

```
"CosimStackup:=", <string>,  
"CosimDmbedRatio:=", <int>
```

```
final array members for Circuit:
```

```
"ExportAsNport:=", <int>,  
"UsePjt:=", <int>
```

```
final array member for Custom:
```

"DefinitionCompName:=", <string>

final array member for Netlist:

"NetlistString:=", <string>

*VB Example:*

```
Dim name  
  
oComponentManager.Add (Array("NAME:MyComponent", _  
"Info:=", Array("Type:=", 4294901767, _  
"NumTerminals:=", 2, _  
"DataSource:=", "", _  
"ModifiedOn:=", 1071096503, _  
"Manufacturer:=", "Ansys", _  
"Symbol:=", "bendo", _  
"Footprint:=", "BENDO", _  
"Description:=", "", _  
"InfoTopic:=", "", _  
"InfoHelpFile:=", "", _  
"IconFile:=", "", _  
"LibraryName:=", "", _
```

```
"OriginalLocation:=", "Project", _
"Author:=", "", _
"OriginalAuthor:=", "", _
"CreationDate:= ", 1147460679), _
"Refbase:=", "U", _
"NumParts:=", 1, _
"OriginalComponent:=", "", _
"Terminal:=", Array("n1", _
"n1", _
"A", _
false, _
0, _
1, _
"", _
"Electrical"), _
"Terminal:=", Array("n2", _
"n2", _
"A", _
false, _
1, _
0, _
```

```
""" , _  
"Electrical") , _  
Array("NAME:Parameters" , _  
"MenuProp:=", Array("CoSimulator" , _  
"D" , _  
""" , _  
"Default, HFSS 3D Layout, Circuit, Custom, Netlist" , _  
0) , _  
"ButtonProp:=", Array("CosimDefinition" , _  
"D" , _  
""" , _  
""" , _  
"Edit" , _  
0 , _  
"ButtonPropClientData:=", Array()), _  
Array("NAME:CosimDefinitions" , _  
Array("NAME:CosimDefinition" , _  
"CosimulatorType:=", 0 , _  
"CosimDefName:=", "HFSS 3D Layout" , _  
"IsDefinition:=", true , _
```

```
"CosimStackup:=", "Layout stackup", _  
"CosimDmbedRatio:=", 3), _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 1, _  
"CosimDefName:=", "Circuit", _  
"IsDefinition:=", true, _  
"ExportAsNport:=", 0, _  
"UsePjt:=", 0), _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 2, _  
"CosimDefName:=", "Custom", _  
"IsDefinition:=", true, _  
"DefinitionCompName:=", ""), _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 3, _  
"CosimDefName:=", "Netlist", _  
"IsDefinition:=", true, _  
"NetlistString:=", "") ))
```

**Python Syntax**

```
Add [("NAME:<ComponentName>",  
"Info:=", <ComponentInfo>,
```

```
"RefBase:=", <string>, // reference designator  
"NumParts:=", <int>, // parts per component  
"OriginalComponent:=", <string>  
"Terminal:=", <TerminalInfo>,  
"Terminal:=", <TerminalInfo>, ...  
The remaining parameters are optional.  
["NAME:Parameters", // any combo of the following  
"VariableProp:=", <VariableInfo>,  
"CheckboxProp:=", <CheckBoxInfo>,  
"ButtonProp:=", <ButtonInfo>,  
"TextProp:=", <TextInfo>,  
"NumberProp:=", <NumberInfo>,  
"SeparatorProp:=", <SeparatorInfo>,  
"ValueProp:=", <ValueInfo>,  
"MenuProp:=", <MenuInfo>],  
["NAME:Properties", Any combination of the following:  
"CheckboxProp:=", <CheckBoxInfo>,  
"TextProp:=", <TextInfo>,  
"NumberProp:=", <NumberInfo>,  
"SeparatorProp:=", <SeparatorInfo>,
```

	<pre>"ValueProp:=", &lt;ValueInfo&gt;, "MenuProp:=", &lt;MenuInfo&gt;, "VPointProp:=", &lt;VPointInfo&gt;, "PointProp:=", &lt;PointInfo&gt;], ["Quantities", "QuantityProp:=", &lt;QuantityPropInfo&gt;...], ["NAME:CosimDefinitions", &lt;CosimDefInfo&gt;, &lt;CosimDefInfo&gt;...]]</pre>
Python Example	<pre>oComponentManager.Add( [ "NAME:Component", "Info:=", [ "Type:=", 0, "NumTerminals:=", 0, "DataSource:=", "",  "ModifiedOn:=", 1467910752, "Manufacturer:=", "",  "Symbol:=", "Component", "ModelNames:=", "",  "Footprint:=", "",</pre>

```
"Description:=", "",  
"InfoTopic:=", "",  
"InfoHelpFile:=", "",  
"IconFile:=", "",  
"Library:=", "",  
"OriginalLocation:=", "Project",  
"IEEE:=", "",  
"Author:=", "",  
"OriginalAuthor:=", "",  
"CreationDate:=", 1467910746,  
"ExampleFile:=", ""],  
"Refbase:=", "U",  
"NumParts:=", 1,  
"ModSinceLib:=", True,  
"CompExtID:=", 2  
])
```

## AddDynamicNPortData [component manager]

Adds a component using the specified data

*Command:* Project Menu > Add Model > Add 2DExtractor Model

Project Menu > Add Model > Add HFSS Model

Project Menu > Add Model > Add Nexsys Matlab Model

Project Menu > Add Model > Add Nport Model

Project Menu > Add Model > Add Parametric Model

Project Menu > Add Model > Add Q3D Model

Project Menu > Add Model > Add SIwave Model

Project Menu > Add Model > Add State-space Model

Syntax: AddDynamicNPortData Array("NAME:<ComponentDataName>","  
"ComponentDataType:=", <DataType>,  
"name:=", <string>, // Name of the item  
"filename:=", <string>, // Path to the file to find the data>  
"numberofports:=", <int>,  
"DesignName:=", <string>, // Name of the internal design  
"SolutionName:=", <string>, // Name of the solution to reference  
"Simulate:=", <bool>,  
"CloseProject:=", <bool>,  
"SaveProject:=", <bool>,  
"RefNode:=", <bool>,  
"InterpY:=", <bool>, // true to choose interpolating  
"InterpAlg:=", <InterpolationAlgorithm>,  
"NewToOldMap:=", <NewToOldMapPairs>,  
"OldToNewMap:=", <OldToNewMapPairs>,

---

"PinNames:=", Array(<string>, <string>...))

*Return Value:* <string>

```
// composite name of the component.  
// If the name requested conflicts with the name of an existing  
// component, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

*Parameters:* <ComponentDataName>:

```
<string> // simple name of the component
```

<DataType>:

```
<string> // "DesignerData" or "Q3DData"
```

<InterpolationAlgorithm>:

```
<string> // "auto", "lin", "shadH", or "shadNH"
```

<NewToOldMapPairs>:

Array of name/value pairs such as "V1", "V2" that will have the new variable name first and the old variable name second.

<OldToNewMapPairs>:

Array of name/value pairs such as "V1", "V2" that will have the old variable name first and the new variable name second.

*VB Example:*

```
oComponentManager.AddDynamicNPortData  
Array("NAME:ComponentData", _  
"ComponentDataType:=", "DesignerData", _  
"name:=", "DesignerData", _  
"filename:=", _  
"C:/Program Files/AnsysEM/Designer/Examples/Projects/optiguides/optiguides.adsn", _  
"numberofports:=", 2, _  
"DesignName:=", "DesignerModell", _  
"SolutionName:=", "Setup1 : Adaptive_1", _  
"Simulate:=", true, _  
"CloseProject:=", false, _  
"SaveProject:=", true, _  
"RefNode:=", false, _  
"InterpY:=", true, _  
"InterpAlg:=", "auto", _  
"NewToOldMap:=", Array("nport_height:=", "$height", _  
"nport_length:=", "$length", _  
"nport_width:=", "$width"), _  
"OldToNewMap:=", Array("$height:=", "nport_height", _  
"$length:=", "nport_length", _
```

```
"$width:=", "nport_width"), _  
"PinNames:=", Array( "WavePort1:1", "WavePort2:1"))
```

## AddNPortData [component manager]

*Use:* Adds a component using the specified data

*Command:* Project Menu > Add Model > Add Nport Model

*Syntax:* AddNPortData Array("NAME:<ComponentDataName>>,

```
"ComponentDataType:=", "NportData",  
"name:=", <string>, // Name of the item  
"filename:=", <string>, // Path to the file to find the data  
"numberofports:=", <int>,  
"filelocation:=", <LocationType>,  
"domain:=", <string>, // "time" or "frequency"  
"datemode:=", <string> // "EnterData", "Import", or "Link"  
"devicename:=", <string>,  
"ImpedanceTab:=", <bool>,  
"NoiseDataTab:=", <bool>,  
"DCBehaviorTab:=", <bool>,  
"SolutionName:=", <string>,  
"displayformat:=", <DisplayInfo>,  
"datatype:=", <string>, // "SMatrix", "YMatrix", or "ZMatrix"
```

```
"ShowRefPin:=", <bool>,  
"RefNodeCheckbox:=", <bool>, ...  
<ProductOptionsInfo>)
```

*Return Value:* <string>

```
// composite name of the component.  
// If the name requested conflicts with the name of an existing  
// component, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

*Parameters:* <ComponentDataName>:

```
<string> // simple name of the component
```

<LocationType>:

```
<string> // one of "UsePath", "PersonalLib", "UserLib", "SysLib",  
// or "Project".
```

<dclInfo>:

```
<string> // one of "DCOpen", "DCShort", "DCShShort",  
// "DCNone", or "DCEmpty".
```

<DisplayInfo>:

```
<string> // one of "MagnitudePhase", "ReallImaginary",
// or "DbPhase".

<ProductOptionsInfo>:
// The remaining parameters differ by product

// HFSS 3D Layout - doesn't support interpolation/DC behavior
"DCOption:=", -1,
"InterpOption:=", -1,
"ExtrapOption:=", -1,
"DataType:=", 0

// Nexxim
"DCOption:=", <NexximDCOption>,
"InterpOption:=", <NexximInterpOption>,
"ExtrapOption:=", <NexximExtrapOption>,
"DataType:=", 2

<NexximDCOption>:
<int> // 0 : Zero Padding
```

```
// 1 : Same as last point  
// 2 : Linear extrapolation from last 2 points  
// 3 : Constant magnitude, linear phase extrapolation  
// 4 : Leave all signal lines open circuited  
// 5 : Short all signal lines together  
// 6 : Short all signal lines to ground
```

```
<NexximInterpOption>:
```

```
<int> // 0 : Step
```

```
// 1 : Linear
```

```
<NexximExtrapOption>:
```

```
<int> // 0 : Zero padding
```

```
// 1 : Same as last point
```

```
// 2 : Linear extrapolation from last 2 points
```

```
// 3 : Constant magnitude, linear phase extrapolation
```

```
<CircuitDCOption>:
```

```
<int> // 0 : Leave all signal lines open circuited
```

```
// 1 : Short all signal lines together
```

```
// 2 : Short all signal lines to ground
```

```
// 3 : Extrapolate from data provided (not recommended)
```

```
<CircuitInterpOption>:
```

```
<int> // 0 : Linear
```

```
// 1 : Cubic spline
```

```
// 2 : Rational polynomial
```

```
<CircuitExtrapOption>:
```

```
<int> // 0 : Same as interpolation
```

```
// 1 : Zero padding
```

```
// 2 : Same as last point
```

*VB Example:*

```
oComponentManager.AddNPortData Array("NAME:ComponentData", __  
"ComponentDataType:=", "NPortData", __  
"name:=", "NportData", __  
"filename:=", "", __  
"numberofports:=", 2, __  
"filelocation:=", "UsePath", __  
"domain:=", "frequency", __  
"datemode:=", "Import", __
```

```
"devicename:="", "", _  
"ImpedanceTab:=", true, _  
"NoiseDataTab:=", true, _  
"DCBehaviorTab:=", true, _  
"SolutionName:=", "", _  
"displayformat:=", "MagnitudePhase", _  
"datatype:=", "SMatrix", _  
"ShowRefPin:=", true, _  
"RefNodeCheckbox:=", false, _  
"DCOption:=", 3, _  
"InterpOption:=", 1, _  
"ExtrapOption:=", 3, _  
"DataType:=", 2, _  
"DCOption:=", 3, _  
"InterpOption:=", 1, _  
"ExtrapOption:=", 3, _  
"DataType:=", 2)
```

## AddSolverOnDemandModel

**Use:** This method looks for a local component of the name passed in, and to this component it adds an SOD model definition using the information passed in the VARIANT. It returns the name of the SOD model added.

*Parameters:* BSTR component name.

*Parameters:* VARIANT which is the SOD model data.

*Return Value:* Returns the name of the model added.

## **ClearSolutionCache [component manager]**

*Use:* Clear the solution cache for dynamic link component.

*Command:* Each of the following commands will clear the solution cache:

- Dynamic Link Item RCM > Clear Solution Cache
- Dynamic Link Component in schematic RCM > Clear Solution Cache

*Syntax:* ClearSolutionCache <Component Name>

*Return Value:* None

*Parameters:* <component name> is the name of the dynamic link component

*VB Example:*

```
oComponentManager.ClearSolutionCache "TeeModel1"
```

## **Edit [component manager]**

Modifies an existing component

*Command:* Tools > Edit Configured Libraries > Components > Edit Component

*Syntax:* Edit <ComponentName>,

```
    Array("NAME:<NewComponentName>",
          "Info:=", <ComponentInfo>,
          "RefBase:=", <string>, // reference designator
          "NumParts:=", <int>, // parts per component
          "OriginalComponent:=", <string>)
```

```
"Terminal:=", <TerminalInfo>,
"Terminal:=", <TerminalInfo>, ...

// The remaining parameters are optional

Array("NAME:Parameters", // any combo of the following

"VariableProp:=", <VariableInfo>,
"CheckboxProp:=", <CheckBoxInfo>,
"ButtonProp:=", <ButtonInfo>,
"TextProp:=", <TextInfo>,
"NumberProp:=", <NumberInfo>,
"SeparatorProp:=", <SeparatorInfo>,
"ValueProp:=", <ValueInfo>,
"MenuProp:=", <MenuInfo>),

Array("NAME:Properties", // any combo of the following

"CheckboxProp:=", <CheckBoxInfo>,
"TextProp:=", <TextInfo>,
"NumberProp:=", <NumberInfo>,
"SeparatorProp:=", <SeparatorInfo>,
"ValueProp:=", <ValueInfo>,
"MenuProp:=", <MenuInfo>),
"VPointProp:=", <VPointInfo>,
"PointProp:=", <PointInfo>),
```

```
Array("Quantities",
    "QuantityProp:=", <QuantityPropInfo>...),
    Array("NAME:CosimDefinitions",
        <CosimDefInfo>,
        <CosimDefInfo>...)
```

*Return Value:* <string>

```
// composite name of the component.  
// If the name requested conflicts with the name of an existing  
// component, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

*Parameters:* <ComponentName>:

```
<string> // composite name of the component to edit
```

```
<NewComponentName>:  
<string> // new simple name for the component
```

```
<ComponentInfo>:  
Array("Type:=", <TypeInfo>,  
    "NumTerminals:=", <int>,
```

```
"DataSource:=", <string>,
"ModifiedOn:=", <ModifiedOnInfo>,
"Manufacturer:=", "<string>,
"Symbol:=", <string>,
"Footprint:=", <string>,
"Description:=", <string>,
"InfoTopic:=", <string>,
"InfoHelpFile:=", <string>,
"IconFile:=", <string>,
"LibraryName:=", <string>,
"OriginalLocation:=", <string>, // Project Location
"Author:=", <string>,
"OriginalAuthor:=", <string>,
"CreationDate:=" , <int>)
```

#### <TypeInfo>:

An integer that is the or-ing of bits for each product listed below. The default setting is 0xffffffff (4294967295) which indicates valid for all products. In the component editing dialog, checking different boxes in the "Specify products for which this component is valid" grid control sets specific flags that correspond to the following hex/decimal settings:

Nexxim -- 100 binary, 4 decimal, 0x4

SIwaveDeNovo -- 1000 binary, 8 decimal, 0x8

Simplorer -- 10000 binary, 16 decimal, 0x10

MaxwellCircuit -- 100000 binary, 32 decimal, 0x20

**<ModifiedOnInfo>:**

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

**<TerminalInfo>:**

```
Array(<string>, // symbol pin  
<string> // footprint pin  
<string>, // gate name  
<bool>, // shared  
<int>, // equivalence number  
<int>, // what to do if unconnected: flag as error:0, ignore:1  
<string>, // description  
<Nature>)
```

**<Nature>:**

<string> // content varies as follows

Nexxim/Circuit:

"Electrical" // the only choice

Simplorer:

```
// several choices  
"Electrical", "Magnetic", "Fluidic", "Translational",  
"Translational_V", "Rotational", "Rotational_V",  
""Radian", "Thermal", or <VHDLPackageName>
```

<VHDLPackageName>:

```
<string> // in the form <Library>.<Package>
```

<Library>:

```
<string> // name of the VHDL library
```

<Package>:

```
<string> // name of the VHDL package
```

<VariableInfo>:

```
Array(<string>, // name  
<FlagLetters>,  
<string>, // description
```

```
"CB:=", <string>, // optional - script for call back  
<string>) // value: number, variable, or expression
```

```
<FlagLetters>:  
<string> // "D" - has description parameter,  
// "RD" - readonly & has description parameter,  
// or "RHD" - readonly, hidden, & has description parameter
```

```
<CheckBoxInfo>:  
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<bool>) // value: true or false
```

```
<ButtonInfo>:  
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
<string>, // button title
```

```
<string>, // extra text  
<ClientID>,  
"ButtonPropClientData:= ", <ClientdataArray>  
  
<ClientID>:  
  <int> // specifies Button Prop Client  
  // 0 - unknown, "ButtonPropClientData  
  // array will be empty  
  // 1 - Netlist Prop Client  
  // 2 - not used  
  // 3 - File Name Prop Client  
  
<ClientdataArray>:  
  varies with <ClientID>  
  
<ClientID> is 0 or 1: empty array  
Array()  
  
<ClientID> is 3:  
Array("InternalFormatText:=", "<prefix><RelativePath>")
```

```
<prefix>:  
<string> // "<Project>", "<PersonalLib>", "<UserLib>", or "<SysLib>"
```

```
<RelativePath>:  
<string> // relative path to file from <prefix>
```

```
<TextInfo>:  
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<string>) // value: a text string
```

```
<NumberInfo>:  
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<real>, // value: a number  
<string>) // units
```

```
<SeparatorInfo>
  Array(<string>, // name
        <FlagLetters>,
        <string>, // description
        "CB:=", <string>, // optional - script for call back
        <string>) // value: a text string

<ValueInfo>
  Array(<string>, // name
        <FlagLetters>,
        <string>, // description
        "CB:=", <string>, // optional - script for call back
        <string>) // value: a number, variable or expression

<MenuPropInfo>
  Array(<string>, // name
        <FlagLetters>,
        <string>, // description
        <string>, // menu choices - separated by commas
        <int>) // 0 based index of current menu choice

<VPointInfo>
```

```
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<string>, // x value: number with length units  
<string>) // y value: number with length units
```

```
<PointInfo>:  
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
"CB:=", <string>, // optional - script for call back  
<real>, // x value  
<real>) // y value
```

```
<QuantityPropInfo>:  
Array(<string>, // name  
<FlagLetters>,  
<string>, // description  
<string>, // value
```

```
<TypeString>,
<TypeStringDependentInfo>

<TypeString>:
<string> // "Across", "Through", or "Free"

<TypeStringDependentInfo>:

"Free" :
<string>, // direction: "In", "Out", "InOut", or "DontCare"
// Following <string> is not present if direction is "DontCare"
<string> // when to calculate: "BeforeAnalogSolver",
// "BeforeStateGraph", "AfterStateGraph", or "DontCareWhen"

"Across" or "Through"
<int>, // terminal 1
<int> // terminal 2

<CosimDefInfo>:
Array("NAME:CosimDefinition",
"CosimulatorType:=", <int>,
"CosimDefName:=", <string> // "HFSS3D", "Circuit",
```

```
// "Custom", or "Netlist"  
"IsDefinition:=", <bool>,  
final array member(s) vary with CosimDefName)
```

final array members for HFSS 3D Layout:

```
"CosimStackup:=", <string>,  
"CosimDmbedRatio:=", <int>
```

final array members for Circuit:

```
"ExportAsNport:=", <int>,  
"UsePjt:=", <int>
```

final array member for Custom:

```
"DefinitionCompName:=", <string>
```

final array member for Netlist:

```
"NetlistString:=", <string>
```

*VB Example:*

```
Dim name  
  
name = oComponentManager.Edit ("Nexxim Circuit Elements\BJTs:Level01_NPN", _
```

---

```
Array("NAME:Level01_NPN", _  
"Info:=", Array("Type:=", 4294901764,_  
"NumTerminals:=", 3, _  
"DataSource:=", "Ansys built-in component", _  
"ModifiedOn:=", 1152722112, _  
"Manufacturer:=", "", _  
"Symbol:=", "nexx_bjt_npn", _  
"Footprint:=", "", _  
"Description:=", "BJT, GP, NPN", _  
"InfoTopic:=", "NXBJT1.htm", _  
"InfoHelpFile:=", "nexximcomponents.chm", _  
"IconFile:=", "bjtsn.bmp", _  
"Library:=", "Nexxim Circuit Elements\BJTs", _  
"OriginalLocation:=", "SysLibrary ", _  
"Author:=", "", _  
"OriginalAuthor:=", "", _  
"CreationDate:=", 1152722102), _  
"Refbase:=", "Q", _  
"NumParts:=", 1, _  
"Terminal:=", Array("collector", _  
"collector", _
```

```
"A", _  
false, _  
6, _  
0, _  
"", _  
"Electrical"), _  
"Terminal:=", Array("base", _  
"base", _  
"A", _  
false, _  
7, _  
0, _  
"", _  
"Electrical"), _  
"Terminal:=", Array("emitter", _  
"emitter", _  
"A", _  
false, _  
8, _  
0, _
```

```
""" , _  
"Electrical") , _  
Array("NAME:Parameters", _  
"TextProp:=", Array("LabelID", _  
"HD", _  
"Property string for netlist ID", _  
"Q@ID") , _  
"TextProp:=", Array("MOD", _  
"D", _  
"Name of model data reference", _  
"required") , _  
"VariableProp:=", Array("AREA", _  
"D", _  
"Emitter area multiplying factor, which affects currents, resistances, and capacitances", _ "1"), _  
_  
"VariableProp=", Array("AREAB", _  
"D", _  
"Base AREA", _  
"1") , _  
"VariableProp:=", Array( "AREAC", _  
"D", _  
Collector AREA", _
```

```
"1"), _  
"VariableProp:=", Array("DTEMP", _  
"D", _  
"The difference between element and circuit temperature deg Cel)", _  
"0"), _  
"VariableProp:=", Array("M", _  
"D", _  
"Multiplier factor to simulate multiple BJTs in parallel", _ "1"), _  
"ButtonProp:=", Array("NexximNetlist", _  
"HD", _  
"", _  
"Q@ID %0 %1 %2 *MOD(@MOD) *AREA(AREA=@AREA) " & _  
" *AREAB(AREAB=@AREAB) *AREAC(AREAC=@" & _  
"AREAC) *DTEMP(DTEMP=@DTEMP) *M(M=@M) ", _  
"Q@ID %0 %1 %2 *MOD(@MOD) " & _ "*AREA(AREA=@AREA) AREAB(AREAB=@AREAB) *AREAC(AREAC=@" & _  
"AREAC) *DTEMP(DTEMP=@DTEMP) *M(M=@M) ", _  
1, _  
"ButtonPropClientData:=", Array()), _  
"TextProp:=", Array( "modelName", _  
"HD", _
```

```
""" , _  
"Q" )) )  
  
VB Example:  
  
Dim name2  
  
name2 = oComponentManager.Edit "MyComponent", _  
(Array("NAME:MyOtherComponent", _  
"Info:=", Array("Type:=", 4294901767, _  
"NumTerminals:=", 2, _  
"DataSource:=", "", _  
"ModifiedOn:=", 1071096503, _  
"Manufacturer:=", "Ansys", _  
"Symbol:=", "bendo", _  
"Footprint:=", "BENDO", _  
"Description:=", "", _  
"InfoTopic:=", "", _  
"InfoHelpFile:=", "", _  
"IconFile:=", "", _  
"LibraryName:=", "", _  
"OriginalLocation:=", "Project", _  
"Author:=", "", _  
"OriginalAuthor:=", "", _
```

```
"CreationDate:= ", 1147460679), _  
"Refbase:=", "U", _  
"NumParts:=", 1, _  
"OriginalComponent:=", "", _  
"Terminal:=", Array("n1", _  
"n1", _  
"A", _  
false, _  
0, _  
0, _  
"", _  
"Electrical"), _  
"Terminal:=", Array("n2", _  
"n2", _  
"A", _  
false, _  
1, _  
0, _  
"", _  
Electrical"), _
```

```
Array("NAME:Parameters", _  
"MenuProp:=", Array("CoSimulator", _  
"D", _  
"", _  
"Default, HFSS3D, Circuit, Custom, Netlist", _  
0), _  
"ButtonProp:=", Array("CosimDefinition", _  
"D", _  
"", _  
"", _  
"Edit", _  
0, _  
"ButtonPropClientData:=", Array()), _  
Array("NAME:CosimDefinitions", _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 0, _  
"CosimDefName:=", "HFSS3D", _  
"IsDefinition:=", true, _  
"CosimStackup:=", "Layout stackup", _  
"CosimDmbedRatio:=", 3), _  
Array("NAME:CosimDefinition", _
```

```
"CosimulatorType:=", 1, _  
"CosimDefName:=", "Circuit", _  
"IsDefinition:=", true, _  
"ExportAsNport:=", 0, _  
"UsePjt:=", 0), _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 2, _  
"CosimDefName:=", "Custom", _  
"IsDefinition:=", true, _  
"DefinitionCompName:=", ""), _  
Array("NAME:CosimDefinition", _  
"CosimulatorType:=", 3, _  
"CosimDefName:=", "Netlist", _  
"IsDefinition:=", true, _  
"NetlistString:=", "")))
```

**Python Syntax**

Edit <ComponentName>, ["NAME:<NewComponentName>", "Info:=", <ComponentInfo>, "RefBase:=", <string>, // reference designator
--------------------------------------------------------------------------------------------------------------------------------------

```
"NumParts:=", <int>, // parts per component  
"OriginalComponent:=", <string>  
"Terminal:=", <TerminalInfo>,  
"Terminal:=", <TerminalInfo>, ...  
#The remaining parameters are optional  
["NAME:Parameters", // any combo of the following  
"VariableProp:=", <VariableInfo>,  
"CheckboxProp:=", <CheckBoxInfo>,  
"ButtonProp:=", <ButtonInfo>,  
"TextProp:=", <TextInfo>,  
"NumberProp:=", <NumberInfo>,  
"SeparatorProp:=", <SeparatorInfo>,  
"ValueProp:=", <ValueInfo>,  
"MenuProp:=", <MenuInfo>],  
["NAME:Properties", # any combo of the following  
"CheckboxProp:=", <CheckBoxInfo>,  
"TextProp:=", <TextInfo>,  
"NumberProp:=", <NumberInfo>,  
"SeparatorProp:=", <SeparatorInfo>,  
"ValueProp:=", <ValueInfo>,  
"MenuProp:=", <MenuInfo>),
```

	<pre>"VPointProp:=", &lt;VPointInfo&gt;, "PointProp:=", &lt;PointInfo&gt;), ["Quantities", "QuantityProp:=", &lt;QuantityPropInfo&gt;...], ["NAME:CosimDefinitions", &lt;CosimDefInfo&gt;, &lt;CosimDefInfo&gt;...])</pre>
<b>Python Example</b>	<pre>name = oComponentManager.Edit ("Simplorer Circuit Elements\BJTs:Level01_NPN", _ ["NAME:Level01_NPN", "Info:=", ["Type:=", 4294901764,_ "NumTerminals:=", 3, "DataSource:=", "Ansoft built-in component",_ "ModifiedOn:=", 1152722112, "Manufacturer:=", "",_ "Symbol:=", "nexx_bjt_npn", "Footprint:=", "",_ "Description:=", "BJT, GP, NPN", "InfoTopic:=", "NXBJT1.htm",_ "InfoHelpFile:=", "nexximcomponents.chm", "IconFile:=", "bjtsn.bmp",_ "Library:=", "Nexxim Circuit Elements\BJTs",_ "OriginalLocation:=", "SysLibrary ", "Author:=", "",_ "OriginalAuthor:=", "", "CreationDate:=", 1152722102],_ "Refbase:=", "Q", "NumParts:=", 1, "Terminal:=", ["collector",_ "collector", "A", false, 6, 0, "", "Electrical"],_ "Terminal:=", ["base", "base", "A", false, _</pre>

```
7, 0, "", "Electrical"], "Terminal:=", ["emitter", _  
"emitter", "A", false, 8, 0, "", "Electrical"], _  
["NAME:Parameters", "TextProp:=", ["LabelID", _  
"HD", "Property string for netlist ID", _  
"Q@ID"], "TextProp:=", ["MOD", "D", _  
"Name of model data reference", "required"], _  
"VariableProp:=", ["AREA", "D", _  
"Emitter area multiplying factor, which affects  
currents, resistances, and capacitances", "1"], _  
"VariableProp:=", ["AREAB", "D", "Base AREA", _  
"1"], "VariableProp:=", ["AREAC", "D", "Collector AREA", _  
"1"], "VariableProp:=", ["DTEMP", "D", _  
"The difference between element and circuit temperature (deg Cel)", _  
"0"], "VariableProp:=", ["M", "D", _  
"Multiplier factor to simulate multiple BJTs in parallel", _  
"1"], "ButtonProp:=", ["NexximNetlist", "HD", "", _  
"Q@ID %0 %1 %2 *MOD(@MOD) *AREA(AREA=@AREA)" & _  
" *AREAB (AREAB=@AREAB) *AREAC (AREAC=@" &  
"AREAC) *DTEMP (DTEMP=@DTEMP) *M (M=@M)", _  
"Q@ID %0 %1 %2 *MOD (@MOD) " & "*AREA (AREA=@AREA)
```

	*AREAB (AREAB=@AREAB) *AREAC (AREAC=@" & _ "AREAC) *DTEMP (DTEMP=@DTEMP) *M (M=@M)", 1, _ "ButtonPropClientData:=", [], "TextProp:=", ["ModelName", "HD", "", "Q"])))
<b>Python Example 2</b>	name2 = oComponentManager.Edit ("MyComponent",_  (["NAME:MyOtherComponent", "Info:=", ["Type:=", 4294901767, _  "NumTerminals:=", 2, "DataSource:=", "", _  "ModifiedOn:=", 1071096503, "Manufacturer:=", "Ansoft", _ "Symbol:=", "bendo", "Footprint:=", "BENDO", _  "Description:=", "", "InfoTopic:=", "", _  "InfoHelpFile:=", "", "IconFile:=", "", _

```
"LibraryName:=", "", "OriginalLocation:=", "Project", _

"Author:=", "", "OriginalAuthor:=", "", _

"CreationDate:= ", 1147460679], "Refbase:=", "U", _

"NumParts:=", 1, "OriginalComponent:=", "", _

"Terminal:=", ["n1", "n1", "A", false, 0, 0, "", _

"Electrical"], "Terminal:=", ["n2", "n2", "A", _

false, 1, 0, "", Electrical"], ["NAME:Parameters", _

"MenuProp:=", ["CoSimulator", "D", "", _

"Default,Custom,Netlist", 0], "ButtonProp:=", ["CosimDefinition", _

"D", "", "", "Edit", 0, "ButtonPropClientData:=", []]], _
```

```
[ "NAME:CosimDefinitions", [ "NAME:CosimDefinition", _  
  
    "CosimulatorType:=", 0, "CosimDefName:=", "HFSS3D", _  
  
    "IsDefinition:=", true, "CosimStackup:=", "Layout stackup", _  
  
    "CosimDmbedRatio:=", 3], [ "NAME:CosimDefinition", _  
  
    "CosimulatorType:=", 1, "CosimDefName:=", "", _  
  
    "IsDefinition:=", true, "ExportAsNport:=", 0, _  
  
    "UsePjt:=", 0], [ "NAME:CosimDefinition", _  
  
    "CosimulatorType:=", 2, "CosimDefName:=", "Custom", _  
  
    "IsDefinition:=", true, "DefinitionCompName:=", ""], _
```

```
[ "NAME:CosimDefinition", "CosimulatorType:=", 3, _
  "CosimDefName:=", "Netlist", "IsDefinition:=", true, _
  "NetlistString:=", ""]])
```

## EditSolverOnDemandModel

*Use:* This method looks for a local component of the name passed in, and in this component it looks for an SOD model using the name passed in the second BSTR. It modifies the SOD model using the data in the VARIANT. It returns the name of the SOD model edited.

*Return Value:* Returns the name of the model edited.

*Parameters:* BSTR component name.

*Parameters:* BSTR SOD model name.

*Parameters:* VARIANT which is the new SOD model data (can include changed name).

## EditWithComps [component manager]

Edit an existing component.

*Command:* None

*Syntax:* EditWithComps <ComponentName>,

```
Array("NAME:<NewComponentName>",
  "ModTime:=", <ModifiedTimeInfo>,
  "Library:=", <string>, // Library name
  "LibLocation:=", <string>, // Project Location
```

```
<PinDefInfo>,
<PinDefInfo>,... // optional, to define pins
<GraphicsDataInfo>, // optional, to define graphics
<PropDisplayMapInfo>), // optional, to define property displays
Array(<ListOfComponentNames>) // Component names
```

*Return Value:* <string>

```
// composite name of the component.
// If the name requested conflicts with the name of an existing
// component, the requested name is altered to be unique.
// The name returned reflects any change made to be unique.
```

*Parameters:* <ComponentName>:

```
<string> // composite name of the component being edited
```

<NewComponentName>:

```
<string> // new simple name for the component
```

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed

since 00:00 hours, Jan 1, 1970 UTC from the system clock.

```
<PinDefInfo>:  
  Array("NAME:PinDef",  
    "Pin:=", Array (<string>, // pin name  
      <real>, // x location  
      <real>, // y location  
      <real>, // angle in radians  
      <PinType>,  
      <real>, // line width  
      <real>, // line length  
      <bool>, // mirrored  
      <int>, // color  
      <bool>, // true if visible, false if not  
      <string>, // hidden net name  
      <OptionalPinInfo>, // optional info  
      <PropDisplayMapInfo>)) // optional
```

```
<PinType>:  
  <string> // "N" : normal pin  
  // "I" : input pin
```

```
// "O" : output pin

<OptionalPinInfo>:
// Specify both or neither
<bool>, // true if name is to be shown
<bool>, // true if number is to be shown

<PropDisplayMapInfo>:
Array("NAME:PropDisplayMap",
<PropDisplayInfo>,
<PropDisplayInfo>,...)

<PropDisplayInfo>:
<NameString>, Array(<DisplayTypeInfo>,
<DisplayLocationInfo>,
<int>, // optional, level number
<TextInfo>)

<NameString>:
<string> // PropertyName:=, where PropertyName is the name of
```

```
// the property to be displayed

<DisplayTypeInfo>
<int> // 0 : No display
// 1 : Display name only
// 2 : Display value only
// 3 : Display both name and value
// 4: Display evaluated value only
// 5: Display both name and evaluated value

<DisplayLocationInfo>
<int> // 0 : Left
// 1 : Top
// 2 : Right
// 3 : Bottom
// 4 : Center
// 5 : Custom placement

<GraphicsDataInfo>
Array("NAME:Graphics",
// one or more of the following
```

```
<RectInfo>,
<CircleInfo>,
<ArcInfo>,
<LineInfo>,
<PolygonInfo>,
<TextInfo>,
<ImageInfo>)
```

```
<RectInfo>:
"Rect:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // angle, in radians
<real>, // x position of center
<real>, // y position of center
<real>, // width
<real>) // height
```

```
<CircleInfo>:
"Circle:=", Array(<real>, // line width
```

```
<int>, // fill pattern  
<int>, // color  
<real>, // x position of center  
<real>, // y position of center  
<real>) // radius
```

<ArcInfo>:

```
"Arc:=", Array(<real>, // line width  
<int>, // line pattern  
<int>, // color  
<real>, // x position of center  
<real>, // y position of center  
<real>, // radius  
<real>, // start angle, in radians  
<end>) // end angle, in radians
```

<LineInfo>:

```
"Line:=", Array(<real>, // line width  
<int>, // line pattern  
<int>, // color  
<PointInfo>, // must specify at least 2 points
```

```
<PointInfo>...)

<PointInfo>:
    <real>, // x position
    <real> // y position

<PolygonInfo>:
    "Polygon:=", Array(<real>, // line width
    <int>, // fill pattern
    <int>, // color
    <PointInfo>, // must specify at least 3 points
    <PointInfo>...)

<TextInfo>:
    "Text:=", Array(<real>, // x position
    <real>, // y position
    <real>, // angle, in radians
    <Justification>,
    <bool>, // is plotter font
    <string>, // font name
    <int>, // color
```

```
<string>) // text string
```

```
<Justification>:
```

```
<int> // 0 : left top
```

```
// 1 : left base
```

```
// 2 : left bottom
```

```
// 3 : center top
```

```
// 4 : center base
```

```
// 5 : center bottom
```

```
// 6 : right top
```

```
// 7 : right base
```

```
// 8 : right bottom
```

```
<ImageInfo>:
```

```
"Image:=", Array(<RectInfo>,
```

```
<ImageData>,
```

```
<bool>) // is mirrored
```

```
<ImageData>:
```

```
<string>, // file path
```

```
<int>, // 0 : use the file path and link to it
```

```
// 1 : ignore file path and use next parameter  
<string> // text data, only present if preceding int is 1  
  
<ListOfComponentNames>:  
<string>,<string> ...  
  
// The list may be empty. When not empty, each string that is listed is a component  
// that references the component to be edited. Prior to editing, a clone of the component is  
// made, and the components that are listed are modified so that they now refer to  
// the clone.
```

*VB Example:*

```
Dim nam  
  
oModelManager>EditWithComps_  
("Nexxim Circuit Elements\\Distributed\\Distributed:bendo", _  
Array("NAME:bendo new name", _  
"ModTime:=", 1152722165, _  
"Library:=", "Nexxim Circuit Elements\\Distributed\\Distributed", _  
"LibLocation:=", "SysLibrary", _  
Array("NAME:PinDef", _  
"Pin:=", Array( "n1", _
```

```
-0.00254, _
0.00254, _
0, _
"N", _
0, _
0, _
false, _
0, _
true, _
"", _
false, _
false)), _
Array("NAME:PinDef", _
"Pin:=", Array( "n2", _
0.00254, _
-0.00254, _
1.5708, _
"N", _
0, _
0, _
false, _
```

```
0, _
true, _
"", _
false, _
false)), _
Array("NAME:Graphics", _
"Polygon:=", Array( 0, 0, 12566272, 0, 0.00381, 0, .00127, 0.00127, _
0.00127, 0.00127, 0, 0.00381, 0, 0.00381, _
0.002032, 0.00127, 0.00381), _
"Line:=", Array(0, 1, 12566272, -0.00254, 0.00254, 0, .00254), _
"Line:=", Array(0, 1, 12566272, 0.00254, -0.00254, .00254, 0)), _
Array("NAME:PropDisplayMap", _
"W:=", Array(3, _
5, _
0, _
"Text:=", Array( 2.1684E-019, _
0.00504119, _
0, _
1, _
5, _
```

```
false, _  
"Arial", _  
0, _  
"W=***")) ))), _  
Array("MY_COMP")
```

## Export [component manager]

Export component(s) to a library

*Command:* Tools > Edit Configured Libraries > Components > Export to Library

*Syntax:* Export Array("NAME:<LibraryName>","

```
<ComponentName>,  
<ComponentName>...),  
<LibraryLocation>
```

*Return Value:* None

*Parameters:* <LibraryName>:

```
<string> // name of the library
```

```
<ComponentName>:  
<string> // composite name of the component to export
```

```
<LibraryLocation>:
```

```
<string> // location of the library in <LibraryName>
// One of "Project", "PersonalLib", or "UserLib"
```

*VB Example:*

```
oComponentManager.Export Array("NAME:mylib", "Nexxim Circuit Elements\BJTs:Level01_NPN"), "PersonalLib"
```

<b>Python Syntax</b>	Export(["NAME:<LibraryName>", <ComponentName>, <ComponentName>...], <LibraryLocation>)
<b>Python Example</b>	<pre>oComponentManager.Export (["NAME:mylib",     "Simplorer Circuit Elements\BJTs:Level01_NPN"],     "PersonalLib")</pre>

## GetData [component manager]

Gets data that describes the definition.

*Command:* None

*Syntax:* GetData(<DefinitionName>)

*Return Value:* <DefinitionData> This is an array of data for the definition.

*Parameters:* <DefinitionName>:

```
<string> // composite name of the definition to edit
```

*VB Example:*

```
Dim compData
```

```
compData = oComponentManager.GetData("Level01_NPN")
```

**Note:**

GetData allows the user to access definition information, make modifications, and then use the Edit or EditWithComps script commands to save the modified definition. Accordingly, for each type of definition, the array data returned to GetData should be the same array information that is supplied to the [Edit](#) or [EditWithComps](#) commands.

<b>Python Syntax</b>	GetData(<DefinitionName>)
<b>Python Example</b>	compData = oComponentManager.GetData("Level01_NPN")

**GetNames [component manager]**

Returns the names of the components (used and unused) in a design. The following script command, **IsUsed**, can then be used to separate used and unused components.

*Command:* None

*Syntax:* GetNames()

*Return Value:* An array of strings

*Parameters:* None

*VB Example:*

```
Dim componentNames
```

```
componentNames = oComponentManager.GetNames()
```

<b>Python Syntax</b>	GetNames()
<b>Python Example</b>	componentNames = oComponentManager.GetNames()

## GetNPortData [component manager]

Returns NPort data for the component with the specified name.

*Command:* None

*Return Value:* Variant array, whose contents depend on the type of component. The array will be empty if the component does not have NPort data. See the syntax for AddDynamicNPortData and AddNPortData for descriptions of the array contents for components with those types of NPort data.

*Parameters:* <ComponentName>:

<string>

*Example:* This script displays each item in the returned array for each component in the design.

```
Sub DisplayVariant(x)
    Dim index
    index = 0

    For Each info In x
        curr = x(index)
        If TypeName(curr) <> "Variant()" Then
            If TypeName(curr) <> "Empty" And TypeName(curr) <> "Null" Then
                str = CStr(curr)
```

```
If str = "" Then
    str = ChrW(34) & ChrW(34)
End If
Msgbox str

Else
    str = "Empty/Null item."
    Msgbox str
End If

Else
    DisplayVariant curr
End If

index = index + 1

Next

End Sub
```

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim dnpInfo
```

```
Dim index
```

```
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.GetActiveProject()
Set oDefinitionManager = oProject.GetDefinitionManager()

Set oComponentManager = oDefinitionManager.GetManager("Component")
```

```
Dim componentNames
```

```
componentNames = oComponentManager.GetNames()
```

```
index = 0
```

```
For Each name In componentNames
```

```
    name = componentNames(index)
```

```
    message = "NPort data for component " + name
```

```
    MsgBox message
```

```
    dnpInfo = oComponentManager.GetNPortData(name)
```

```
DisplayVariant dnpInfo  
index = index + 1  
  
Next
```

## **GetSolverOnDemandData**

*Use:* This method looks for a local component of the name passed in, and in this component it looks for an SOD model of the name passed in and returns the SOD data pertaining to that model.

*Parameters:* BSTR component name.

*Parameters:* BSTR SOD model name

*Return Value:* VARIANT which is the SOD data.

## **GetSolverOnDemandModelList**

*Use:* This method looks for a local component of the name passed in, and returns a list of SOD model names defined in the component.

*Parameters:* BSTR component name.

*Return Value:* VARIANT which is a list of SOD model names.

## **IsUsed [component manager]**

Used to determine if a component is used in the design.

*Command:* None

*Syntax:* IsUsed(<ComponentName>)

*Return Value:* <Boolean> // true if the specified component is used in the design

*Parameters:* <ComponentName>:

<string>

*VB Example:*

```
Dim isUsed  
isUsed = oComponentManager.IsUsed("MyComponent")
```

<b>Python Syntax</b>	IsUsed(<ComponentName>)
<b>Python Example</b>	IsUsed = oComponentManager.IsUsed("MyComponent")

## Remove [component manager]

Remove a component from a library

*Command:* Tools > Edit Configured Libraries > Components > Remove Component

*Syntax:* Remove <ComponentName>,

```
<IsProjectComponent>,  
<LibraryName>,  
<LibraryLocation>
```

*Return Value:* None

*Parameters:* <ComponentName>:

```
<string> // composite name of the component to remove
```

```
<IsProjectComponent>:
```

```
<bool>
```

```

<LibraryName>:
<string> // name of the library

<LibraryLocation>:
<string> // location of the library in <LibraryName>
// One of "Project", "PersonalLib", or "UserLib"

```

*VB Example:*

```
oComponentManager.Remove "Nexxim Circuit Elements\BJTs:Level01_NPN", _ true, "Project"
```

<b>Python Syntax</b>	Remove (<ComponentName>, <IsProjectComponent>, <LibraryName>, <LibraryLocation>)
<b>Python Example</b>	oComponentManager.Remove ("Simplorer Circuit Elements\BJTs:Level01_NPN", _ true, "Project")

## RemoveSolverOnDemandModel

*Use:* This method looks for a local component of the name passed in, and in this component it looks for an SOD model of the name passed in and deletes the SOD model definition from the component.

*Parameters:* BSTR component name.

*Parameters:* BSTR SOD model name

*Return Value:* None.

## RemoveUnused [component manager]

Removes components that are not used in the design.

*Command:* **Project->Remove Unused Definitions** is similar but operates slightly different and does not record script commands.

*Syntax:* RemoveUnused()

*Return Value:* <bool> True if one or more components are removed.

*Parameters:* None

*VB Example:*

```
Dim removedDefs  
removedDefs = oComponentManager.RemoveUnused()
```

**Note:**

The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other components in dependent definitions to be rendered unusable.

Also, the symbol and footprint of an unused component are not unusable until after the component itself is removed using the Component Manager Remove script.

<b>Python Syntax</b>	RemoveUnused()
<b>Python Example</b>	removedDefs = oComponentManager.RemoveUnused()

## Update Dynamic Link [component manager]

*Use:* Reads data from the linked design and updates the dynamic link component. This will update the following: properties, solutions, ports, geometry.

*Command:* Each of the following commands will record a non-undoable script command:

- Dynamic Link Item RCM > Refresh Dynamic Link
- Dynamic Link Component in schematic RCM > Refresh Dynamic Link
- Click the Refresh Dynamic Link button in the Footprint tab of the Properties Window for selected Dynamic Link components in the Layout Editor.

*Syntax:* UpdateDynamicLink(<component name>)

*Return Value:* None

*Parameters:* <component name> is the name of the dynamic link component

*VB Example:*

```
oComponentManager.UpdateDynamicLink("TeeModel_L1")
```

## Material Manager Script Commands

The material manager provides access to materials in a project. The manager object is accessed via the definition manager.

```
Set oDefinitionManager = oProject.GetDefinitionManager()  
Set oMaterialManager = oDefinitionManager.GetManager("Material")
```

**The topics for this section include:**

[GetData](#)  
[GetNames](#)  
[GetProperties](#)  
[IsUsed](#)  
[RemoveUnused](#)

## GetData [material manager]

*Use:* Get material data

*Command:* None

*Syntax:* GetData <material\_name>

*Return Value:* Array of material data

*Parameters:* <material\_name>

*Type:* string

*Value:* Name of the project material

*VB Example:*

```
materialData = oMaterialMgr.GetData("MyMaterial2")
```

<b>Python Syntax</b>	GetData(<SimpleName>)
<b>Python Example</b>	dataArray = oMaterialManager.GetData ("mymaterial")

## GetNames [material manager]

*Use:* Get the names of the materials in a project

*Command:* None

*Syntax:* GetNames

*Return Value:* Names of the materials in a project

*Parameters:* None

*Example:*

```
materialNames = oMaterialMgr.GetNames()

Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow
Set oProject = oDesktop.NewProject
oProject.InsertDesign "Nexxim Circuit", "Nexxim1", "", ""
Set oMaterialMgr = oProject.GetDefinitionManager().GetManager("Material")
oMaterialMgr.Add( Array("NAME:MyMaterial1", "CoordinateSystemType:=", "Cartesian", "permittivity:=", "0.123") )
oMaterialMgr.Add( Array("NAME:MyMaterial2", "CoordinateSystemType:=", "Cartesian", "permittivity:=", "0.456") )
Dim materialNames
materialNames = oMaterialMgr.GetNames()
```

<b>Python Syntax</b>	GetNames()
<b>Python Example</b>	<pre>materialnames = oMaterialManager.GetNames()</pre>

## GetProperties [material manager]

Get material properties. Differs from GetData in that only material properties available to the user are returned by GetProperties.

<b>UI Access</b>	NA						
<b>Parameters</b>	<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>&lt;material_name&gt;</td><td>string Boolean</td><td>Name of the project material  True or False. If you use False or only a single argument, then GetProperties returns only the properties that are different from the defaults.</td></tr></tbody></table>	Name	Type	Description	<material_name>	string Boolean	Name of the project material  True or False. If you use False or only a single argument, then GetProperties returns only the properties that are different from the defaults.
Name	Type	Description					
<material_name>	string Boolean	Name of the project material  True or False. If you use False or only a single argument, then GetProperties returns only the properties that are different from the defaults.					
<b>Return Value</b>	Array of material data						

<b>Python Syntax</b>	GetProperties (<materialname>)
<b>Python Example</b>	<pre>oMaterialManager.GetProperties("Gold", True) oMaterialManager.GetProperties("vacuum")</pre>

<b>VB Syntax</b>	GetProperties <material_name>
<b>VB Example</b>	<pre>materialProps = oMaterialMgr.GetProperties( "MyMaterial2" )</pre>

## IsUsed [material manager]

*Use:* Checks if a project material is in use

*Command:* None

*Syntax:* IsUsed <material\_name>

*Return Value:* Returns 'True' if the material is in use.

*Parameters:* <material\_name>

*Type:* string

*Value:* Name of the project material to check.

*VB Example:*

```
used = oMaterialMgr.IsUsed( "MyMaterial2" )
```

<b>Python Syntax</b>	IsUsed(<ComboName>)
<b>Python Example</b>	isused = oMaterialManager.IsUsed("mylib:mymaterial")

## RemoveUnused [material manager]

*Use:* Remove all unused materials from the project.

*Command:* None

*Syntax:* RemoveUnused

*Return Value:* None

*Parameters:* None

*VB Example:*

```
oMaterialMgr.RemoveUnused()
```

<b>Python Syntax</b>	RemoveUnused()
<b>Python Example</b>	<pre>thereWereExtras = oMaterialManager.RemoveUnused()</pre>

## Model Manager Script Commands

The model manager provides access to models in a project. The manager object is accessed via the definition manager.

```
Set oDefinitionManager = oProject.GetDefinitionManager()  
Set oModelManager = oDefinitionManager.GetManager("Model")
```

The model manager script commands are listed below:

[Add](#)

[ConvertToDynamic](#)

[ConvertToParametric](#)

[Edit](#)

[EditWithComps](#)

[Export](#)

[GetData](#)

[GetNames](#)

[IsUsed](#)

[Remove](#)

[RemoveUnused](#)

## Add [model manager]

*Use:* Add a model

*Command:* Tools > Edit Configured Libraries > Models > Add Model

*Syntax:* Add Array("NAME:<modelName>","

```
"ModTime:=", <ModifiedTimeInfo>,  
"Library:="", "", // Library name  
"LibLocation:=", "Project", // Project Location  
<PinDefInfo>,  
<PinDefInfo>,... // optional, to define pins  
<GraphicsDataInfo>, // optional, to define graphics  
<PropDisplayMapInfo>)) // optional, to define property displays
```

*Return Value:* <string>

```
// composite name of the model.  
// If the name requested conflicts with the name of an existing  
// model, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

*Parameters:* <modelName>:

<string> // simple name of the model being added

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<PinDefInfo>:

```
Array("NAME:PinDef",
"Pin:=", Array (<string>, // pin name
<real>, // x location
<real>, // y location
<real>, // angle in radians
<PinType>,
<real>, // line width
<real>, // line length
<bool>, // mirrored
<int>, // color
<bool>, // true if visible, false if not
<string>, // hidden net name
<OptionalPinInfo>, // optional info
<PropDisplayMapInfo>)) // optional
```

```
<PinType>:  
  <string> // "N" : normal pin  
  // "I" : input pin  
  // "O" : output pin  
  
<OptionalPinInfo>:  
  // Specify both or neither  
  <bool>, // true if name is to be shown  
  <bool>, // true if number is to be shown  
  
<PropDisplayMapInfo>:  
  Array("NAME:PropDisplayMap",  
    <PropDisplayInfo>,  
    <PropDisplayInfo>,...)  
  
<PropDisplayInfo>:  
  <NameString>, Array(<DisplayTypeInfo>,  
    <DisplayLocationInfo>,  
    <int>, // optional, level number  
    <TextInfo>)
```

```
<NameString>:  
  <string> // PropertyName:=, where PropertyName is the name of  
  // the property to be displayed  
  
<DisplayTypeInfo>:  
  <int> // 0 : No display  
  // 1 : Display name only  
  // 2 : Display value only  
  // 3 : Display both name and value  
  // 4: Display evaluated value only  
  // 5: Display both name and evaluated value  
  
<DisplayLocationInfo>:  
  <int> // 0 : Left  
  // 1 : Top  
  // 2 : Right  
  // 3 : Bottom  
  // 4 : Center  
  // 5 : Custom placement
```

```
<GraphicsDataInfo>
  Array("NAME:Graphics",
    // one or more of the following
    <RectInfo>,
    <CircleInfo>,
    <ArcInfo>,
    <LineInfo>,
    <PolygonInfo>,
    <TextInfo>,
    <ImageInfo>)
```

```
<RectInfo>
  "Rect:=", Array(<real>, // line width
    <int>, // fill pattern
    <int>, // color
    <real>, // angle, in radians
    <real>, // x position of center
    <real>, // y position of center
    <real>, // width
    <real>) // height
```

```
<CircleInfo>:  
"Circle:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<real>, // x position of center  
<real>, // y position of center  
< real>) // radius
```

```
<ArcInfo>:  
"Arc:=", Array(<real>, // line width  
<int>, // line pattern  
<int>, // color  
<real>, // x position of center  
<real>, // y position of center  
< real>, // radius  
<real>, // start angle, in radians  
<end>) // end angle, in radians
```

```
<LineInfo>:  
"Line:=", Array(<real>, // line width
```

```
<int>, // line pattern  
<int>, // color  
<PointInfo>, // must specify at least 2 points  
<PointInfo>...)  
<PointInfo>:  
<real>, // x position  
<real> // y position  
  
<PolygonInfo>:  
"Polygon:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<PointInfo>, // must specify at least 3 points  
<PointInfo>...)  
  
<TextInfo>:  
"Text:=", Array(<real>, // x position  
<real>, // y position  
<real>, // angle, in radians  
<Justification>,  
<bool>, // is plotter font
```

```
<string>, // font name  
<int>, // color  
<string>) // text string
```

```
<Justification>:
```

```
<int> // 0 : left top  
// 1 : left base  
// 2 : left bottom  
// 3 : center top  
// 4 : center base  
// 5 : center bottom  
// 6 : right top  
// 7 : right base  
// 8 : right bottom
```

```
<ImageInfo>:
```

```
"Image:=", Array(<RectInfo>,  
<ImageData>,  
<bool>) // is mirrored
```

```
<ImageData>:  
  <string>, // file path  
  <int>, // 0 : use the file path and link to it  
  // 1 : ignore file path and use next parameter  
  <string> // text data, only present if preceding int is 1
```

*VB Example:*

```
oModelManager.Add Array("NAME:MyModel",_  
"ModTime:=", 1070989137, _  
"Library:=", "", _  
"LibLocation:=", "Project", _  
Array("NAME:PinDef", _  
"Pin:=", Array ("newpin0", _  
0.00254, _  
0, _  
0, _  
"N", _  
0, _  
0.00254, _  
false, _  
0, _  
true, _
```

```
""",_  
false, _  
false)), _  
Array("NAME:PinDef", _  
"Pin:=", Array ("newpin1", _  
-0.00254, _  
0, _  
3.14159265358979, _  
"N", _  
0, _  
0.00254, _  
false, _  
0, _  
true, _  
""",_  
false, _  
false)),  
Array("NAME:Graphics", _  
"Rect:=", Array(0, _  
0, _
```

```
12566272, _
0, _
4.33680868994202e-019, _
-0.000635, _
0.00508, _
0.002794), _
"Circle:=", Array(0, _
0, _
12566272, _
0.000127, _
-0.000635, _
0.000635)))
```

## ConvertToDynamic

*Use:* Build a new dynamic model based on an existing parametric model.

*Command:* Right-click on a model under Definitions/Models in the Project Tree and choose ConvertToDynamic.

*Syntax:* ConvertToDynamic(defName, newname)

*Return Value:* <newname> // Name of the new model added

*Parameters:* <defName> // Model that is the base for the new conversion

*VB Example:*

```
Dim newname
```

```
newname = oModelManager.ConvertToDynamic([in] BSTR defName, [out, retval] BSTR* newName);
```

## ConvertToParametric

*Use:* Build a new parametric model based on an existing dynamic model.

*Command:* Right-click on a model under Definitions/Models in the Project Tree and choose ConvertToParametric.

*Syntax:* ConvertToParametric(defName, newname)

*Return Value:* <newname> // Name of the new model added

*Parameters:* <defName> // Model that is the base for the new conversion

*VB Example:* Dim newname

```
newname = oModelManager.ConvertToParametric([in] BSTR defName, [out, retval] BSTR* newName);
```

## Edit [deprecated]

Deprecated command — please use [EditWithComps](#).

## EditWithComps [model manager]

*Use:* Edit an existing model.

*Command:* None

*Syntax:* EditWithComps <ModelName>,

```
Array("NAME:<NewModelName>",
      "ModTime:=", <ModifiedTimeInfo>,
      "Library:=", <string>, // Library name
      "LibLocation:=", <string>, // Project Location
      <PinDefInfo>,
      <PinDefInfo>,... // optional, to define pins
```

```
<GraphicsDataInfo>, // optional, to define graphics  
<PropDisplayMapInfo>), // optional, to define property displays  
Array(<ListOfComponentNames>) // Component names
```

*Return Value:* <string>

```
// composite name of the model.  
// If the name requested conflicts with the name of an existing  
// model, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

*Parameters:* <ModelName>:

```
<string> // composite name of the model being edited
```

<NewmodelName>:

```
<string> // new simple name for the model
```

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed  
since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<PinDefInfo>:

```
Array("NAME:PinDef",
  "Pin:=", Array (<string>, // pin name
    <real>, // x location
    <real>, // y location
    <real>, // angle in radians
    <PinType>,
    <real>, // line width
    <real>, // line length
    <bool>, // mirrored
    <int>, // color
    <bool>, // true if visible, false if not
    <string>, // hidden net name
    <OptionalPinInfo>, // optional info
    <PropDisplayMapInfo>)) // optional

<PinType>:
<string> // "N" : normal pin
// "I" : input pin
// "O" : output pin
```

```
<OptionalPinInfo>:  
// Specify both or neither  
<bool>, // true if name is to be shown  
<bool>, // true if number is to be shown  
  
<PropDisplayMapInfo>:  
Array("NAME:PropDisplayMap",  
<PropDisplayInfo>,  
<PropDisplayInfo>,...)  
  
<PropDisplayInfo>:  
<NameString>, Array(<DisplayTypeInfo>,  
<DisplayLocationInfo>,  
<int>, // optional, level number  
<TextInfo>)  
  
<NameString>:  
<string> // PropertyName:=, where PropertyName is the name of  
// the property to be displayed  
  
<DisplayTypeInfo>:
```

```
<int> // 0 : No display  
// 1 : Display name only  
// 2 : Display value only  
// 3 : Display both name and value  
// 4: Display evaluated value only  
// 5: Display both name and evaluated value
```

<DisplayLocationInfo>:

```
<int> // 0 : Left  
// 1 : Top  
// 2 : Right  
// 3 : Bottom  
// 4 : Center  
// 5 : Custom placement
```

<GraphicsDataInfo>:

```
Array("NAME:Graphics",  
// one or more of the following  
<RectInfo>,  
<CircleInfo>,
```

```
<ArcInfo>,
<LineInfo>,
<PolygonInfo>,
<TextInfo>,
<ImageInfo>

<RectInfo>:
"Rect:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // angle, in radians
<real>, // x position of center
<real>, // y position of center
<real>, // width
<real>) // height

<CircleInfo>:
"Circle:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // x position of center
```

```
<real>, // y position of center  
< real>) // radius
```

<ArcInfo>:

```
"Arc:=", Array(<real>, // line width  
<int>, // line pattern  
<int>, // color  
<real>, // x position of center  
<real>, // y position of center  
< real>, // radius  
<real>, // start angle, in radians  
<end>) // end angle, in radians
```

<LineInfo>:

```
"Line:=", Array(<real>, // line width  
<int>, // line pattern  
<int>, // color  
<PointInfo>, // must specify at least 2 points  
<PointInfo>...)  
<PointInfo>:
```

```
<real>, // x position  
<real> // y position  
  
<PolygonInfo>:  
"Polygon:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<PointInfo>, // must specify at least 3 points  
<PointInfo>...)  
  
<TextInfo>:  
"Text:=", Array(<real>, // x position  
<real>, // y position  
<real>, // angle, in radians  
<Justification>,  
<bool>, // is plotter font  
<string>, // font name  
<int>, // color  
<string>) // text string  
  
<Justification>:
```

```
<int> // 0 : left top  
// 1 : left base  
// 2 : left bottom  
// 3 : center top  
// 4 : center base  
// 5 : center bottom  
// 6 : right top  
// 7 : right base  
// 8 : right bottom
```

```
<ImageInfo>:  
"Image:=", Array(<RectInfo>,  
<ImageData>,  
<bool>) // is mirrored
```

```
<ImageData>:  
<string>, // file path  
<int>, // 0 : use the file path and link to it  
// 1 : ignore file path and use next parameter  
<string> // text data, only present if preceding int is 1
```

```
<ListOfComponentNames>:  
  <string>,<string> ...  
  // The list may be empty. When not empty, each string that is listed is a component  
  // that references the model to be edited. Prior to editing, a clone of the model is  
  // made, and the components that are listed are modified so that they now refer to  
  // the clone.
```

*VB Example:*

```
Dim nam  
  
oModelManager>EditWithComps_  
("Nexxim Circuit Elements\ Distributed\ Distributed:bendo", _  
Array("NAME:bendo new name", _  
"ModTime:=", 1152722165, _  
"Library:=", "Nexxim Circuit Elements\ Distributed\ Distributed", _  
"LibLocation:=", "SysLibrary", _  
Array("NAME:PinDef", _  
"Pin:=", Array( "n1", _  
-0.00254, _  
0.00254, _  
0, _
```

```
"N", _
0, _
0, _
false, _
0, _
true, _
"", _
false, _
false)), _
Array("NAME:PinDef", _
"Pin:=", Array( "n2", _
0.00254, _
-0.00254, _
1.5708, _
"N", _
0, _
0, _
false, _
0, _
true, _
```

```
""" , _  
false, _  
false)), _  
Array("NAME:Graphics", _  
"Polygon:=", Array( 0, 0, 12566272, 0, 0.00381, 0, .00127, 0.00127, _  
0.00127, 0.00127, 0, 0.00381, 0, 0.00381, _  
0.002032, 0.00127, 0.00381), _  
"Line:=", Array(0, 1, 12566272, -0.00254, 0.00254, 0, .00254), _  
"Line:=", Array(0, 1, 12566272, 0.00254, -0.00254, .00254, 0)), _  
Array("NAME:PropDisplayMap", _  
"W:=", Array(3, _  
5, _  
0, _  
"Text:=", Array( 2.1684E-019, _  
0.00504119, _  
0, _  
1, _  
5, _  
false, _  
"Arial", _  
0, _
```

```
"W=***") )) ) , _  
rray("MY_COMP")
```

## Export [model manager]

*Use:* Exports model(s) to a library

*Command:* Tools > Edit Configured Libraries > Models > Export to Library

*Syntax:* Export Array("NAME:<LibraryName>,"

```
<ModelName>,  
<ModelName>...),  
<LibraryLocation>
```

*Return Value:* None

*Parameters:* <LibraryName>:

```
<string> // name of the library
```

```
<ModelName>:
```

```
<string> // composite name of model to export
```

```
<LibraryLocation>:
```

```
<string> // location of the library in <LibraryName>
```

```
// One of "Project", "PersonalLib", or "UserLib"
```

*VB Example:*

```
oModelManager.Export  
Array("NAME Nexxim Circuit Elements\ Distributed\ Distributed:bendo:mylib", _ "myModel"), "Per-  
sonalLib"
```

<b>Python Syntax</b>	Export (["NAME:<LibraryName>", <ComboName>, <ComboName>...], <LibraryLocation>)
<b>Python Example</b>	oModelManager.Export ( ["NAME:mylib", "model1", "model2"] )

## GetData [model manager]

*Use:* Gets data that describes the definition.

*Command:* None

*Syntax:* GetData(<DefinitionName>)

*Return Value:* <DefinitionData> This is an array of data for the definition.

*Parameters:* <DefinitionName>:

<string> // [composite name](#) of the definition to edit

*VB Example:*

```
Dim ModelData
```

```
ModelData = oModelManager.GetData("Nexxim Circuit Elements\Hspice:nexx_bjt_npn")
```

**Note:**

GetData allows the user to access definition information, make modifications, and then use the Edit or EditWithComps script commands to save the modified definition. Accordingly, for each type of definition, the array data returned to GetData should be the same array information that is supplied to the [Edit](#) or [EditWithComps](#) commands.

<b>Python Syntax</b>	GetData(<ComboName>)
<b>Python Example</b>	dataArray = oModelManager.GetData ("mylib:mymodel")

## GetNames [model manager]

*Use:* Returns the names of the models (used and unused) in a design. The following script command, **IsUsed**, can then be used to separate used and unused models.

*Command:* None

*Syntax:* GetNames()

*Return Value:* An array of strings

*Parameters:* None

*VB Example:*

```
Dim modelNames
```

```
modelNames = oModelManager.GetNames()
```

<b>Python Syntax</b>	GetNames()
<b>Python Example</b>	modelnames = oModelManager.GetNames()

## IsUsed [model manager]

*Use:* Used to determine if a model is used in the design.

*Command:* None

*Syntax:* IsUsed(<ModelName>)

*Return Value:* <Boolean> // true if the specified model is used in the design

*Parameters:* <ModelName>:

<string>

*VB Example:*

```
Dim isUsed
isUsed = oModelManager.IsUsed("MyModel")
```

<b>Python Syntax</b>	IsUsed(<ComboName>)
<b>Python Example</b>	isused = oModelManager.IsUsed ("mylib:mymodel")

## Remove [model manager]

*Use:* Removes a model from a library

*Command:* Tools > Edit Configured Libraries > Models > Remove Model

*Syntax:* Remove <ModelName>,

<IsProjectModel>,

```
<LibraryName>,
<LibraryLocation>
```

*Return Value:* None

*Parameters:* <ModelName>:

```
<string> // composite name of the model to remove
```

**<IsProjectModel>:**

```
<bool>
```

**<LibraryName>:**

```
<string> // name of the library
```

**<LibraryLocation>:**

```
<string> // location of the library in <LibraryName>
```

```
// One of "Project", "PersonalLib", or "UserLib"
```

*VB Example:*

```
oModelManager.Remove "Nexxim Circuit Elements\ Distributed\ Distributed:bendo", true, "Project"
```

**Python Syntax**

Remove (<ModelName>, <IsLocal>, <LibraryName>, <LibraryLocation>)

**Python Example**

```
oModelManager.Export([  
    "NAME:mylib", "model1", "model2"])
```

**RemoveUnused [model manager]**

*Use:* Removes models that are not used in the design.

*Command:* **Project->Remove Unused Definitions** is similar but operates slightly different and does not record script commands.

*Syntax:* RemoveUnused()

*Return Value:* <bool> True if one or more models are removed.

*Parameters:* None

*VB Example:*

```
Dim removedDefs
```

```
removedDefs = oModelManager.RemoveUnused()
```

**Note:**

The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other models in dependent definitions to be rendered unusable.

Also, the model and footprint of an unused component are not unusable until after the component itself is removed using the Component Manager Remove script.

**Python Syntax**

```
RemoveUnused()
```

**Python Example**

```
thereWereExtras = oModelManager.RemoveUnused()
```

## Network Data Explorer Manager Script Commands

The network data Explorer (NDE) Manager provides access to certain NDE data.

```
Set oDefinitionManager = oProject.GetDefinitionManager()  
Set oNdExplorerManager = oDefinitionManager.GetManager("NdExplorer")
```

For NDE scripts accessed via the ndExplorer tool, see: [Network Data Explorer Script Commands](#).

**The topics for this section include:**

[ExportFullWaveSpice](#)

[ExportNetworkData](#)

[ExportNMFDATA](#)

### ExportFullWaveSpice

*Use:* Export FullWaveSpice data in a format of your choice.

*Command:* File > Export MacroModel > Broadband (SYZ, FWS....)

*Syntax:* ExportFullWaveSpice

```
"DesignName", // Design name. Can be left blank, if loading solution from a file.  
true/false, // true - solution loaded from file, false- loaded from design  
"Name", // If loading from design this is the solution name, else this is the  
// full path of the file from which the solution is loaded  
"variation", // Pick a particular variation. Leave blank if no variation.  
Array("NAME:Frequencies"), // Optional; if none defined all frequencies are used
```

```
Array("NAME:SpiceData", // Spice export options object
      "SpiceType:=", "SSS", // SpiceType can be "PSpice", "HSpice", "Spectre", "SSS",
           // "Simplorer", "TouchStone1.0", "TouchStone2.0"
      "EnforcePassivity:=", false, // Enforce Passivity true/false
      "EnforceCausality:=", false, // Enforce Causality true/false
      "UseCommonGround:=", false, // Use common ground true/false
      "FittingError:=", 0.5, // Fitting error
      "MaxPoles:=", 400, // Maximum Order
      "PassivityType:=", "ConvexOptimization", // Passivity Type can be "ConvexOptimization",
           // "PassivityByPerturbation", or "IteratedFittingOfPV"
      "ColumnFittingType:=", "Column", // Column FittingType can be "Column", "Entry", "Matrix"
      "SSFittingType:=", "TWA", // SS Fitting Type can be "TWA", "IterativeRational"
      "RelativeErrorTolerance:=", false, // Relative error tolerance true/false
      "TouchstoneFormat:=", "MA", // Touchstone Format "MA", "RI", "DB"
      "TouchstoneUnits:=", "Hz", // Touchstone Units "Hz", "KHz", "MHz", "MHz"
      "TouchStonePrecision:=", 8, // Touchstone precision
      "ExportDirectory:=", "C:/Examples/LNA/", // Directory to export to
      "ExportSpiceFileName:=", "Linckt_HBTest_2.sss", // Spice export file
      "FullwaveSpiceFileName:=", "Linckt_HBTest.sss", // FWS file
      "CreateNPortModel:=", true // Create a model based on the exported file true/false
    )
```

## ExportNetworkData

Exports matrix solution data to a file.

UI Access	N/A		
Parameters	Name	Type	Description
	<DesignVariationKey>	String	Design variation key. Pass empty string for the current nominal variation.
	<SolnSelectionArray>	Array	Array of selected solutions.  Array (<SolnSelector>, <SolnSelector>, ...)  If more than one array entry, this indicates a combined Interpolating sweep.
	<SolnSelector>	String	Solution setup name and solution name, separated by a colon.
	<FileFormat>	Integer	File format value.  2 : Tab delimited spreadsheet format (.tab)  3 : Touchstone (.sNp)  4 : CitiFile (.cit)  7 : Matlab (.m)  8 : Terminal Z0 spreadsheet
	<OutFile>	String	Full path to the file to write out.
	<FreqsArray>	Array	The frequencies to export. The <FreqsArray> argument contains a vector (e.g. "1GHz", "2GHz", ...) to use, or "all". To export all frequencies, use Array("all"). If no frequencies are specified, all frequencies are used.
	<DoRenorm>	Boolean	Specifies whether to renormalize the data before export.
	<RenormImped>	Double	Real impedance value in ohms, for renormalization. Required in syn-

		tax, but ignored if DoRenorm is false.
	<code>&lt;DataType&gt;</code>	String Optional. Type: "S", "Y", or "Z". The matrix to export.
	<code>&lt;pass&gt;</code>	Integer Optional. The pass to export. This is ignored if the sourceName is a frequency sweep. Leaving out this value or specifying -1 gets all passes.
	<code>&lt;ComplexFormat&gt;</code>	Integer Optional. Type: "0", "1", or "2"  The format to use for the exported data.  0 = Magnitude/Phase.  1= Real/Imaginary.  2= db/Phase.
	<code>&lt;Precision&gt;</code>	Integer Optional. Touchstone number of digits precision. Default if not specified is 15.
	<code>&lt;UseExportFreqs&gt;</code>	Boolean Specifies whether to use export frequencies.
	<code>&lt;IncludeGammaComments&gt;</code>	Boolean Specifies whether to include Gamma and Impedance comments.
	<code>&lt;SupportNonStdExport&gt;</code>	Boolean Specifies whether to support non-standard Touchstone extensions for mixed reference impedance.
<b>Return Value</b>	None.	

<b>Python Syntax</b>	ExportNetworkData(<DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>, <FreqsArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <SupportNonStdExport>])
<b>Python Example</b>	

<b>VB Syntax</b>	ExportNetworkData <DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>,
------------------	--------------------------------------------------------------------------------------------------------

	<FreqsArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <SupportNonStdExport>]
<b>VB Example</b>	<pre>oModule.ExportNetworkData "width='2in'", _     Array("Setup1:Sweep1"), 2, "c:\mydir\out.tab", _     Array("all"), false, 0 oModule.ExportNetworkData "width='2in'", _     Array("Setup1:Sweep1", "Setup1:Sweep2"), 3, _     "c:\mydir\out.s2p", Array(1.0e9, 1.5e9, 2.0e9), _     true, 50.0</pre>

## ExportNMFData

## Symbol Manager Script Commands

The symbol manager provides access to symbols in a project. The manager object is accessed via the definition manager.

```
Set oDefinitionManager = oProject.GetDefinitionManager()
Set oSymbolManager = oDefinitionManager.GetManager("Symbol")
```

The symbol manager script commands are listed below.

[Add](#)

[BringToFront](#)

[Edit](#)

[EditWithComps](#)

[Export](#)[GetData](#)[GetNames](#)[IsUsed](#)[Remove](#)[RemoveUnused](#)

## Add [symbol manager]

*Use:* Add a symbol

*Command:* Tools > Edit Configured Libraries > Symbols > Add Symbol

*Syntax:* Add Array("NAME:<SymbolName>,"

```
"ModTime:=", <ModifiedTimeInfo>,  
"Library:=", "", // Library name  
"LibLocation:=", "Project", // Project Location  
<PinDefInfo>,  
<PinDefInfo>,... // optional, to define pins  
<GraphicsDataInfo>, // optional, to define graphics  
<PropDisplayMapInfo>)) // optional, to define property displays
```

*Return Value:* <string>

```
// composite name of the symbol.  
// If the name requested conflicts with the name of an existing
```

```
// symbol, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

*Parameters:* <SymbolName>:

<string> // [simple name](#) of the symbol being added

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<PinDefInfo>:

```
Array("NAME:PinDef",  
    "Pin:=", Array (<string>, // pin name  
        <real>, // x location  
        <real>, // y location  
        <real>, // angle in radians  
        <PinType>,  
        <real>, // line width  
        <real>, // line length  
        <bool>, // mirrored
```

```
<int>, // color  
<bool>, // true if visible, false if not  
<string>, // hidden net name  
<OptionalPinInfo>, // optional info  
<PropDisplayMapInfo>)) // optional
```

```
<PinType>:  
<string> // "N" : normal pin  
// "I" : input pin  
// "O" : output pin
```

```
<OptionalPinInfo>:  
// Specify both or neither  
<bool>, // true if name is to be shown  
<bool>, // true if number is to be shown
```

```
<PropDisplayMapInfo>:  
Array("NAME:PropDisplayMap",  
<PropDisplayInfo>,  
<PropDisplayInfo>,...)
```

```
<PropDisplayInfo>:  
  <NameString>, Array(<DisplayTypeInfo>,  
  <DisplayLocationInfo>,  
  <int>, // optional, level number  
  <TextInfo>)  
  <NameString>:  
    <string> // PropertyName:=, where PropertyName is the name of  
    // the property to be displayed  
  
  <DisplayTypeInfo>:  
    <int> // 0 : No display  
    // 1 : Display name only  
    // 2 : Display value only  
    // 3 : Display both name and value  
    // 4: Display evaluated value only  
    // 5: Display both name and evaluated value  
  
  <DisplayLocationInfo>:  
    <int> // 0 : Left
```

```
// 1 : Top
// 2 : Right
// 3 : Bottom
// 4 : Center
// 5 : Custom placement

<GraphicsDataInfo>:
    Array("NAME:Graphics",
        // one or more of the following
        <RectInfo>,
        <CircleInfo>,
        <ArcInfo>,
        <LineInfo>,
        <PolygonInfo>,
        <TextInfo>,
        <ImageInfo>)

<RectInfo>:
    "Rect:=", Array(<real>, // line width
                    <int>, // fill pattern
                    <int>, // color
```

```
<real>, // angle, in radians  
<real>, // x position of center  
<real>, // y position of center  
<real>, // width  
< real>) // height
```

```
<CircleInfo>:  
"Circle:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<real>, // x position of center  
<real>, // y position of center  
< real>) // radius
```

```
<ArcInfo>:  
"Arc:=", Array(<real>, // line width  
<int>, // line pattern  
<int>, // color  
<real>, // x position of center  
<real>, // y position of center
```

```
<real>, // radius  
<real>, // start angle, in radians  
<end> // end angle, in radians  
  
<LineInfo>:  
"Line:=", Array(<real>, // line width  
<int>, // line pattern  
<int>, // color  
<PointInfo>, // must specify at least 2 points  
<PointInfo>...)  
<PointInfo>:  
<real>, // x position  
<real> // y position  
  
<PolygonInfo>:  
"Polygon:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<PointInfo>, // must specify at least 3 points  
<PointInfo>...)
```

```
<TextInfo>:  
"Text:=", Array(<real>, // x position  
<real>, // y position  
<real>, // angle, in radians  
<Justification>,  
<bool>, // is plotter font  
<string>, // font name  
<int>, // color  
<string>) // text string
```

```
<Justification>:  
<int> // 0 : left top  
// 1 : left base  
// 2 : left bottom  
// 3 : center top  
// 4 : center base  
// 5 : center bottom  
// 6 : right top  
// 7 : right base  
// 8 : right bottom
```

```
<ImageInfo>:  
  "Image:=", Array(<RectInfo>,  
    <ImageData>,  
    <bool>) // is mirrored  
  
<ImageData>:  
  <string>, // file path  
  <int>, // 0 : use the file path and link to it  
  // 1 : ignore file path and use next parameter  
  <string> // text data, only present if preceding int is 1
```

*VB Example:*

```
oSymbolManager.Add Array("NAME:MySymbol",_  
  "ModTime:=", 1070989137, _  
  "Library:=", "", _  
  "LibLocation:=", "Project", _  
  Array("NAME:PinDef", _  
    "Pin:=", Array ("newpin0", _  
      0.00254, _ 0, _  
      0, _  
      "N", _
```

```
0, _
0.00254, _
false, _
0, _
true, _
"",_
false, _
false)), _
Array("NAME:PinDef", _
"Pin:=", Array ("newpin1", _
-0.00254, _
0, _
3.14159265358979, _
"N",_
0, _
0.00254, _
false, _
0, _
true, _
"",_
```

```
false, _  
false)),  
Array("NAME:Graphics", _  
"Rect:=", Array(0, _  
0, _  
12566272, _  
0, _  
4.33680868994202e-019, _  
-0.000635, _  
0.00508, _  
0.002794), _  
"Circle:=", Array(0, _  
0, _  
12566272, _  
0.000127, _  
0.000635, _  
0.000635)))
```

## BringToFront [symbol manager]

**Use:** Changes the drawing for the symbol so that the specified objects are drawn on top of other overlapping objects.

**Command:** Draw > Bring To Front

*Syntax:* BringToFront Array("NAME:Selections", "Selections:=", Array (<Object>, <Object>, ...))

*Return Value:* None

*Parameters:* <Object>

<string> // object to bring to the front

*VB Example:*

```
oDefinitionEditor.BringToFront Array("NAME:Selections", "Selections:=", Array( "SchObj@10"))
```

## Edit [deprecated]

Deprecated command — please use [EditWithComps](#).

## EditWithComps [symbol manager]

*Use:* Edit an existing symbol.

*Command:* None

*Syntax:* EditWithComps <SymbolName>,

```
    Array("NAME:<NewSymbolName>",  
          "ModTime:=", <ModifiedTimeInfo>,  
          "Library:=", <string>, // Library name  
          "LibLocation:=", <string>, // Project Location  
          <PinDefInfo>,  
          <PinDefInfo>,... // optional, to define pins  
          <GraphicsDataInfo>, // optional, to define graphics
```

```
<PropDisplayMapInfo>), // optional, to define property displays  
Array(<ListOfComponentNames>) // Component names
```

*Return Value:* <string>

```
// composite name of the symbol.  
// If the name requested conflicts with the name of an existing  
// symbol, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

*Parameters:* <SymbolName>:

```
<string> // composite name of the symbol being edited
```

<NewSymbolName>:

```
<string> // new simple name for the symbol
```

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed  
since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<PinDefInfo>:

```
Array("NAME:PinDef",
```

```
"Pin:=", Array (<string>, // pin name  
                <real>, // x location  
                <real>, // y location  
                <real>, // angle in radians  
                <PinType>,  
                <real>, // line width  
                <real>, // line length  
                <bool>, // mirrored  
                <int>, // color  
                <bool>, // true if visible, false if not  
                <string>, // hidden net name  
                <OptionalPinInfo>, // optional info  
                <PropDisplayMapInfo>)) // optional  
  
<PinType>:  
    <string> // "N" : normal pin  
    // "I" : input pin  
    // "O" : output pin  
  
<OptionalPinInfo>:
```

```
// Specify both or neither  
<bool>, // true if name is to be shown  
<bool>, // true if number is to be shown  
  
<PropDisplayMapInfo>:  
  Array("NAME:PropDisplayMap",  
    <PropDisplayInfo>,  
    <PropDisplayInfo>,...)  
  
<PropDisplayInfo>:  
  <NameString>, Array(<DisplayTypeInfo>,  
    <DisplayLocationInfo>,  
    <int>, // optional, level number  
    <TextInfo>)  
  
<NameString>:  
  <string> // PropertyName:=, where PropertyName is the name of  
  // the property to be displayed  
  
<DisplayTypeInfo>:  
  <int> // 0 : No display
```

```
// 1 : Display name only  
// 2 : Display value only  
// 3 : Display both name and value  
// 4: Display evaluated value only  
// 5: Display both name and evaluated value
```

```
<DisplayLocationInfo>:
```

```
<int> // 0 : Left
```

```
// 1 : Top
```

```
// 2 : Right
```

```
// 3 : Bottom
```

```
// 4 : Center
```

```
// 5 : Custom placement
```

```
<GraphicsDataInfo>:
```

```
Array("NAME:Graphics",
```

```
// one or more of the following
```

```
<RectInfo>,
```

```
<CircleInfo>,
```

```
<ArcInfo>,
```

```
<LineInfo>,
<PolygonInfo>,
<TextInfo>,
<ImageInfo>

<RectInfo>:
"Rect:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // angle, in radians
<real>, // x position of center
<real>, // y position of center
<real>, // width
<real>) // height

<CircleInfo>:
"Circle:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // x position of center
<real>, // y position of center
```

```
<real>) // radius
```

```
<ArcInfo>:
```

```
"Arc:=", Array(<real>, // line width  
<int>, // line pattern  
<int>, // color  
<real>, // x position of center  
<real>, // y position of center  
<real>, // radius  
<real>, // start angle, in radians  
<end>) // end angle, in radians
```

```
<LineInfo>:
```

```
"Line:=", Array(<real>, // line width  
<int>, // line pattern  
<int>, // color  
<PointInfo>, // must specify at least 2 points  
<PointInfo>...)  
<PointInfo>:  
<real>, // x position
```

```
<real> // y position
```

```
<PolygonInfo>:
```

```
"Polygon:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<PointInfo>, // must specify at least 3 points  
<PointInfo>...)
```

```
<TextInfo>:
```

```
"Text:=", Array(<real>, // x position  
<real>, // y position  
<real>, // angle, in radians  
<Justification>,  
<bool>, // is plotter font  
<string>, // font name  
<int>, // color  
<string>) // text string
```

```
<Justification>:
```

```
<int> // 0 : left top
```

```
// 1 : left base  
// 2 : left bottom  
// 3 : center top  
// 4 : center base  
// 5 : center bottom  
// 6 : right top  
// 7 : right base  
// 8 : right bottom
```

```
<ImageInfo>:  
"Image:=", Array(<RectInfo>,  
<ImageData>,  
<bool>) // is mirrored
```

```
<ImageData>:  
<string>, // file path  
<int>, // 0 : use the file path and link to it  
// 1 : ignore file path and use next parameter  
<string> // text data, only present if preceding int is 1
```

```
<ListOfComponentNames>:  
<string>,<string> ...  
// The list may be empty. When not empty, each string that is listed is a component  
// that references the symbol to be edited. Prior to editing, a clone of the symbol is  
// made, and the components that are listed are modified so that they now refer to  
// the clone.
```

*VB Example:*

```
Dim nam  
  
oSymbolManager>EditWithComps _  
("Nexxim Circuit Elements\ Distributed\ Distributed:bendo", _  
Array("NAME:bendo new name", _  
"ModTime:=", 1152722165, _  
"Library:=", "Nexxim Circuit Elements\ Distributed\ Distributed", _  
"LibLocation:=", "SysLibrary", _  
Array("NAME:PinDef", _  
"Pin:=", Array( "n1", _  
-0.00254, _  
0.00254, _  
0, _  
"N", _
```

```
0, _
0, _
false, _
0, _
true, _
"", _
false, _
false)), _
Array("NAME:PinDef", _
"Pin:=", Array( "n2", _
0.00254, _
-0.00254, _
1.5708, _
"N", _
0, _
0, _
false, _
0, _
true, _
"", _
```

```
false, _  
false)), _  
Array("NAME:Graphics", _  
"Polygon:=", Array( 0, 0, 12566272, 0, 0.00381, 0, .00127, 0.00127, _  
0.00127, 0.00127, 0, 0.00381, 0, 0.00381, _  
0.002032, 0.00127, 0.00381), _  
"Line:=", Array(0, 1, 12566272, -0.00254, 0.00254, 0, .00254), _  
"Line:=", Array(0, 1, 12566272, 0.00254, -0.00254, .00254, 0)), _  
Array("NAME:PropDisplayMap", _  
"W:=", Array(3, _  
5, _  
0, _  
"Text:=", Array( 2.1684E-019, _  
0.00504119, _  
0, _  
1, _  
5, _  
false, _  
"Arial", _  
0, _  
"W=***") ) ) ), _
```

```
Array("MY_COMP")
```

## Export [symbol manager]

*Use:* Exports symbol(s) to a library

*Command:* Tools > Edit Configured Libraries > Symbols > Export to Library

*Syntax:* Export Array("NAME:<LibraryName>","

    <SymbolName>,

    <SymbolName>...),

    <LibraryLocation>

*Return Value:* None

*Parameters:* <LibraryName>:

    <string> // name of the library

    <SymbolName>:

        <string> // composite name of symbol to export

    <LibraryLocation>:

        <string> // location of the library in <LibraryName>

        // One of "Project", "PersonalLib", or "UserLib"

*VB Example:*

```
oSymbolManager.Export _  
Array("NAME NEXXIM Circuit Elements\ Distributed\ Distributed:bendo:mylib", _ "mySymbol"), "Per-  
sonalLib"
```

## GetData [symbol manager]

*Use:* Gets data that describes the definition.

*Command:* None

*Syntax:* GetData(<DefinitionName>)

*Return Value:* <DefinitionData> This is an array of data for the definition.

*Parameters:* <DefinitionName>:

<string> // composite name of the definition to edit

*VB Example:*

```
Dim symbolData  
  
symbolData = oSymbolManager.GetData("NEXXIM Circuit Elements\ HSPICE:nexx_bjt_npn")
```

### Note:

GetData allows the user to access definition information, make modifications, and then use the Edit or EditWithComps script commands to save the modified definition. Accordingly, for each type of definition, the array data returned to GetData should be the same array information that is supplied to the [Edit](#) or [EditWithComps](#) commands.

## GetNames [symbol manager]

*Use:* Returns the names of the symbols (used and unused) in a design. The following script command, **IsUsed**, can then be used to separate used and unused symbols.

*Command:* None

*Syntax:* GetNames()

*Return Value:* An array of strings

*Parameters:* None

*VB Example:*

```
Dim symbolNames  
symbolNames = oSymbolManager.GetNames()
```

### **IsUsed [symbol manager]**

*Use:* Used to determine if a symbol is used in the design.

*Command:* None

*Syntax:* IsUsed(<SymbolName>)

*Return Value:* <Boolean> // true if the specified symbol is used in the design

*Parameters:* <SymbolName>:

<string>

*VB Example:* Dim isUsed

```
isUsed = oSymbolManager.IsUsed("MySymbol")
```

### **Remove [symbol manager]**

*Use:* Removes a symbol from a library

*Command:* Tools > Edit Configured Libraries > Symbols > Remove Symbol

*Syntax:* Remove <SymbolName>,

```
<IsProjectSymbol>,
<LibraryName>,
<LibraryLocation>
```

*Return Value:* None

*Parameters:* <SymbolName>:

```
<string> // composite name of the symbol to remove
```

**<IsProjectSymbol>:**

```
<bool>
```

**<LibraryName>:**

```
<string> // name of the library
```

**<LibraryLocation>:**

```
<string> // location of the library in <LibraryName>
```

```
// One of "Project", "PersonalLib", or "UserLib"
```

*VB Example:*

```
oSymbolManager.Remove "Nexxim Circuit Elements\ Distributed\ Distributed:bendo", true, "Project"
```

## RemoveUnused [symbol manager]

*Use:* Removes symbols that are not used in the design.

*Command:* Project->Remove Unused Definitions is similar but operates slightly different and does not record script commands.

*Syntax:* RemoveUnused()

*Return Value:* <bool> True if one or more symbols are removed.

*Parameters:* None

*VB Example:*

```
Dim removedDefs  
removedDefs = oSymbolManager.RemoveUnused()
```

**Note:**

The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other symbols in dependent definitions to be rendered unusable.

Also, the symbol and footprint of an unused component are not unusable until after the component itself is removed using the Component Manager Remove script.

## Add [footprint manager]

*Use:* Add a footprint

*Command:* Tools > Edit Configured Libraries > Footprints > Add Footprint

*Syntax:* Add Array("NAME:<FootprintName>,

```
"ModTime:=", <ModifiedOnInfo>,
```

```
"Library:=", "",
```

```
"LibLocation:=", "Project",
"OkayToMirror:=", <bool>,
"DefUnits:=", <UnitType>,
Array(NAME:Lyr$",
"Layer:=", <LayerArray>,
"Layer:=", <LayerArray>...,
"SLayer:=", <StackupLayerArray>,
"SLayer:=", <StackupLayerArray>...),
"ActLyr:=", <string>, // name of active layer
"Tol:=", <ToleranceArray> // optional
<PrimitivesInfo>, // optional
<PinsInfo>, // optional
<ViasInfo>, // optional
<EdgeportsInfo>, // optional
<ComponentPropertyInfo>,
<ScriptInfo>) // optional, specified for scripted footprints
```

*Return Value:* <string>

```
// composite name of the footprint.
// If the name requested conflicts with the name of an existing
// footprint, the requested name is altered to be unique.
```

```
// The name returned reflects any change made to be unique.
```

*Parameters:* <FootprintName>:

```
<string> // simple name of footprint to create
```

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<UnitType>:

```
<string> // default length units to use if units are not specified in other  
// parameters
```

<LayerArray>:

```
Array("N:=", <string>, // layer name  
"ID:=", <int> ,  
"T:=", <LayerTypeInfo>, // layer type  
"TB:=", <TopBottomInfo>,  
"Col:=", <int>, // optional - color  
"Pat:=", <int>, // optional - fill pattern  
"Vis:=", <bool>, // optional - are objects on layer visible
```

```
"Sel:=", <bool>, // optional - are objects on layer selectable
"L:=", <bool>) // optional
// are objects on layer locked (can't be edited)

<LayerTypeInfo>:
<string> // one of: signal, dielectric, metalized signal, assembly, silkscreen, soldermask, solderpaste, glue, or user

<TopBottomInfo>:
<string> // one of: top, neither, bottom, or template

<StackupLayerArray>:
Array(<LayerArray>,
"Elev:=", <ElevationInfo>,
"SubL:=", Array("Th:=", <Dimension>,
"LElev:=", <Dimension>,
"R:=", <Dimension>,
"M:=", <MaterialInfo>))

<ElevationInfo>:
<string> // "top" - snap to top
// "mid" - snap to middle
// "bot" - snap to bottom
```

```
// "edit" - manual edit
// "none"

<Dimension>:
<string> // real number, may include units

<MaterialInfo>:
<string> // name of the layer material

<ToleranceArray>:
Array(<real>, // distance tolerance
      <real>, // angle tolerance (radians)
      <real>) // dimensionless tolerance

<PrimitivesInfo>:
Array("NAME:Prims",
      // one or more of the following
      <RectInfo>,
      <CircleInfo>,
      <ArcInfo>,
```

```
<LineInfo>,
<PolygonInfo>,
<TextInfo>,
<ImageInfo>

<RectInfo>:
"Rect:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // angle, in radians
<real>, // x position of center
<real>, // y position of center
<real>, // width
<real>) // height

<CircleInfo>:
"Circle:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // x position of center
<real>, // y position of center
```

```
<real>) // radius
```

```
<ArcInfo>:
```

```
"Arc:=", Array(<real>, // line width  
<int>, // line pattern  
<int>, // color  
<real>, // x position of center  
<real>, // y position of center  
<real>, // radius  
<real>, // start angle, in radians  
<end>) // end angle, in radians
```

```
<LineInfo>:
```

```
"Line:=", Array(<real>, // line width  
<int>, // line pattern  
<int>, // color  
<PointInfo>, // must specify at least 2 points  
<PointInfo>...)  
<PointInfo>:  
<real>, // x position
```

```
<real> // y position
```

```
<PolygonInfo>:
```

```
"Polygon:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<PointInfo>, // must specify at least 3 points  
<PointInfo>...)
```

```
<TextInfo>:
```

```
"Text:=", Array(<real>, // x position
```

```
<real>, // y position  
<real>, // angle, in radians  
<Justification>,  
<bool>, // is plotter font  
<string>, // font name  
<int>, // color  
<string>) // text string  
<Justification>: <int>  
// 0 : left top  
// 1 : left base
```

```
// 2 : left bottom  
// 3 : center top  
// 4 : center base  
// 5 : center bottom  
// 6 : right top  
// 7 : right base  
// 8 : right bottom
```

```
<ImageInfo>:  
"Image:=", Array(<RectInfo>,  
<ImageData>,  
<bool>) // is mirrored
```

```
<ImageData>:  
<string>, // file path  
<int>, // 0 : use the file path and link to it  
// 1 : ignore file path and use next parameter  
<string> // text data, only present if preceding int is 1
```

```
<PinsInfo>:
```

```
Array("NAME:_pins",
      "P:=", <PinArray>,
      "P:=", <PinArray>, ...)

<PinArray>:
  Array("Port:=", Array("Id:=", <int>,
                        "Clr:=", <real>, // optional - clearance
                        "N:=", <string>), // pin name
        "Pos:=", Array("x:=", <Location>, // padstack (x,y) position
                      "y:=", <Location>),
        "VRt:=", <Angle>, // optional - rotation
        "HD:=", <Size>, // optional - hole diameter
        <PadstackImplementationInfo>)

<Location>:
  <string> specifying real number and units (may use variables)

<Angle>:
  <string> specifying angle with a real number and units

<Size>:
```

<string> specifying size with a real number and units

<PadstackImplementationInfo>:

If another with the same implementation has already been specified:

"Ref:=", <int> // id of the other

If not:

"Ref:=", <string>, // name

"Frm:=", <int>, // id of highest layer

"To:=", <int>, // id of lowest layer

<LayerPlacementInfo>,

"Man:=", <int>, // optional, 1 if manually placed, 0 if not (default)

"Use:=", Array(<PadUseInfo>, <PadUseInfo>...) // array may be empty

<LayerPlacementInfo>:

"Lyr:=", Array("Mrg:=", <int>, // optional,

// 1 if all layers have been merged (default)

// 0 if layer are not merged

"Flp:=", <int>, // optional,

// 1 if placed bottom up

// 0 if not (default)

```
"Map:=", <ParentToLocalLayerInfo>

<ParentToLocalLayerInfo>:
    Array("U:=", <DirectionOfUniqueness>,
          "F:=", Array(<int>, <int>, ...), // forward mapping
          // -1 is not mapped
          "B:=", Array(<int>, <int>, ...)) // backward mapping
          // -1 is not mapped

<PadUseInfo>:
    "Pad:=", Array("Lid:=", <int>, // layer id
                   "T:=", <string>, // type : "connected", "thermal",
                   // "no_pad", "not_connected",
                   // or "not_connected_thermal"
                   "Man:=", <int>) // optional, 1 if manually placed
                   // 0 if not (default)

<DirectionOfUniqueness>:
    <string> // one of "forward", "backward", or "two ways"

<ViasInfo>:
```

```
Array("NAME:Vias", <VialInfo>, <VialInfo>...)
```

```
<VialInfo>:
```

```
"V:=", Array("Id:=", <int>,
"N:=", <string>, // name
"Pos:=", Array("x:=", <Location>, // via (x,y) position
"y:=", <Location>),
"VRt:=", <Angle>, // optional - rotation
<ImplementationInfo>
```

```
<EdgeportsInfo>:
```

```
Array("NAME:EPorts", <EdgePortArray>, <EdgePortArray>...)
```

```
<EdgePortArray>:
```

```
Array("NAME:EP",
"LP:=", Array("Id:=", <int>, // port id
"N:=", <string>), // port name
"Eo:=", Array(<edge description>, <edge description>,...))
```

```
<edge description> for primitive edges
```

"et:=", "pe", "pr:=", <id>, "ei:=", <edge#>

<id>: integer that is the primitive id

<edge#>: integer that is the edge number on the primitive

<edge description> for via edges

"et:=", "pse", "layer:=", <layer id>, "se:=", <via id>,  
"sx:=", <start X location>, "sy:=", <start Y location>, "ex:=", <end X location>,  
"ey:=", <end Y location>, "h:=", <arc height>, "rad:=", <radians>

<via id>: an integer that is the id of the via to use

<layer id>: an integer that is the id of the layer of the pad of the via to use

<start X location>

<start Y Location>:

doubles that are the X, Y location of the start point of the edge arc <end X location>

<end Y Location>:

doubles that are the X, Y location of the end point of the edge arc

<arc height>: double giving the height of the edge arc (0 for a straight edge)

<radians>: double giving the arc size in radians (0 for a straight edge)

<ComponentPropertyInfo>:

```
Array("NAME:CProps",
"VariableProp:=", <VariableInfo>,
"VariableProp:=", <VariableInfo>,
...)
```

<VariableInfo>:

```
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>) // value: number, variable, or expression
```

<ScriptInfo>:

```
Array("NAME:script",
"language:=", <string>, // one of "javascript" or "vbscript"
```

```
"UsesScript:=", true,  
"script:=", <string>) // contents of script
```

*VB Example:*

```
oFootprintManager.Add (Array("NAME:BCL", _  
"ModTime:=", 1023388445, _  
"Library:=", "", _  
"LibLocation:=", "Project", _  
"OkayToMirror:=", false, _  
"DefUnits:=", "mm", _  
Array("NAME:LyrS", _  
"Layer:=", Array("N:=", "Measures", _  
"ID:=", 8, _  
"T:=", "measures", _  
"Col:=", 4144959, _  
"Pat:=", 1), _  
"Layer:=", Array("N:=", "Rats", _  
"ID:=", 1, _  
"T:=", "rat", _  
"Col:=", 16711680, _  
"Pat:=", 1), _
```

```
"Layer:=", Array("N:=", "Errors", _  
"ID:=", 2, _  
"T:=", "error", _  
"Col:=", 255, _  
"Pat:=", 1, _  
"L:=", true), _  
"Layer:=", Array("N:=", "Symbols", _  
"ID:=", 3, _  
"T:=", "symbol", _  
"Col:=", 8323199, _  
"Pat:=", 4), _  
"Layer:=", Array("N:=", "Assembly", _  
"ID:=", 4, _  
"T:=", "assembly", _  
"TB:=", "top", _  
"Col:=", 16711680, _  
"Pat:=", 3), _  
"Layer:=", Array("N:=", "Silkscreen", _  
"ID:=", 5, _  
"T:=", "silkscreen", _
```

```
TB:="", "top", _
"Col:=", 8454143, _
"Pat:=", 6), _
SLayer:="", Array("Layer:=", Array("N:=", "Cover", _
"ID:=", 12, _
"T:=", "metalizedsignal", _
"Col:=", 32639, _
Pat:=", 7), _
"Elev:=", "mid", _
"SubL:=", Array("Th:=", "0mm", _
"LElev:=", "118.110236220472mil", _
"R:=", "0mm", _
"Mat:=", "") ), _
"SLayer:=", Array("Layer:=", Array("N:=", "Dielec3", _
"ID:=", 11, _
"T:=", "dielectric", _
"Col:=", 8323199, _
"Pat:=", 6), _
"Elev:=", "none", _
"SubL:=", Array("Th:=", "1mm", _
"LElev:=", "78.740157480315mil", _
```

```
"R:=", "0mm", _
"Mat:="", ""))
"SLayer:=", Array("Layer:=", Array("N:=", "Trace2", _
"ID:=", 10, _
"T:=", "signal", _
"Col:=", 8355584, _
"Pat:=", 5), _
"Elev:=", "mid", _
"SubL:=", Array("Th:=", "0mm", _
"LElev:=", "78.740157480315mil", _
"R:=", "0mm", _
"Mat:="", ""))
"SLayer:=", Array("Layer:=", Array("N:=", "Dielec2", _
"ID:=", 6, _
"T:=", "dielectric", _
"Col:=", 8323072, _
"Pat:=", 4), _
"Elev:=", "none", _
"SubL:=", Array("Th:=", "1mm", _
"LElev:=", "39.3700787401575mil", _
```

```
"R:=", "0mm", _
"Mat:="", ""))
"SLayer:=", Array("Layer:=", Array("N:=", "Trace1", _
"ID:=", 0, _
"T:=", "signal", _
"Col:=", 32512, _
"Pat:=", 3), _
"Elev:=", "mid", _
"SubL:=", Array("Th:=", "0mm", _
"LElev:=", "39.3700787401575mil", _
"R:=", "0mm", _
"Mat:="", ""))
"SLayer:=", Array("Layer:=", Array("N:=", "Dielec1", _
"ID:=", 7, _
"T:=", "dielectric", _
"Col:=", 127, _
"Pat:=", 7), _
"Elev:=", "none", _
"SubL:=", Array("Th:=", "1mm", _
"LElev:=", "0mil", _
"R:=", "0mil", _
```

```
"Mat:="", "") ), _  
"SLayer:=", Array("Layer:=", Array("N:=", "Ground", _  
"ID:=", 9, _  
"T:=", "metalizedsignal", _  
"Col:=", 4144959, _  
"Pat:=", 6), _  
"Elev:=", "mid", _  
"SubL:=", Array("Th:=", "0mil", _  
"LElev:=", "0mil", _  
"R:=", "0mil", _  
"Mat:="", "") ) ), _  
"ActLyr:=", "Trace2", _  
Array("NAME:Prims",  
"rect:=", Array("Id:=", 10003, _  
"Lyr:=", 10, _  
"N:=", "rect101", _  
"x:=", "0mm", _  
"y:=", "0mm", _  
"w:=", "P", _  
"h:=", "W", _
```

```
"Vds:=", Array()), _  
"rect:=", Array("Id:=", 10103, _  
"Lyr:=", 0, _  
"N:=", "rect103", _  
"x:=", "0mm", _  
"y:=", "0mm", _  
"w:=", "P", _  
"h:=", "W", _  
"Vds:=", Array()))), _  
Array("NAME:Pins",  
"P:=", Array("Port:=", Array("Id:=", 3, "N:=", "n1"), _  
"Pos:=", Array("x:=", "-P/2", "y:=", "0mm"), _  
"VRT:=", "180deg", _  
"Ref:=", "NoPad SMT East", _  
"Frm:=", 12, _  
"To:=", 9, _  
"Lyr:=", Array("Map:=", Array("U:=", "forward", _  
"F:=", Array(-1, -1, -1, -1, _  
-1, -1, -1, -1, _  
-1, -1, 0, -1, -1), _  
"B:=", Array(10))), _
```

```
"Use:=", Array(), _  
"P:=", Array("Port:=", Array("Id:=", 4, "N:=", "n4"), _  
"Pos:=", Array("x:=", "P/2", "y:=", "0mm"), _  
"HD:=", "1mm", _  
"Ref:=", 3), _  
"P:=", Array("Port:=", Array("Id:=", 5, "N:=", "n2"), _  
"Pos:=", Array("x:=", "-P/2", "y:=", "0mm"), _  
"VRT:=", "180deg", _  
"Ref:=", "NoPad SMT East", _  
"Frm:=", 12, _  
"To:=", 9, _  
"Lyr:=", Array("Map:=", Array("U:=", "forward", _  
"F:=", Array(0, -1, -1, -1, -1, _  
-1, -1, -1, -1, _  
-1, -1, -1, -1), _  
"B:=", Array(0))), _  
"Use:=", Array()), _  
"P:=", Array("Port:=", Array("Id:=", 6, "N:=", "n3"), _  
"Pos:=", Array("x:=", "P/2", "y:=", "0mm"), _  
"HD:=", "1mm", _
```

```
"Ref:=", 5)), _  
Array("NAME:Nets"), _  
Array("NAME:CProps",  
"VariableProp:=", Array("W", _  
"UD", _  
"", _  
"1.5mm"), _  
"VariableProp:=", Array("P", _  
"UD", _  
"", _  
"10mm"))))
```

## Edit [footprint manager]

Deprecated command — please use [EditWithComps](#).

## EditWithComps [footprint manager]

*Use:* Edit an existing footprint.

*Command:* None

*Syntax:* EditWithComps <FootprintName>,

```
    Array("NAME:<NewFootprintName>,  
    "ModTime:=", <ModifiedOnInfo>,  
    "Library:=", "",
```

```
"LibLocation:=", "Project",
"OkayToMirror:=", <bool>,
"DefUnits:=", <UnitType>,
Array(NAME:Lyr$",
"Layer:=", <LayerArray>,
"Layer:=", <LayerArray>...,
"SLayer:=", <StackupLayerArray>,
"SLayer:=", <StackupLayerArray>...),
"ActLyr:=", <string>, // name of active layer
"Tol:=", <ToleranceArray> // optional
<PrimitivesInfo>, // optional
<PinsInfo>, // optional
<ViasInfo>, // optional
<EdgeportsInfo>, // optional
<ComponentPropertyInfo>,
<ScriptInfo>, // optional, specified for scripted footprints
Array(<ListofComponentNames>) // Component names
```

*Return Value:* <string>

```
// composite name of the footprint.
// If the name requested conflicts with the name of an existing
```

```
// footprint, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

*Parameters:* <FootprintName>:  
    <string> // [composite name](#) of the footprint being edited

```
<NewFootprintName>:  
    <string> // new simple name for the footprint
```

```
<ModifiedOnInfo>:  
    An integer that corresponds to the number of seconds that have elapsed  
    since 00:00 hours, Jan 1, 1970 UTC from the system clock.
```

```
<UnitType>:  
    <string> // default length units to use if units are not specified in other  
    // parameters
```

```
<LayerArray>:  
    Array("N:=", <string>, // layer name  
        "ID:=", <int> ,  
        "T:=", <LayerTypeInfo>, // layer type
```

```
"TB:=", <TopBottomInfo>,  
"Col:=", <int>, // optional - color  
"Pat:=", <int>, // optional - fill pattern  
"Vis:=", <bool>, // optional - are objects on layer visible  
"Sel:=", <bool>, // optional - are objects on layer selectable  
"L:=", <bool>) // optional  
// are objects on layer locked (can't be edited)
```

**<LayerTypeInfo>:**

```
<string> // one of: signal, dielectric, metalized signal, assembly, silkscreen, soldermask, solderpaste, glue, or user
```

**<TopBottomInfo>:**

```
<string> // one of: top, neither, bottom, or template
```

**<StackupLayerArray>:**

```
Array(<LayerArray>,  
"Elev:=", <ElevationInfo>,  
"SubL:=", Array("Th:=", <Dimension>,  
"LElev:=", <Dimension>,  
"R:=", <Dimension>,
```

```
"M:=", <MaterialInfo>))
```

**<ElevationInfo>:**

```
<string> // "top" - snap to top  
// "mid" - snap to middle  
// "bot" - snap to bottom  
// "edit" - manual edit  
// "none"
```

**<Dimension>:**

```
<string> // real number, may include units
```

**<MaterialInfo>:**

```
<string> // name of the layer material
```

**<ToleranceArray>:**

```
Array(<real>, // distance tolerance  
<real>, // angle tolerance (radians)  
<real>) // dimensionless tolerance
```

**<PrimitivesInfo>:**

```
Array("NAME:Prims",
// one or more of the following
<RectInfo>,
<CircleInfo>,
<ArcInfo>,
<LineInfo>,
<PolygonInfo>,
<TextInfo>,
<ImageInfo>)
```

**<RectInfo>:**

```
"Rect:=", Array(<real>, // line width
<int>, // fill pattern
<int>, // color
<real>, // angle, in radians
<real>, // x position of center
<real>, // y position of center
<real>, // width
<real>) // height
```

**<CircleInfo>:**

```
"Circle:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<real>, // x position of center  
<real>, // y position of center  
< real>) // radius
```

**<ArcInfo>:**

```
"Arc:=", Array(<real>, // line width  
<int>, // line pattern  
<int>, // color  
<real>, // x position of center  
<real>, // y position of center  
< real>, // radius  
<real>, // start angle, in radians  
<end>) // end angle, in radians
```

**<LineInfo>:**

```
"Line:=", Array(<real>, // line width  
<int>, // line pattern
```

```
<int>, // color  
<PointInfo>, // must specify at least 2 points  
<PointInfo>...)  
<PointInfo>:  
<real>, // x position  
<real> // y position
```

```
<PolygonInfo>:  
"Polygon:=", Array(<real>, // line width  
<int>, // fill pattern  
<int>, // color  
<PointInfo>, // must specify at least 3 points  
<PointInfo>...)
```

```
<TextInfo>:  
"Text:=", Array(<real>, // x position  
<real>, // y position  
<real>, // angle, in radians  
<Justification>,  
<bool>, // is plotter font
```

```
<string>, // font name  
<int>, // color  
<string>) // text string  
<Justification>: <int>  
// 0 : left top  
// 1 : left base  
// 2 : left bottom  
// 3 : center top  
// 4 : center base  
// 5 : center bottom  
// 6 : right top  
// 7 : right base  
// 8 : right bottom
```

**<ImageInfo>:**

```
"Image:=", Array(<RectInfo>,  
<ImageData>,  
<bool>) // is mirrored
```

**<ImageData>:**

```
<string>, // file path
```

```
<int>, // 0 : use the file path and link to it  
// 1 : ignore file path and use next parameter  
<string> // text data, only present if preceding int is 1
```

**<PinsInfo>:**

```
Array("NAME:Pins",  
"P:=", <PinArray>,  
"P:=", <PinArray>,...)
```

**<PinArray>:**

```
Array("Port:=", Array("Id:=", <int>,  
"Clr:=", <real>, // optional - clearance  
"N:=", <string>), // pin name  
"Pos:=", Array("x:=", <Location>, // padstack (x,y) position  
"y:=", <Location>),  
"VRt:=", <Angle>, // optional - rotation  
"HD:=", <Size>, // optional - hole diameter  
<PadstackImplementationInfo>)
```

**<Location>:**

<string> specifying real number and units (may use variables)

**<Angle>:**

<string> specifying angle with a real number and units

**<Size>:**

<string> specifying size with a real number and units

**<PadstackImplementationInfo>:**

If another with the same implementation has already been specified:

"Ref:=", <int> // id of the other

If not:

"Ref:=", <string>, // name

"Frm:=", <int>, // id of highest layer

"To:=", <int>, // id of lowest layer

<LayerPlacementInfo>,

"Man:=", <int>, // optional, 1 if manually placed, 0 if not (default)

"Use:=", Array(<PadUseInfo>, <PadUseInfo>...) // array may be empty

**<LayerPlacementInfo>:**

"Lyr:=", Array("Mrg:=", <int>, // optional,

```
// 1 if all layers have been merged (default)
// 0 if layer are not merged
"Flp:=", <int>, // optional,
// 1 if placed bottom up
// 0 if not (default)
"Map:=", <ParentToLocalLayerInfo>
```

**<ParentToLocalLayerInfo>:**

```
Array("U:=", <DirectionOfUniqueness>,
"F:=", Array(<int>, <int>, ...), // forward mapping
// -1 is not mapped
"B:=", Array(<int>, <int>, ...)) // backward mapping
// -1 is not mapped
```

**<PadUseInfo>:**

```
"Pad:=", Array("Lid:=", <int>, // layer id
"T:=", <string>, // type : "connected", "thermal",
// "no_pad, "not_connected",
// or "not_connected_thermal"
"Man:=", <int>) // optional, 1 if manually placed
```

```
// 0 if not (default)
```

**<DirectionOfUniqueness>:**

```
<string> // one of "forward", "backward", or "two ways"
```

**<ViasInfo>:**

```
Array("NAME:Vias", <VialInfo>, <VialInfo>...)
```

**<VialInfo>:**

```
"V:=", Array("Id:=", <int>,
"N:=", <string>, // name
"Pos:=", Array("x:=", <Location>, // via (x,y) position
"y:=", <Location>),
"VRt:=", <Angle>, // optional - rotation
<ImplementationInfo>
```

**<EdgeportsInfo>:**

```
Array("NAME:EPorts", <EdgePortArray>, <EdgePortArray>...)
```

**<EdgePortArray>**

```
Array("NAME:EP",
```

```
"LP:=", Array("Id:=", <int>, // port id  
"N:=", <string>), // port name  
"Eo:=", Array(<edge description>, <edge description>, ...))
```

<edgedescription> for primitive edges

```
"et:=", "pe", "pr:=", <id>, "ei:=", <edge#>
```

<id>: integer that is the primitive id

<edge#>: integer that is the edge number on the primitive

<edge description> for via edges

```
"et:=", "pse", "sel:=", <"via">, "layer:=", <layer id>,  
"sx:=", <start X location>, "sy:=", <start Y location>, "ex:=", <end X location>, "ey:=", <end Y location>, "h:=", <arc height>,  
"rad:=", <radians>
```

<"via">: text that is the name of the via to use

<layer id>: an integer that is the id of the layer of the pad of the via to use

<start X location>, <start Y Location>:

doubles that are the X, Y location of the start point of the edge arc

<end X location>, <end Y Location>:

doubles that are the X, Y location of the end point of the edge arc

<arc height>: double giving the height of the edge arc (0 for a straight edge)

<radians>: double giving the arc size in radians (0 for a straight edge)

**<ComponentPropertyInfo>:**

```
Array("NAME:CProps",
"VariableProp:=", <VariableInfo>,
"VariableProp:=", <VariableInfo>,
...)
```

**<VariableInfo>:**

```
Array(<string>, // name
<FlagLetters>,
<string>, // description
"CB:=", <string>, // optional - script for call back
<string>) // value: number, variable, or expression
```

**<ScriptInfo>:**

```
Array("NAME:script",
"language:=", <string>, // one of "javascript" or "vbscript"
"UsesScript:=", true,
"script:=", <string>) // contents of script
```

**<ListOfComponentNames>:**

```
<string>,<string> ...
// The list may be empty. When not empty, each string that is listed is a component
// that references the footprint to be edited. Prior to editing, a clone of the footprint is
// made, and the components that are listed are modified so that they now refer to
// the clone.
```

*VB Example:*

```
Dim nam
oFootprintManager>EditWithComps _
("Nexxim Circuit Elements\ Distributed\ Distributed:bendo", _
Array("NAME:bendo new name", _
"ModTime:=", 1152722165, _
"Library:=", "Nexxim Circuit Elements\ Distributed\ Distributed", _
```

```
"LibLocation:=", "SysLibrary", _  
Array("NAME:PinDef", _  
"Pin:=", Array( "n1", _  
-0.00254, _  
0.00254, _  
0, _  
"N", _  
0, _  
0, _  
false, _  
0, _  
true, _  
"", _  
false, _  
false)), _  
Array("NAME:PinDef", _  
"Pin:=", Array( "n2", _  
0.00254, _  
-0.00254, _  
1.5708, _  
"N", _
```

```
0, _
0, _
false, _
0, _
true, _
"", _
false, _
false)), _
Array("NAME:Graphics", _
"Polygon:=", Array( 0, 0, 12566272, 0, 0.00381, 0, .00127, 0.00127, _
0.00127, 0.00127, 0, 0.00381, 0, 0.00381, _
0.002032, 0.00127, 0.00381), _
"Line:=", Array(0, 1, 12566272, -0.00254, 0.00254, 0, .00254), _
"Line:=", Array(0, 1, 12566272, 0.00254, -0.00254, .00254, 0)), _
Array("NAME:PropDisplayMap", _
"W:=", Array(3, _
5, _
0, _
"Text:=", Array( 2.1684E-019, _
0.00504119, _
```

```
0, _  
1, _  
5, _  
false, _  
"Arial", _  
0, _  
"W=***") )) ), _  
Array("MY_COMP")
```

## Export [footprint manager]

*Use:* Export a footprint to a library

*Command:* Tools > Edit Configured Libraries > Footprints > Export to Library

*Syntax:* Export Array("NAME:<LibraryName>,"

```
<FootprintName>,  
<FootprintName>...),  
<LibraryLocation>
```

*Return Value:* None

*Parameters:* <LibraryName>:

```
<string> // name of the library
```

```
<FootprintName>:
```

```
<string> // composite name of footprint to export
```

```
<LibraryLocation>:  
  <string> // location of the library in <LibraryName>  
  // One of "Project", "PersonalLib", or "UserLib"
```

*VB Example:*

```
oFootprintManager.Export Array("NAME:mylib", "Distributed Footprints:BPAD"), "PersonalLib"
```

## **GetData [footprint manager]**

### **FootprintGetData**

*Use:* Gets data that describes the definition.

*Command:* None

*Syntax:* GetData(<DefinitionName>)

*Return Value:* <DefinitionData> This is an array of data for the definition.

*Parameters:* <DefinitionName>:

```
  <string> // composite name of the definition to edit
```

*VB Example:*

```
Dim footprintData
```

```
footprintData = oFootprintManager.GetData("Nexxim Circuit Elements\ Distributed\Nexxim_Foot-  
prints:MCPL13_Nexx")
```

**Note:**

GetData allows the user to access definition information, make modifications, and then use the Edit or EditWithComps script commands to save the modified definition. Accordingly, for each type of definition, the array data returned to GetData should be the same array information that is supplied to the [Edit](#) or [EditWithComps](#) commands.

## **GetNames [footprint manager]**

*Use:* Returns the names of the footprints (used and unused) in a design. The following script command, **IsUsed**, can then be used to separate used and unused footprints.

*Command:* None

*Syntax:* GetNames()

*Return Value:* An array of strings

*Parameters:* None

*VB Example:*

```
Dim footprintNames  
footprintNames = oFootprintManager.GetNames()
```

## **IsUsed [footprint manager]**

*Use:* Used to determine if a footprint is used in the design.

*Command:* None

*Syntax:* IsUsed(<FootprintName>)

*Return Value:* <Boolean> // true if the specified footprint is used in the design

*Parameters:* <FootprintName>:

<string>

*VB Example:*

```
Dim isUsed  
isUsed = oFootprintManager.IsUsed("MyFootprint")
```

## **Remove [footprint manager]**

*Use:* Removes a footprint from a library

*Command:* Tools > Edit Configured Libraries > Footprints > Remove Footprint

*Syntax:* Remove <FootprintName>,

```
<IsProjectFootprint>,  
<LibraryName>,  
<LibraryLocation>
```

*Return Value:* None

*Parameters:* <FootprintName>:

```
<string> // composite name of the footprint to remove
```

```
<IsProjectFootprint>:
```

```
<bool>
```

```
<LibraryName>:
```

```
<string> // name of the library
```

```
<LibraryLocation>:  
    <string> // location of the library in <LibraryName>  
    // One of "Project", "PersonalLib", or "UserLib"
```

*VB Example:*

```
oFootprintManager.Remove "BPAD", true, "Distributed Footprints", "Project"  
oFootprintManager.Remove "BPAD", false, "MyLib", "PersonalLib"
```

## **RemoveUnused [footprint manager]**

*Use:* Removes footprints that are not used in the design.

*Command:* **Project->Remove Unused Definitions** is similar but operates slightly different and does not record script commands.

*Syntax:* RemoveUnused()

*Return Value:* <bool> True if one or more footprints are removed.

*Parameters:* None

*VB Example:*

```
Dim removedDefs  
removedDefs = oFootprintManager.RemoveUnused()
```

**Note:**

The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other footprints in dependent definitions to be rendered unusable.

Also, the symbol and footprint of an unused component are not unusable until after the component itself is removed using the Component ManagerRemove script.

## Add [padstack manager]

*Use:* Add a padstack

*Command:* Tools > Edit Configured Libraries > Padstacks > Add Padstack

*Syntax:* Add Array("NAME:<PadstackName>","

```
"ModTime:=", <ModifiedOnInfo>,  
"Library:=", "", // name of the library  
"LibLocation:=", "Project", // location of the named library  
Array("NAME:psd",  
"nam:=", <PadstackName>,  
"lib:=", "", // name of the library  
"mat:=", "", // hole plating material  
"plt:=", "0", // percent of hole's radius filled by plating  
Array("NAME:pds",
```

```
<LayerGeometryArray>,
<LayerGeometryArray....),
"hlc:=", <PadInfo>
"hrg:=", <HoleRange>,
"sbsh:=", <SolderballShape>,
"sbpl:=", <SolderballPlacement>,
"sbr:=", <string>, // solderball diameter, real with units
"sb2:=", <string>, // solderball mid diameter, real with units
"sbm:=", <string>), // name of solderball material
"ppl:=", <PadPortLayerArray>)
```

*Return Value:* [simple name](#) of the added padstack

```
// If the name requested conflicts with the name of an existing
// padstack, the requested name is altered to be unique.
// The name returned reflects any change made to be unique.
```

*Parameters:* <PadstackName>:

<string> // [simple name](#) of padstack to create

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed  
since 00:00 hours, Jan 1, 1970 UTC from the system clock.

```
<LayerGeometryArray>
  Array("Name:lgm",
    "lay:=", <string>, // definition layer name
    "id:=", <int>, // definition layer id
    "pad:=", <PadInfo>, // pad
    "ant:=", <PadInfo>, // antipad
    "thm:=", <PadInfo>, // themal pad
    "X:=", <string>, // pad x connection, real with units
    "Y:=", <string>, // pad y connection, real with units
    "dir:=", <DirectionString>) // pad connection direction

<PadInfo>
  Array("shp:=", <PadShape>,
    "Szs:=", <DimensionArray>,
    "X:=", <string>, // x offset, real with units
    "Y:=", <string>, // y offset, real with units
    "R:=", <string>) // rotation, real with units

<PadShape>
```

<string> one of these choices

"No" // no pad

"Cir" // Circle

"Sq" // Square

"Rct" // Rectangle

"Ov" // Oval

"Blt" // Bullet

"Ply" // Polygons

"R45" // Round 45 thermal

"R90" // Round 90 thermal

"S45" // Square 45 thermal

"S90" // Square 90 thermal

<DimensionArray>:

Array(<string>, ...) // each string is a real with units for one of the dimensions of the shape

<DirectionString>:

<string> one of these choices

"No" // no direction

"Any" // any direction

"0" // 0 degrees

"45" // 45 degrees

"90" // 90 degrees

"135" // 135 degrees

"180" // 180 degrees

"225" // 225 degrees

"270" // 270 degrees

"315" // 315 degrees

<HoleRange>:

<string> one of these choices

"Thr" // through all layout layers

"Beg" // from upper pad layer to lowest layout layer

"End" // from upper layout layer to lowest pad layer

"UTL" // from upper pad layer to lowest pad layer

<SolderballShape>:

<string> one of these choices

"None" // no solderball

"Cyl" // cylinder solderball

"Sph" // spheroid solderball

<SolderballPlacement>:

<string> one of these choices

"abv" // above padstack

"blw" // below padstack

<PadPortLayerArray>:

Array( <int>, <int>,....) where each int is a layer id

*VB Example:*

```

oPadstackManager.Add Array("NAME:Circle - through3", "ModTime:=", 1235765635, "Library:=", _
"", "LibLocation:=", "Project", Array("NAME:psd", "nam:=", "Circle - through3", "lib:=", _
"", "mat:="", "", "plt:=", "0", Array("NAME:pds", Array("NAME:lgm", "lay:=", "Top Signal",
"id:=", _
0, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2.5mm"), "X:=", "0mm", "Y:=", _
"0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array("3.5mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",
"lay:=", _
"SignalA", "id:=", 1, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _
"3mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "No", "Szs:=", Array(),
"X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",
"lay:=", _
"SignalB", "id:=", 2, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2mm"), "X:=", _

```

```
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _  
"3mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "thm:=", Array("shp:=", "No", "Szs:=", Array()  
( ), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",  
"lay:=", _  
"Ground", "id:=", 3, "pad:=", Array("shp:=", "No", "Szs:=", Array()), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "No", "Szs:=", Array()), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "R90", "Szs:=", Array( _  
"3mm", "0.75mm", "1mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", _  
"0mm", "dir:=", "Any"), Array("NAME:lgm", "lay:=", "Bottom signal", "id:=", 5, "pad:=", Array  
("shp:=", _  
"Cir", "Szs:=", Array("1mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=",  
_  
"Cir", "Szs:=", Array("2mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "thm:=", Array("shp:=",  
_  
"No", "Szs:=", Array(), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", _  
"0mm", "dir:=", "Any)), "hle:=", Array("shp:=", "Cir", "Szs:=", Array("1.5mm"), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0deg"), "hRg:=", "End", "sbsh:=", "Sph", "sbpl:=", _  
"abv", "sbr:=", "750um", "sb2:=", "1200um", "1200um", "sbn:=", "solder"), "ppl:=", Array( _  
0, 1, 2, 3, 5))
```

## Edit [padstack manager]

*Use:* Edit an existing padstack.

*Command:* Tools > Edit Configured Libraries > Padstacks > Edit Padstack

*Syntax:* Edit <PadstackName>,

```
Array("NAME:<NewPadstackName>",
"ModTime:=", <ModifiedOnInfo>,
"Library:=", "", // name of the library
"LibLocation:=", "Project", // location of the named library
Array("NAME:psd",
"nam:=", <PadstackName>,
"lib:=", "", // name of the library
"mat:=", "", // hole plating material
"plt:=", "0", // percent of hole's radius filled by plating
Array("NAME:pds",
<LayerGeometryArray>,
<LayerGeometryArray....>),
"hole:=", <PadInfo>
"holeRg:=", <HoleRange>,
"sbsh:=", <SolderballShape>,
"sbpl:=", <SolderballPlacement>,
"sbr:=", <string>, // solderball diameter, real with units
"sb2:=", <string>, // solderball mid diameter, real with units
"sbm:=", <string>), // name of solderball material
```

```
"ppl:=", <PadPortLayerArray>)
```

*Return Value:* <string> // [composite name](#) of the padstack

```
// If the name requested conflicts with the name of an existing  
// padstack, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

*Parameters:* <PadstackName>:

```
<string> // composite name of padstack to edit
```

<NewPadstackName>:

```
<string> // new simple name for padstack
```

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed  
since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<LayerGeometryArray>:

```
Array("Name:lgm",
```

```
"lay:=", <string>, // definition layer name
```

```
"id:=", <int>, // definition layer id
```

```
"pad:=", <PadInfo>, // pad
"ant:=", <PadInfo>, // antipad


<PadInfo>:


```

```
Array("shp:=", <PadShape>,
"Szs:=", <DimensionArray>,
"X:=", <string>, // x offset, real with units
"Y:=", <string>, // y offset, real with units
"R:=", <string>) // rotation, real with units
```

<PadShape>:

```
<string> one of these choices
"No" // no pad
"Cir" // Circle
" Sq" // Square
"Rct" // Rectangle
"Ov" // Oval
```

```
"Blt" // Bullet  
"Ply" // Polygons  
"R45" // Round 45 thermal  
"R90" // Round 90 thermal  
"S45" // Square 45 thermal  
"S90" // Square 90 thermal
```

```
<DimensionArray>:  
Array(<string>, ...) // each string is a real with units for one of the  
// dimensions of the shape
```

```
<DirectionString>:  
<string> one of these choices  
"No" // no direction  
"Any" // any direction  
"0" // 0 degrees  
"45" // 45 degrees  
"90" // 90 degrees  
"135" // 135 degrees  
"180" // 180 degrees
```

```
"225" // 225 degrees
"270" // 270 degrees
"315" // 315 degrees

<HoleRange>:
<string> one of these choices
"Thr" // through all layout layers
"Beg" // from upper pad layer to lowest layout layer
"End" // from upper layout layer to lowest pad layer
"UTL" // from upper pad layer to lowest pad layer

<SolderballShape>:
<string> one of these choices
"None" // no solderball
"Cyl" // cylinder solderball
"Sph" // spheroid solderball

<SolderballPlacement>:
<string> one of these choices
"abv" // above padstack
"blw" // below padstack

<PadPortLayerArray>:
```

Array(<int>, <int>, ....) where each int is a layer id

*VB Example:*

```
oPadstackManager.Edit "Circle - through1", Array("NAME:Circle - through1", "ModTime:=", _  
1235765635, "Library:=", "", "LibLocation:=", "Project", Array("NAME:psd", "nam:=", _  
"Circle - through1", "lib:=", "", "mat:=", "", "plt:=", "0", Array("NAME:pds", Array("NAME:lgm",  
"lay:=", _  
"Top Signal", "id:=", 0, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2.5mm"), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _  
"3.5mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "No", "Szs:=", Array()  
( ), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",  
"lay:=", _  
"SignalA", "id:=", 1, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2mm"), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _  
"3mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "No", "Szs:=", Array()  
( ), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",  
"lay:=", _  
"SignalB", "id:=", 2, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2mm"), "X:=", _  
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _
```

```

"3mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "thm:", Array("shp:=", "No", "Szs:", Array(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",
"lay:", _
"Ground", "id:", 3, "pad:", Array("shp:=", "No", "Szs:", Array(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:", Array("shp:=", "No", "Szs:", Array(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "thm:", Array("shp:=", "R90", "Szs:", Array( _
"3mm", "0.75mm", "1mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", _
"0mm", "dir:=", "Any"), Array("NAME:lgm", "lay:", "Bottom signal", "id:", 5, "pad:", Array(
"shp:=", _
"Cir", "Szs:", Array("1mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:", Array(
"shp:=", _
"Cir", "Szs:", Array("2mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "thm:", Array(
"shp:=", _
"No", "Szs:", Array(), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", _
"0mm", "dir:=", "Any)), "hle:", Array("shp:=", "Cir", "Szs:", Array("1.5mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "hRg:", "End", "sbsh:", "Sph", "sbpl:", _
"abv", "sbr:", "750um", "sb2:", "1200um", "1200um", "sbn:", "solder"), "ppl:", Array( _
0, 1, 2, 3, 5))

```

## **EditWithComps [padstack manager]**

*Use:* Edit an existing padstack.

*Command:* None

*Syntax:* EditWithComps <PadstackName>,

```
Array("NAME:<NewPadstackName>",
"ModTime:=", <ModifiedOnInfo>,
"Library:=", "", // name of the library
"LibLocation:=", "Project", // location of the named library
Array("NAME:psd",
"nam:=", <PadstackName>,
"lib:=", "", // name of the library
"mat:=", "", // hole plating material
"plt:=", "0", // percent of hole's radius filled by plating
Array("NAME:pds",
<LayerGeometryArray>,
<LayerGeometryArray....>),
"hlc:=", <PadInfo>
"hrG:=", <HoleRange>,
"sbsh:=", <SolderballShape>,
"sbpl:=", <SolderballPlacement>,
"sbr:=", <string>, // solderball diameter, real with units
"sb2:=", <string>, // solderball mid diameter, real with units
"sbn:=", <string>), // name of solderball material
"ppl:=", <PadPortLayerArray>,
```

```
Array(<ListOfComponentNames>) // Component names
```

*Return Value:* <string>

```
// composite name of the padstack.  
// If the name requested conflicts with the name of an existing  
// padstack, the requested name is altered to be unique.  
// The name returned reflects any change made to be unique.
```

*Parameters:* <PadstackName>:

```
<string> // composite name of the padstack being edited
```

<NewPadstackName>:

```
<string> // new simple name for the padstack
```

<ModifiedOnInfo>:

An integer that corresponds to the number of seconds that have elapsed  
since 00:00 hours, Jan 1, 1970 UTC from the system clock.

<LayerGeometryArray>:

```
Array("Name:lgm",  
"lay:=", <string>, // definition layer name
```

```
"id:=", <int>, // definition layer id  
"pad:=", <PadInfo>, // pad  
"ant:=", <PadInfo>, // antipad  
"thm:=", <PadInfo>, // themal pad  
"X:=", <string>, // pad x connection, real with units  
"Y:=", <string>, // pad y connection, real with units  
"dir:=", <DirectionString>) // pad connection direction
```

```
<PadInfo>:  
Array("shp:=", <PadShape>,  
"Szs:=", <DimensionArray>,  
"X:=", <string>, // x offset, real with units  
"Y:=", <string>, // y offset, real with units  
"R:=", <string>) // rotation, real with units
```

```
<PadShape>:  
<string> one of these choices  
"No" // no pad  
"Cir" // Circle  
"Sq" // Square
```

```
"Rct" // Rectangle  
"Ov" // Oval  
"Blt" // Bullet  
"Ply" // Polygons  
"R45" // Round 45 thermal  
"R90" // Round 90 thermal  
"S45" // Square 45 thermal  
"S90" // Square 90 thermal
```

```
<DimensionArray>:  
Array(<string>, ...) // each string is a real with units for one of the  
// dimensions of the shape
```

```
<DirectionString>:  
<string> one of these choices  
"No" // no direction  
"Any" // any direction  
"0" // 0 degrees  
"45" // 45 degrees  
"90" // 90 degrees  
"135" // 135 degrees
```

"180" // 180 degrees

"225" // 225 degrees

"270" // 270 degrees

"315" // 315 degrees

<HoleRange>:

<string> one of these choices

"Thr" // through all layout layers

"Beg" // from upper pad layer to lowest layout layer

"End" // from upper layout layer to lowest pad layer

"UTL" // from upper pad layer to lowest pad layer

<SolderballShape>:

<string> one of these choices

"None" // no solderball

"Cyl" // cylinder solderball

"Sph" // spheroid solderball

<SolderballPlacement>:

<string> one of these choices

"abv" // above padstack

```
"blw" // below padstack

<PadPortLayerArray>:
Array( <int>, <int>,....) where each int is a layer id

<ListOfComponentNames>:
<string>,<string> ...
// The list may be empty. When not empty, each string that is listed is a component
// that references the padstack to be edited. Prior to editing, a clone of the padstack is
// made, and the components that are listed are modified so that they now refer to
// the clone.
```

*VB Example:*

```
oPadstackManager>EditWithComps "Circle - through1", Array("NAME:Circle - through1", "ModTime:=",_
-
1235765635, "Library:=", "", "LibLocation:=", "Project", Array("NAME:psd", "nam:=", _
"Circle - through1", "lib:=", "", "mat:=", "", "plt:=", "0", Array("NAME:pds", Array("NAME:lgm",
"lay:=", _
"Top Signal", "id:=", 0, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2.5mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _
"3.5mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "No", "Szs:=", Array(
()), "X:=", _
```

```
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",
"lay:=", _
"SignalA", "id:=", 1, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _
"3mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",
"lay:=", _
"SignalB", "id:=", 2, "pad:=", Array("shp:=", "Cir", "Szs:=", Array("2mm"), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "Cir", "Szs:=", Array( _
"3mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "thm:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", "0mm", "dir:=", "Any"), Array("NAME:lgm",
"lay:=", _
"Ground", "id:=", 3, "pad:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=", "No", "Szs:=", Array(), "X:=", _
"0mm", "Y:=", "0mm", "R:=", "0"), "thm:=", Array("shp:=", "R90", "Szs:=", Array( _
"3mm", "0.75mm", "1mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", _
"0mm", "dir:=", "Any"), Array("NAME:lgm", "lay:=", "Bottom signal", "id:=", 5, "pad:=", Array(
"shp:=", _
"Cir", "Szs:=", Array("1mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "ant:=", Array("shp:=",
```

---

```
"Cir", "Szs:=", Array("2mm"), "X:=", "0mm", "Y:=", "0mm", "R:=", "0deg"), "thm:=", Array  
("shp:=", __  
"No", "Szs:=", Array(), "X:=", "0mm", "Y:=", "0mm", "R:=", "0"), "X:=", "0mm", "Y:=", __  
"0mm", "dir:=", "Any")), "hle:=", Array("shp:=", "Cir", "Szs:=", Array("1.5mm"), "X:=", __  
"0mm", "Y:=", "0mm", "R:=", "0deg"), "hRg:=", "End", "sbsh:=", "Sph", "sbpl:=", __  
"abv", "sbr:=", "750um", "sb2:=", "1200um", "1200um", "sbn:=", "solder"), "ppl:=", Array( __  
0, 1, 2, 3, 5), Array("")
```

## Export [padstack manager]

*Use:* Export a padstack to a library

*Command:* Tools > Edit Configured Libraries > Padstacks > Export to Library

*Syntax:* Export Array("NAME:<LibraryName>","  
    <PadstackName>,  
    <PadstackName>...),  
    <LibraryLocation>

*Return Value:* None

*Parameters:* <LibraryName>:

    <string> // name of the library

<PadstackName>:

    <string> // [simple name](#) of padstack to export

<LibraryLocation>:

```
<string> // location of the library in <LibraryName>
// One of "Project", "PersonalLib", or "UserLib"
```

*VB Example:*

```
oPadstackManager.Export Array("NAME:mylib", "myPadstack"), "PersonalLib"
```

## **GetData [padstack manager]**

*Use:* Gets data that describes the definition.

*Command:* None

*Syntax:* GetData(<DefinitionName>)

*Return Value:* <DefinitionData> This is an array of data for the definition.

*Parameters:* <DefinitionName>:

```
<string> // composite name of the definition to edit
```

*VB Example:*

```
Dim padstackData
padstackData = oPadstackManager.GetData("NoPad SMT East")
```

**Note:**

GetData allows the user to access definition information, make modifications, and then use the Edit or EditWithComps script commands to save the modified definition. Accordingly, for each type of definition, the array data returned to GetData should be the same array information that is supplied to the [Edit](#) or [EditWithComps](#) commands.

## GetNames [padstack manager]

*Use:* Returns the names of the padstack (used and unused) in a design. The following script command, **IsUsed**, can then be used to separate used and unused padstacks.

*Command:* None

*Syntax:* GetNames()

*Return Value:* An array of strings

*Parameters:* None

*VB Example:*

```
Dim padstackNames  
padstackNames = oPadstackManager.GetNames()
```

## IsUsed [padstack manager]

*Use:* Used to determine if a component is used in the design.

*Command:* None

*Syntax:* IsUsed(<PadstackName>)

*Return Value:* <Boolean> // true if the specified padstack is used in the design

*Parameters:* <PadstackName>:

<string>

*VB Example:*

```
Dim isUsed  
isUsed = oPadstackManager.IsUsed("MyPadstack")
```

## **Remove [padstack manager]**

*Use:* Removes a padstack from a library

*Command:* Tools > Edit Configured Libraries > Padstacks > Remove Padstacks

*Syntax:* Remove <PadstackName>,

```
<IsProjectPadstack>,  
<LibraryName>,  
<LibraryLocation>
```

*Return Value:* None

*Parameters:* <PadstackName>:

```
<string> // simple name of the padstack to remove
```

```
<IsProjectPadstack>:
```

```
<bool>
```

```
<LibraryName>:
```

```
<string> // name of the library
```

```
<LibraryLocation>:
```

```
<string> // location of the library in <LibraryName>
// One of "Project", "PersonalLib", or "UserLib"
```

*VB Example:*

```
oPadstackManager.Remove "Polygon SMT", true, "Padstacks", "Project"
oPadstackManager.Remove "Polygon SMT", false, "MyLib", "PersonalLib"
```

## RemoveUnused [padstack manager]

*Use:* Removes padstacks that are not used in the design.

*Command:* Project->**Remove Unused Definitions** is similar but operates slightly different and does not record script commands.

*Syntax:* RemoveUnused()

*Return Value:* <bool> True if one or more padstacks are removed.

*Parameters:* None

*VB Example:*

```
Dim removedDefs
removedDefs = oPadstackManager.RemoveUnused()
```

### Note:

The order of calls to RemoveUnused is significant. As a result, removing definitions in an unordered fashion may cause other padstacks in dependent definitions to be rendered unusable.

## Script and Library Scripts

The definition manager provides access to materials in a project. The manager object is accessed via the definition manager.

```
Set oDefinitionManager = oProject.GetDefinitionManager()
```

The script and library script commands are listed below.

[AddScript](#)

[EditScript](#)

[ExportScript](#)

[RemoveScript](#)

[ModifyLibraries](#)

### AddScript

Adds a script to the definition manager.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;AddScriptArray&gt;</i>	Type Array	Description Structured array.  Array ("NAME:<string script name>", "ScriptLang:=", <string "vbscript" or "javascript"> "ScriptText:=, <string text of script>")
<b>Return Value</b>	None.		

<b>Python Syntax</b>	AddScript(<AddScriptArray>)
<b>Python Example</b>	<pre> oDefinitionManager.AddScript(     [         "NAME:MyScript",         "ScriptLang:=", "vbscript",         "ScriptText:=", "MsgBox(\\"HelloWorld\\")"     ] ) </pre>

<b>VB Syntax</b>	AddScript <AddScriptArray>
<b>VB Example</b>	<pre> oDefinitionManager.AddScript Array("NAME:MyScript", "ScriptLang:=", "vbscript",     "ScriptText:=", "MsgBox("""HelloWorld""")") </pre>

## EditScript

Edits a script in the definition manager.

<b>UI Access</b>	N/A											
<b>Parameters</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>&lt;OriginalName&gt;</td> <td>String</td> <td>Name of the script to be edited.</td> </tr> <tr> <td>&lt;EditScriptArray&gt;</td> <td>Array</td> <td>           Structured array.            Array("NAME:&lt;string script name&gt;",           "ScriptLang:=&lt;string vbscript or javascript&gt;",           "ScriptText:=&lt;string text of script&gt;")         </td> </tr> </tbody> </table>	Name	Type	Description	<OriginalName>	String	Name of the script to be edited.	<EditScriptArray>	Array	Structured array. Array("NAME:<string script name>",           "ScriptLang:=<string vbscript or javascript>",           "ScriptText:=<string text of script>")		
Name	Type	Description										
<OriginalName>	String	Name of the script to be edited.										
<EditScriptArray>	Array	Structured array. Array("NAME:<string script name>",           "ScriptLang:=<string vbscript or javascript>",           "ScriptText:=<string text of script>")										
<b>Return Value</b>	None.											

<b>Python Syntax</b>	EditScript(<OriginalName>, <EditScriptArray>)
<b>Python Example</b>	<pre>oDefinitionManager.EditScript("MyScript",                                ["NAME:MyNewScript",                                 "ScriptLang:=", "vbscript",                                 "ScriptText:=", "MsgBox(\"HelloAgain\")"                                ],                            )</pre>

<b>VB Syntax</b>	EditScript <OriginalName> <EditScriptArray>
<b>VB Example</b>	<pre>oDefinitionManager&gt;EditScript "MyScript" Array("NAME:MyNewScript",                                                  "ScriptLang:=", "vbscript", "ScriptText:=", "MsgBox(""HelloAgain"")")</pre>

## ExportScript

*Use:* Export to Library in the script definition manager

*Command:* None

*Syntax:* ExportScript <ExportData>,<Library location>

*Return Value:* None

*Parameters:* <ExportData>

Array ("NAME:<LibraryName>",<ScriptName>,<ScriptName>,...)

*VB Example:*

```
oProject.ExportComponent Array("NAME:mylib", "myscript"), "PersonalLib"
```

<b>Python Syntax</b>	ExportScript( <ExportData>, <Library location>)
<b>Python Example</b>	<pre>oProject.ExportComponent ([ "NAME:mylib", "myscript"], "PersonalLib")</pre>

## RemoveScript

*Use:* Remove Script in the script definition manager

*Command:* None

*Syntax:* RemoveScript <ScriptName>,<IsProjectScript>,<LibraryName>,<LibraryLocation>

*Return Value:* None

*Parameters:* <ScriptName>

Type: <string>

<IsProjectScript>

Type: <bool>

<LibraryName>

Type: <string>

<LibraryLocation>

Type: <string>

*VB Example:*

```
oDefinitionManager.RemoveScript "myscript", true, "Local", "Project"
```

<b>Python Syntax</b>	RemoveScript (<ScriptName>, <IsProjectScript>, <LibraryName>, <LibraryLocation>)
<b>Python Example</b>	<pre>oDefinitionManager.RemoveScript(     "myscript", true, "Local", "Project")</pre>

## ModifyLibraries

*Use:* Configure Libraries on the Tools menu

*Command:* None

*Syntax:* ModifyLibraries <DesignName>,Array(<ConfigLibArray>)

*Return Value:* None

*Parameters:* <DesignName>

*Type:* <string>

<ConfigLibArray>

Array ("NAME:<LibraryType>,<ConfiguredLib>,<ConfiguredLib>,...), ...

<ConfiguredLib> // blank to leave unchanged

<DefinitionType>

Array("<libraryname >","<libraryname>,...)

*VB Example:*

```
oDefinitionManager.ModifyLibraries "MyCircuit", _
    Array("NAME:PersonalLib"), _
    Array("NAME:UserLib"), _
    Array("NAME:SystemLib", _
```

```
"Symbols:=", Array( "Circuit Elements", "Symbols", _  
"ParamExtraElements\PE_Symbols", _  
"Vendor Elements\Nonlinear"))
```

Python Syntax	ModifyLibraries (<DesignName>,[<ConfigLibArray>])
Python Example	<pre>oDefinitionManager.ModifyLibraries(     "MyCircuit", _     [ "NAME:PersonalLib"], _     [ "NAME:UserLib"], _     [ "NAME:SystemLib", _         "Symbols:=", [ "Circuit Elements", "Symbols", _         "ParamExtraElements\PE_Symbols", _         "Vendor Elements\Nonlinear"]])</pre>

This page intentionally  
left blank.

# 28 - Definition Editor Script Commands

The Definition Editor controls the use of materials and scripts in a project. Symbol editor script commands and footprint editor script commands are accessed using the Definition Editor.

```
Set oDefinitionEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", "MySymbol")
```

```
Set oDefinitionEditor = oProject.SetActiveDefinitionEditor("FootprintEditor", "MyFootprint")
```

**The topics for this section include:**

**Also refer to the following subtopics:**

[Symbol Editor Scripts](#)

## Symbol Editor Scripts

Symbol editor script commands are accessed with a definition editor.

```
Set oDefinitionEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", "MySymbol")
```

The symbol editor script commands are listed below.

[AddLevel \(Symbol Editor\)](#)

[AlignHorizontal \(Symbol Editor\)](#)

[AlignVertical \(Symbol Editor\)](#)

"BringToFront (Symbol Editor)" on page 28-6

[ChangeProperty \(Symbol Editor\)](#)

[CloseEditor \(Symbol Editor\)](#)

[CreateArc \(Symbol Editor\)](#)

[CreateCircle \(Symbol Editor\)](#)

[CreateImage \(Symbol Editor\)](#)

[CreateLine \(Symbol Editor\)](#)

[CreatePin \(Symbol Editor\)](#)

[CreatePolygon \(Symbol Editor\)](#)

[CreateRectangle \(Symbol Editor\)](#)

[CreateText \(Symbol Editor\)](#)

[Cut \(Symbol Editor\)](#)

[Delete \(Symbol Editor\)](#)

[DisplayPinNames \(Symbol Editor\)](#)

[ExportFile \(Symbol Editor\)](#)

[FlipHorizontal \(Symbol Editor\)](#)

[FlipVertical \(Symbol Editor\)](#)

[GetProperties \(Symbol Editor\)](#)

[GetPropertyValue \(Symbol Editor\)](#)

[GetVisibleLevels \(Symbol Editor\)](#)

[GridSetup \(Symbol Editor\)](#)

[Move \(Symbol Editor\)](#)

[Pan \(Symbol Editor\)](#)

[Paste \(Symbol Editor\)](#)

[Redo \(Symbol Editor\)](#)  
[RemoveLevels \(Symbol Editor\)](#)  
[RemovePort \(Symbol Editor\)](#)  
[Rotate \(Symbol Editor\)](#)  
[Save \(Symbol Editor\)](#)  
[SelectAll \(Symbol Editor\)](#)  
[SendToBack \(Symbol Editor\)](#)  
[SetActiveLevel \(Symbol Editor\)](#)  
[SetInitialLevels \(Symbol Editor\)](#)  
[SetPropertyValue \(Symbol Editor\)](#)  
[SetVisibleLevels \(Symbol Editor\)](#)  
[ToggleLevel \(Symbol Editor\)](#)  
[Undo \(Symbol Editor\)](#)  
[ZoomArea \(Symbol Editor\)](#)  
[ZoomIn \(Symbol Editor\)](#)  
[ZoomOut \(Symbol Editor\)](#)  
[ZoomPrevious \(Symbol Editor\)](#)  
[ZoomToFit \(Symbol Editor\)](#)

## **AddLevel (Symbol Editor)**

To add a graphics level to the symbol.

*Command:***Add Level** in the **View>Levels** dialog.

*Syntax:* AddLevel <level>

*Return Value:* Integer, the previous active level. The current active level is the newly added level.

*Parameters:* <level>

Type: Integer

The new level, an integer greater than 1.

*Example:* Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _  
"Simplorer Elements\basic Elements\Circuit\Passive Elements:R")  
oldLevel = oSymbolEditor.AddLevel(4)
```

## AlignHorizontal (Symbol Editor)

Align items horizontally

*Syntax:* AlignHorizontal(

```
Array("NAME:Selections", _  
"Page:=", page number, _ // [Opt=1] Page number  
"Selections:=", IDs to modify),  
Array("NAME:AlignParameters", _  
"Disconnect:=", bool _ // [Opt=0] Should wires disconnect  
"Rubberband:=", bool) _ // [Opt=1] Should wires staircase  
// Note: Alignment occurs relative to the first item in ids
```

<b>Python Syntax</b>	AlignHorizontal (["NAME:Selections", "Selections:=", [<selections>]], 0)
<b>Python Example</b>	<pre> oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _ "Simpler Elements\basic Elements\Circuit\PassiveElements:R")  oSymbolEditor.AlignHorizontal (["NAME:Selections", _ "Selections:=", ["SchObj@11"]], 0) </pre>

## AlignVertical (Symbol Editor)

Align items vertically

Syntax: AlignVertical(

```

Array("NAME:Selections", _
"Page:=", page number, _ // [Opt=1] Page number
"Selections:=", IDs to modify),
Array("NAME:AlignParameters", _
"Disconnect:=", bool _ // [Opt=0] Should wires disconnect
"Rubberband:=", bool) _ // [Opt=1] Should wires staircase
// Note: Alignment occurs relative
to the first item in ids

```

<b>Python Syn-tax</b>	AlignVertical (["NAME:Selections", "Selections:=", [<selections>]], 0)
<b>Python Example</b>	<pre> oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _ </pre>

```
"Simplorer Elements\basic Elements\Circuit\PassiveElements:R")
oSymbolEditor.AlignVertical ([ "NAME:Selections", "Selections:=", [ "SchObj@11" ] ], 0)
```

## BringToFront (Symbol Editor)

Changes the drawing for the symbol so that the specified objects are drawn on top of other overlapping objects.

**Command:**Draw > Bring To Front

**Syntax:** BringToFront Array("NAME:Selections", "Selections:=", Array (<Object>, <Object>, ...))

**Return Value:** None.

**Parameters:** <string><Object>  
<string> // object to bring to the front

**Example:**

```
oDefinitionEditor.BringToFront Array("NAME:Selections", "Selections:=", Array( "SchObj@10"))
```

## ChangeProperty (Symbol Editor)

**Use:** Changes to properties are scripted using the **ChangeProperty** command. This command can be executed by the **oEditor** to change editor properties, by the **oDesign** to change design level properties, and by the **oProject** to change project level properties. The command can be used to create, edit, and/or remove properties. In Circuit, only Variable and Separator properties can be deleted.

**Command:** None

**Syntax:** ChangeProperty Array("Name:AllTabs", <PropTabArray>, <PropTabArray>, ...)

**Return Value:** None

**Parameters:** <PropTabArray>  
Array("Name:<PropTab>",

```
<PropServersArray>,
<NewPropsArray>,
<ChangedPropsArray>,
<DeletedPropsArray>

<PropServersArray>
Array("Name:PropServers", <PropServer>,
<PropServer>, ...)

<NewPropsArray>
Array("Name>NewProps", <PropdataArray>,
<PropdataArray>, ...)

<ChangedPropsArray>
Array("Name>ChangedProps", <PropdataArray>,
<PropdataArray>, ...)

<DeletedPropsArray>
Array("Name>DeletedProps", <PropName>,
<PropName>, ...)
OR (for PropDisplay deletions only)
```

```
Array("Name:DeletedProps",<PropdataArray>,
      <PropdataArray>, ...)
```

```
<PropdataArray>
  Array("NAME:<PropName>",
        "PropType:=", <PropType>,
        "NewName:=", <string>,
        "Description:=", <string>,
        "Callback:=", <string>,
        "NewRowPosition:=", <int>,
        "ReadOnly:=", <bool>,
        "Hidden:=", <bool>,
        <PropTypeSpecificArgs>)OR (for PropDisplays only)
  Array("Name:<PropName>",<PropDisplayData>)
```

```
<PropDisplayData>
  for adding, changing, deleting PropDisplays
  <PropDisplayAttributes>
    for changing PropDisplays only
```

```
<PropDisplayNewAttributes>
```

```
<PropDisplayAttributes>
```

Layer & Location only used for PropDisplays in layout

For adding PropDisplays, this will add a single PropDisplay with attributes as shown; if an attribute is missing, a default value will be assigned. Adding PropDisplay to schematic with attributes that are identical to one already existing there will fail without an error message.

For deleting PropDisplays, these attributes are used to identify an existing PropDisplay to delete. If there doesn't exist a PropDisplay that matches the given attributes, then nothing will be deleted. If multiple PropDisplays match the given attributes, then all of them will be deleted. If an attribute is missing, then all PropDisplays match that missing attribute. For example, if Layer is missing, then PropDisplays on all layers that match the remaining given attributes will be deleted.

For changing PropDisplays, these attributes are used to identify an existing PropDisplay to change. If no PropDisplay matching the attributes is found, no changes will be made. If multiple PropDisplays match the attributes, all of them will be changed. If an attribute is missing, it matches all PropDisplays. For example, to change the format of PropDisplays that are on the bottom, but have any layer, style or format to show the name only, this command should have Location set to "Bottom" and all other attributes omitted.

"Format:=", <PropDisplayType>,

"Location:=", <PropDisplayLocation>,

"Layer:=", <string>,

"Style:=", <string>

```
<PropDisplayNewAttributes>
```

NewLayer & NewLocation only used for PropDisplays in layout

For changing PropDisplays, these attributes are used to identify which attributes to change and what the new value is. If the attribute should not be changed, the corresponding entry should be omitted.

"NewName:=", <string>,

"NewFormat:=", <PropDisplayType>,

"NewLocation:=", <PropDisplayLocation>,

"NewLayer:=", <string>,

"NewStyle:=", <string>

<PropDisplayType>

Type: string

Identifies the format of PropDisplay.

"Name"

"Value"

"NameAndValue"

"EvaluatedValue"

"NameAndEvaluatedValue"

<PropDisplayLocation>

Type: string

Identifies where PropDisplay is located with respect to object

"Left"

"Top"

"Right"

"Bottom"

"Custom"

<PropType>

Type: string

Identifies the type of property when a new property is added. In Circuit, only separator properties and variable properties can be added.

"SeparatorProp"

"VariableProp"

"TextProp"

"NumberProp"

"ValueProp"

"CheckboxProp"

"MenuProp"

"PointProp"

"VPointProp"

"ButtonProp"

NewName

Specify the new name of a property if the property's name is being edited. In Circuit, the name can only be changed for separators and variables.

#### Description

Specify a description of the property. In Circuit, the description can only be changed for separators and variables.

#### Callback

Specify the name of the script callback to be run when the property value is changed.

#### NewRowPosition

Used to reorder rows in the **Property** dialog box. In Circuit, this only applies to the **Project>Project Variables** panel and the **Designer>DesignProperties** panel. Specify the new zero-based row index of the variable or separator.

#### ReadOnly

Used to mark a property as "read only" so it can not be modified. In Circuit, this flag can only be set for variables and separators.

#### Hidden

Used to hide a property so it can not be viewed outside of the **Property** dialog box. In Circuit, this flag can only be set for variables and separators.

#### <PropTypeSpecificArgs>

**SeparatorProp:** no arguments

TextProp: "Value:=", <string>

NumberProp: "Value:=", <double>

ValueProp: "Value:=", <value>

CheckboxProp: "Value:=", <bool>

MenuProp: "Value:=", <string>

PointProp "X:=", <double>, "Y:=", <double>

VPointProp: "X:=", <value>, "Y:=", <value>

Material Button: "Material:=", <string>

Color Button: "R:="",<int>,"G:="",<int>,"B:="",<int>

Transparency Button: "Value:=", <double>

<PropTypeSpecificArgs> for MenuProps

Syntax for NewProps array: "AllChoices:=",  
<"choice1,choice2,..."> or <Array("choice1" "choice2", ... )>,  
"Value:=", <string>

Syntax for ChangedProps array: "Value:=", <string>

<PropTypeSpecificArgs> for VariableProps

Syntax:

**"Value:=", <value>, <OptimizationFlagsArray>,**

```
<TuningFlagsArray>, <SensitivityFlagsArray>,
<StatisticsFlagsArray>
```

Parameters:

```
<OptimizationFlagsArray>
Array("NAME:Optimization",
"Included:=", <bool>,
"Min:=", <value>,
"Max:=", <value>)
```

```
<TuningFlagsArray>
Array("NAME:Tuning",
"Included:=", <bool>,
"Step:=", <value>,
"Min:=", <value>,
"Max:=", <value>)
```

```
<SensitivityFlagsArray>
Array("NAME:Sensitivity",
"Included:=", <bool>,
"Min:=", <value>,
```

```
"Max:=", <value>,
"IDisp:=", <value> )
```

```
<StatisticsFlagsArray>
Array("NAME:Statistical",
"Included:=", <bool>,
"Dist:=", <Distribution>,
"StdD:=", <value>,
"Min:=", <value>,
"Max:=", <value>,
"Tol:=", <string>)
```

```
<Distribution>
Type: string
Value should be "Gaussian" or "Uniform"
```

StdD

Standard deviation.

Min

Low cut-off for the distribution.

Max

High cut-off for the distribution.

Tol

Tolerance for uniform distributions. Format is "<int>%".

Example: "**20%**".

*VB Example:*

Adding a new project level variable "\$width":

```
oProject.ChangeProperty Array("NAME:AllTabs",_
Array("NAME:ProjectVariableTab",_
Array("NAME:PropServers", "ProjectVariables"),_
Array("NAME:NewProps",_
Array("NAME:$width",_
"PropType:=", "VariableProp",_
"value:=", "3mm",_
"description:=", "my new variable")))
```

*VB Example:* Deleting the design level variable "height":

```
oDesign.ChangeProperty Array("NAME:AllTabs",_
Array("NAME:LocalVariableTab",_
Array("NAME:PropServers", "DefinitionParameters"),_
Array("NAME:DeletedProps", "height"))
```

Changing a property's value. If the following command were executed, then the value of the property "XSize" of the PropServer

"Box1>CreateBox:1" on the "Geometry3DCmdTab" tab would be changed. (oEditor is the Geometry3D editor in Circuit.)

```
oEditor.ChangeProperty Array("NAME:AllTabs",_
Array("NAME:Geometry3DCmdTab",_
Array("NAME:PropServers", "Box1>CreateBox:1"),_
Array("NAME:ChangedProps",_
Array("NAME:XSize", "Value:=", "1.4mil"))))
```

*VB Example:* Changing a property's value. If the following command were executed, then the values of Callback and L on the PassedParameterTab would be changed.

```
oEditor.ChangeProperty Array("NAME:AllTabs", _
Array("NAME:PassedParameterTab", _
Array("NAME:PropServers", "CHOKE2"), _
Array("NAME:ChangedProps", _
Array("NAME:L", "Callback:=", "ac", "OverridingDef:=", true), _
```

```
Array("NAME:L", "Value:=", "1nH")))
```

*VB Example:* Changing the Company Name, Design Name, the background color, and the Axis scaling in a Report.

```
Set oProject = oDesktop.SetActiveProject("wgcombiner")
Set oDesign = oProject.SetActiveDesign("CircuitDesign2")
Set oModule = oDesign.GetModule("ReportSetup")
oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Header", _ Array("NAME:PropServers", "XY
Plot1:Header"), _
Array("NAME:ChangedProps", Array("NAME:Company Name", _
"Value:=", "My Company")))
oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Header", _ Array("NAME:PropServers", "XY
Plot1:Header"), _
Array("NAME:ChangedProps", Array("NAME:Design Name", _
"Value:=", "WG Combiner")))
oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:General", _ Array("NAME:PropServers", "XY
Plot1:General"), _
Array("NAME:ChangedProps", Array("NAME:Back Color", _
"R:=", 128, "G:=", 255, "B:=", 255)))
oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:Axis", _ Array("NAME:PropServers", "XY
Plot1:AxisX"), _
```

```
Array("NAME:ChangedProps", Array("NAME:Axis Scaling", _  
"Value:=", "Log")))
```

*VB Example:* Changing a property's value. Note that the AllChoices parameter is only used when the MenuProp is being added. Also note that either a string of choices separated by commas or an Array("choice1", "choice2", "choice3") works for the AllChoices parameter.

```
Set oEditor = oDesign.SetActiveEditor("SchematicEditor")
oEditor.ChangeProperty Array("NAME:AllTabs", Array("NAME:PassedParameterTab", Array("NAME:PropServers", _  
"CompInst@CAP_2"), Array("NAME>NewProps", Array("NAME:xxxx", "PropType:=", _  
"MenuProp", "AllChoices:=", Array("aa", "bb", "cc", "dd"), "Value:=", "bb"))))
```

## CloseEditor (Symbol Editor)

Closes the specified editor.

*Command:* None

*Syntax:* CloseEditor (VARIANT ptDelta)

*Return Value:* None

<b>Python Syntax</b>	CloseEditor()
<b>Python Example</b>	oDefinitionEditor.CloseEditor()

## **Copy (Symbol Editor)**

Copy items for pasting.

*Command:***Edit>Copy**

*Syntax:* Copy Array("NAME:Selections", "Selections:=", Array(<selections>))

*Return Value:* None.

*Parameters:* <selections>

Type: Comma-separated list of strings

Identifiers for each selected item to be copied, in the form SchObj@<id>, where id is the value of the SchematicID property for each element.

*Example:* Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _  
"Simplorer Elements\basic Elements\Circuit\Passive Elements:R")  
oSymbolEditor .Copy Array("NAME:Selections", "Selections:=", Array("SchObj@11", "SchObj@34"))
```

## **CreateArc (Symbol Editor)**

Draws an arc in the symbol editor

*Command:* None

*Syntax:* CreateArc

*Return Value:* None

*VB Example:*

```
oDefinitionEditor.CreateArc Array("NAME:ArcData", _
```

```
"X:=", -0.004318, "Y:=", -0.00127, _
"Radius:=", 0.00297299377732279, _
"StartAng:=", 1.9195673303788, _
"EndAng:=", 3.32144615338227, _
"Id:=", 10), _
Array("NAME:Attributes", "Page:=", 1)
```

<b>Python Syntax</b>	CreateArc()
<b>Python Example</b>	<pre>oDefinitionEditor.CreateArc ([ "NAME:ArcData", _   "X:=", -0.004318, "Y:=", -0.00127, _   "Radius:=", 0.00297299377732279, _   "StartAng:=", 1.9195673303788, _   "EndAng:=", 3.32144615338227, _   "Id:=", 10], _   ["NAME:Attributes", "Page:=", 1])</pre>

## CreateCircle (Symbol Editor)

Draws a circle in the symbol editor

*Command:* None

*Syntax:* CreateCircle

*Return Value:* None

*VB Example:*

```
oDefinitionEditor.CreateCircle Array("NAME:CircleData", _  
"X:=", -0.004572, "Y:=", -0.000508, _  
"Radius:=", 0.001778, "Id:=", 12), _  
Array("NAME:Attributes", "Page:=", 1)
```

<b>Python Syntax</b>	CreateCircle()
<b>Python Example</b>	<pre>oDefinitionEditor.CreateCircle ( [ "NAME:CircleData", _ "X:=", -0.004572, "Y:=", -0.000508, _ "Radius:=", 0.001778, "Id:=", 12], _ [ "NAME:Attributes", "Page:=", 1 ] )</pre>

## **CreateImage (Symbol Editor)**

Create picture from a BMP or GIF file.

*Command:*Draw>Image

*Syntax:* CreateImage(

*Return Value:* None.

*Parameters:* Array("NAME:ImageData",  
"X1:=", double, // the image rectangle left X value, in meters  
"Y1:=", double, // the image rectangle upper Y value, in meters  
"X2:=", double, // the image rectangle right X value, in meters  
"Y2:=", double, // the image rectangle lower Y value, in meters  
"File:=", string, // the name of the BMP or GIF file  
"DisplayBorder:=", bool, // true or false, whether to display the image border  
"LineWidth:=", double, // the width of the image border, in meters  
"Color:=", int, // the RGB value of the image border color  
"Id:=", int), // Id for this item  
Array("NAME:Attributes",  
"Page:=", int) // [Opt=1] The page number (one-based)

*Example:* oDefinitionEditor.CreateImage Array("NAME:ImageData", "X1:=", 0.00229684174875881,  
"Y1:=", -0.00100608189091032, "X2:=", 0.00890589813779289, "Y2:=", -0.00432967005619977,  
"File:=", "C:/Documents and Settings/user/Desktop/logo.gif", "DisplayBorder:=", false,  
"LineWidth:=", 0, "Color:=", 0, "Id:=", 22), Array("NAME:Attributes", "Page:=", 1)

## CreateLine (Symbol Editor)

Draws a line in the symbol editor

*Command:* None

*Syntax:* CreateLine

*VB Example:*

```
oDefinitionEditor.CreateLine Array("NAME:LineData", _Return Value: None
```

```
"Points:=", Array( "(0.001778, -0.001270)", _  
"(0.004572, 0.002794)", "(0.003048, 0.003556)", _  
"Id:=", 14), Array("NAME:Attributes", "Page:=", 1)
```

<b>Python Syntax</b>	CreateLine()
<b>Python Example</b>	<pre>oDefinitionEditor.CreateLine (["NAME:LineData", _Return Value: None "Points:=", [ "(0.001778, -0.001270)", _ "(0.004572, 0.002794)", "(0.003048, 0.003556)"], _ "Id:=", 14], ["NAME:Attributes", "Page:=", 1])</pre>

## CreatePin (Symbol Editor)

Draws a pin in the symbol editor

*Command:* None

*Syntax:* CreatePin

*Return Value:* None

*VB Example:*

```
oDefinitionEditor.CreatePin Array("NAME:PinData", _  
"Name:=", "newpin3"), Array("NAME:PinParams", _  
"X:=", -0.00762, "Y:=", 0.00254, "Angle:=", 0, "Flip:=", false)
```

<b>Python Syntax</b>	CreatePin()
<b>Python Example</b>	<pre>oDefinitionEditor.CreatePin ([ "NAME:PinData", _     "Name:=", "newpin3"], [ "NAME:PinParams", _     "X:=", -0.00762, "Y:=", 0.00254, "Angle:=", _     0, "Flip:=", false])</pre>

## CreatePolygon (Symbol Editor)

Draws a polygon in the symbol editor

*Command:* None

*Syntax:* CreatePolygon

*Return Value:* None

*VB Example:*

```
oDefinitionEditor.CreatePolygon Array("NAME:PolygonData", _  
    "Points:=", Array( "(0.004826, -0.000508)", "(0.003048, -0.003048)", _  
    "(0.006858, -0.003302)"), "Id:=", 16), _  
    Array("NAME:Attributes", "Page:=", 1)
```

<b>Python Syntax</b>	CreatePolygon()
<b>Python Example</b>	<pre>oDefinitionEditor.CreatePolygon ([ "NAME:PolygonData", _     "X1:=", -0.001524, "Y1:=", 0.000508, _</pre>

```
"X2:=", 0.002286, "Y2:=", -0.002032, "Id:=", 18], __  
["NAME:Attributes", "Page:=", 1])
```

## CreateRectangle (Symbol Editor)

Draws a rectangle in the symbol editor

*Command:* None

*Syntax:* CreateRectangle

*Return Value:* None

*VB Example:*

```
oDefinitionEditor.CreateRectangle Array("NAME:RectData", __  
"X1:=", -0.001524, "Y1:=", 0.000508, __  
"X2:=", 0.002286, "Y2:=", -0.002032, "Id:=", 18), __  
Array("NAME:Attributes", "Page:=", 1)
```

<b>Python Syntax</b>	CreateRectangle()
<b>Python Example</b>	<pre>oDefinitionEditor.CreateRectangle ([ "NAME:RectData", __ "X1:=", -0.001524, "Y1:=", 0.000508, __ "X2:=", 0.002286, "Y2:=", -0.002032, "Id:=", 18], __ [ "NAME:Attributes", "Page:=", 1])</pre>

## CreateText (Symbol Editor)

Draws text in the symbol editor

*Command:* None

*Syntax:* CreateText

*Return Value:* None

*VB Example:*

```
oDefinitionEditor.CreateText Array("NAME:TextData", _
"X:=", 0.001524, "Y:=", 0.00127, _
"Text:=","My text", "Id:=", 20), Array("NAME:Attributes", "Page:=", 1)
```

<b>Python Syntax</b>	CreateText()
<b>Python Example</b>	<pre>oDefinitionEditor.CreateText( [ "NAME:TextData", "X:=", 0.001524, "Y:=", 0.00127, "Text:=","My text", "Id:=", 20], [ "NAME:Attributes", "Page:=", 1])</pre>

## Cut (Symbol Editor)

Cut page selection.

*Command:* None

*Syntax:* Cut(PageNumber, Selections)

*Return Value:* None

*VB Example:*

```
oDefinitionEditor.CreateText Array("NAME:Selections", _  
"Page:=", page number, _ // [Opt=1] Page number  
"Selections:=", IDs to modify)) 1)
```

<b>Python Syntax</b>	Cut(["NAME:Selections", "Selections:=",[<selections>]])
<b>Python Example</b>	<pre>oSимвolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor",_ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSимвolEditor.Cut ([ "NAME:Selections", "Selections:=", [ "SchObj@11", "SchObj@3" ] ])</pre>

## Delete (Symbol Editor)

Delete items - remove from the symbol.

*Command:***Edit>Delete**

*Syntax:* Delete Array("NAME:Selections", "Selections:=", Array(<selections>))

*Return Value:* None.

*Parameters:* <selections>

Type: Comma-separated list of strings

Identifiers for each selected item to be deleted, in the form SchObj@<id>, where id is the value of the SchematicID property for each element.

*Example:* Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _  
"Simplorer Elements\basic Elements\Circuit\Passive Elements:R")
```

```
oSymbolEditor.Delete Array("NAME:Selections", "Selections:=", Array("SchObj@11", "SchObj@34))
```

## DisplayPinNames (Symbol Editor)

To turn on or off the display of pin names.

*Command:*None.

*Syntax:* DisplayPinNames <onoff>

*Return Value:* None.

*Parameters:* <onoff>

Type: Boolean

TRUE or FALSE to turn on or off the display of pin names.

*Example:* Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", "Simplorer Elements\basic  
Elements\Circuit\Passive Elements:R")
```

```
oSymbolEditor.DisplayPinNames TRUE
```

## ExportFile (Symbol Editor)

Export file of symbol.

*Command:*Symbol>Export File

*Syntax:* ExportFile <type>, <filename>

*Return Value:* None.

*Parameters:* <type>

Type: String

Type of file to export - only "EMF" is supported currently.

```
<filename>
```

Type: String

Path, name and extension for the file to be saved.

*Example:* Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _  
    "Simplorer Elements\basic Elements\Circuit\Passive Elements:R")  
oSymbolEditor.ExportFile "EMF", "c:\sym.emf"
```

## **FlipHorizontal (Symbol Editor)**

To flip the specified objects about the horizontal (x) axis.

**Command:**Draw>Flip Horizontal

**Syntax:** FlipHorizontal Array("NAME:Selections", "Selections:=", Array(<selections>)), 0

**Return Value:** None.

*Parameters:* <selections>

Type: Comma-separated list of strings

Identifiers for each selected item to be flipped, in the form SchObj@<id>, where id is the value of the SchematicID property for each element.

*Example:* Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _  
    "Simplorer Elements\basic Elements\Circuit\Passive Elements:R")  
oSymbolEditor.FlipHorizontal Array("NAME:Selections", "Selections:=", Array("SchObj@11")), 0
```

## FlipVertical (Symbol Editor)

To flip the specified objects about the horizontal (x) axis.

*Command:*Draw>Flip Vertical

*Syntax:* FlipVertical Array("NAME:Selections", "Selections:=", Array(<selections>)), 0

*Return Value:* None.

*Parameters:* <selections>

Type: Comma-separated list of strings

Identifiers for each selected item to be flipped, in the form SchObj@<id>, where id is the value of the SchematicID property for each element.

*Example:* Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _  
    "Simplorer Elements\basic Elements\Circuit\Passive Elements:R")  
oSymbolEditor.FlipVertical Array("NAME:Selections", "Selections:=", Array("SchObj@11")), 0
```

## GetProperties (Symbol Editor)

Gets a list of all the properties belonging to a specific **PropServer** and **PropTab**. This can be executed by the **oProject**, **oDesign**, or **oEditor** objects.

*Command:* None

*Syntax:* GetProperties( <PropTab>, <PropServer> )

*Return Value:* Variant array of strings – the names of the properties belonging to the prop server.

*VB Example:* Dim all\_props  
all\_props = oDesign.GetProperties("BaseElementTab",\_  
 "rect\_1")

## **GetPropertyValue (Symbol Editor)**

Gets the value of a single property. This can be executed by the **oProject**, **oDesign**, or **oEditor** objects.

*Command:* None

*Syntax:* GetPropertyValue(<PropTab>, <PropServer>, <PropName>)

*Return Value:* String representing the property value.

*VB Example:* value\_string =  
oEditor.GetPropertyValues("BaseElementTab",\_  
"rect\_1", "Name")

## **GetVisibleLevels (Symbol Editor)**

To determine the levels in the symbol.

*Command:* None.

*Syntax:* GetVisibleLevels

*Return Value:* An array of integers, which are the levels defined for the particular symbol.

*Parameters:* None.

*Example:* Dim oSymbolEditor  
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", \_  
"Simplorer Elements\basic Elements\Circuit\Passive Elements:R")  
levels = oSymbolEditor.GetVisibleLevels

## GridSetup (Symbol Editor)

Changes the settings on the symbol grid.

**Command:**Symbol>Grid Setup

**Syntax:** GridSetup(

**Return Value:** None.

*Parameters:* Array("NAME:Options", \_  
    "MajorGrid:=", string, \_ // Major grid size with units  
    "Divisions:=", int, \_ // Number of minor grid divisions  
    "MajorColor:=", int, \_ // RGB Color of major grid lines  
    "MinorColor:=", int, \_ // RGB Color of minor grid lines  
    >ShowGrid:=", bool, \_ // Should the grid be shown?  
    "SnapToGrid:=", bool, \_ // Should objects snap to grid?  
    "BackgroundColor:=", int, \_ // RGB Color of the background  
    "SaveAsDefault:=", bool)) // Should these settings be the default for new schematics

*Example:* oEditor.GridSetup Array(

```
"NAME:Options",  
"MajorGrid:=", "10mm",  
"Divisions:=", 10,  
"MajorColor:=", 0x00ff0000, // Red  
"MinorColor:=", 0x0000ff00, // Green  
"ShowGrid:=", true,
```

```
"SnapToGrid:=", true,  
"BackgroundColor:=", 0x00ffffff, // White  
"SaveAsDefault:=", false)
```

## Move (Symbol Editor)

To move the specified objects in X and Y.

*Command:*None.

*Syntax:* Move Array("NAME:Selections", "Selections:=", Array(<selections>)), Array("NAME:FlipParameters", "xdelta:=", <xval>, "ydelta:=",<yval>)

*Return Value:* None.

*Parameters:* <selections>

Type: Comma-separated list of strings

Identifiers for each selected item to be moved, in the form SchObj@<id>, where id is the value of the SchematicID property for each element.

<xval>

Type: Real number

The X delta to move.

<yval>

Type: Real number

The Y delta to move.

*Example:* Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor",
```

```
"Simplorer Elements\basic Elements\Circuit\Passive Elements:R")
oSymbolEditor. Move Array("NAME:Selections", "Selections:=", Array("SchObj@34")),
Array("NAME:FlipParameters", "xdelta:=", -0.00254, "ydelta:=", 0.00254)
```

## Pan (Symbol Editor)

Pans the display.

<b>UI Access</b>	<b>View &gt; Pan.</b>		
<b>Parameters</b>	Name <i>&lt;argSelections&gt;</i>	Type Array of doubles	Description First double pans along x-axis; second double pans along y-axis.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	Pan ( <i>&lt;argSelections&gt;</i> )
<b>Python Example</b>	<code>oDefinitionEditor.Pan([-0.00922653869663931, 0.00686839904222147])</code>

<b>VB Syntax</b>	Pan <i>&lt;argSelections&gt;</i>
<b>VB Example</b>	<code>oDefinitionEditor.Pan Array(-0.00922653869663931, 0.00686839904222147)</code>

## Paste (Symbol Editor)

To paste copied objects to a specified X and Y location.

**Command:****Edit>Paste**

*Syntax:* Paste Array("NAME:Attributes", "X:=", <xval>, "Y:=", <yval>)

*Return Value:* None.

*Parameters:* <xval>

Type: Real number

The X placement value.

<yval>

Type: Real number

The Y placement value.

*Example:* Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _  
    "Simplorer Elements\basic Elements\Circuit\Passive Elements:R")  
oSymbolEditor.Paste Array("NAME:Attributes", "X:=", -0.00254, "Y:=", 0.00254)
```

## **Redo (Symbol Editor)**

Redo the last operation

*Command:* Click **Edit>Redo**

*Syntax:* Redo

*Return Value:* None

*Parameters:* None

*VB Example:* oDesign.Redo

## RemoveLevels (Symbol Editor)

Removes one or more graphics levels from a symbol.

<b>UI Access</b>	<b>Tools &gt; Edit Libraries &gt; Symbols.</b> From Symbol editor, click <b>View &gt; Levels &gt; Remove Level.</b>		
<b>Parameters</b>	Name <code>&lt;/levels&gt;</code>	Type Array of integers	Description The levels to be deleted.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	<code>RemoveLevels (&lt;/levels&gt;)</code>
<b>Python Example</b>	<code>oSymbolEditor.RemoveLevels ([3, 4])</code>

<b>VB Syntax</b>	<code>RemoveLevels &lt;/levels&gt;</code>
<b>VB Example</b>	<pre> Dim oSymbolEditor  Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", -     "Simplorer Elements\basic Elements\Circuit\Passive     Elements:R")  oldLevel = oSymbolEditor.RemoveLevels(Array(3, 4)) </pre>

## RemovePort (Symbol Editor)

Selected pins are changed to vias. Selected edge terminal ports are removed.

*Syntax:* RemovePort <NAME:elements", <object\_name> ... // objects to be removed

*Return Value:* None.

*Parameters:* <object\_name>

Type: <String>

*VB Example:* oEditor.RemovePort Array("NAME:elements", "via\_195", "Port1")

## Rotate (Symbol Editor)

To rotate the specified objects about their common center.

*Command:*Draw>Rotate

*Syntax:* Rotate Array("NAME:Selections", "Selections:=", Array(<selections>)), Array("NAME:FlipParameters", <angle>)

*Return Value:* None.

*Parameters:* <selections>

Type: Comma-separated list of strings

Identifiers for each selected item to be rotated, in the form SchObj@<id>, where id is the value of the SchematicID property for each element.

<angle>

Type: A comma-separated string and real number

Either "Angle:=", <rad>, where <rad> is the angle in radians, or "Degrees:=", <deg>, where <deg> is the angle in degrees.

*Example:* Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor",
"Simpler Elements\basic Elements\Circuit\Passive Elements:R")
```

```
oSymbolEditor.Rotate Array("NAME:Selections", "Selections:=", Array("SchObj@11")) ,  
Array("NAME:FlipParameters", "Angle:=", 1.5707963267949)
```

## Save (Symbol Editor)

Saves any changes in the current symbol to the project.

*Command:*Symbol>Update Project

*Syntax:* Save

*Return Value:* None.

*Parameters:* None.

*Example:* Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _  
"Simplorer Elements\basic Elements\Circuit\Passive Elements:R")  
oSymbolEditor.Save
```

## SelectAll (Symbol Editor)

Select all elements in the symbol editor.

*Command:* None.

*Syntax:* SelectAll()

*Parameters:* None

*VB Example:* Dim removedDefs

```
removedDefs = oDefinitionEditor.SelectAll()
```

<b>Python Syntax</b>	SelectAll()
<b>Python Example</b>	<pre>removedDefs = oSymbolManager.SelectAll()</pre>

## SendToBack (Symbol Editor)

Changes the drawing for the symbol so that the specified objects are drawn behind other overlapping objects.

*Command:* Draw > Send To Back

*Syntax:* SendToBack Array("NAME:Selections", "Selections:=", Array (<Object>, <Object>, ...))

*Return Value:* None

*Parameters:* <Object>

<string> // object to send to the back

*VB Example:* oDefinitionEditor.SendToBack Array("NAME:Selections", "Selections:=", Array( "SchObj@10"))

<b>Python Syntax</b>	SendToBack (["NAME:Selections", "Selections:=",[<Object>, <Object>, ...]])
<b>Python Example</b>	<pre>oDefinitionEditor.SendToBack([ "NAME:Selections",                                "Selections:=", ["SchObj@10"] ])</pre>

## SetActiveLevel (Symbol Editor)

To set the active graphics level in the symbol editor.

*Command:*"Current" radio button for a level in the **View>Levels** dialog.

*Syntax:* SetActiveLevel <level>

*Return Value:* None.

*Parameters:* <level>

Type: Integer

The new active level.

*Example:* Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _
    "Simplorer Elements\basic Elements\Circuit\Passive Elements:R")
oldLevel = oSymbolEditor. SetActiveLevel (4)
```

## SetInitialLevels (Symbol Editor)

Sets one or more graphics levels as the default visible levels for a symbol. The levels submitted do not need to contain 1 or 0, which are always visible.

UI Access	<b>Tools &gt; Edit Libraries &gt; Symbols.</b> From Symbol editor, click <b>View &gt; Levels</b> . Select the <b>Default</b> check box for desired levels.		
Parameters	Name </levels>	Type Array of integers	Description The levels to be selected.
Return Value	None.		

Python Syntax	SetInitialLevels (</levels>)
Python Example	oSymbolEditor.SetInitialLevels([3, 4])

VB Syntax	SetInitialLevels </levels>
-----------	----------------------------

**VB Example**

```
Dim  
oSymbolEditor  
  
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor",  
-  
    "Simplorer Elements\basic Elements\Circuit\Passive  
    Elements:R")  
  
oldLevel = oSymbolEditor.SetInitialLevels(Array(3, 4))
```

## **SetPropertyValue (Symbol Editor)**

Set a property.

**Command:** None.

**Syntax:** SetPropertyValue(tab, item, propname, value)

**Parameters:** None

*VB Example:* Dim removedDefs

```
removedDefs = oDefinitionEditor.SetPropertyValue(string tab, // Tab with the property  
string item, // Name of the object  
string propname, // Name of the property  
string value) // The new value of the prop
```

## **SetVisibleLevels (Symbol Editor)**

Makes one or more graphics levels visible in the symbol editor.

<b>UI Access</b>	<b>Tools &gt; Edit Libraries &gt; Symbols.</b> From Symbol editor, click <b>View &gt; Levels</b> . Select the <b>Visible</b> check box for the desired levels.					
<b>Parameters</b>	<table border="1"> <tr> <td>Name <i>&lt;/levels&gt;</i></td> <td>Type Array of integers</td> <td>Description The levels to be made visible.</td> </tr> </table>			Name <i>&lt;/levels&gt;</i>	Type Array of integers	Description The levels to be made visible.
Name <i>&lt;/levels&gt;</i>	Type Array of integers	Description The levels to be made visible.				
<b>Return Value</b>	None.					

<b>Python Syntax</b>	<code>SetVisibleLevels (&lt;/levels&gt;)</code>
<b>Python Example</b>	<code>oSymbolEditor.SetVisibleLevels ([3, 4])</code>

<b>VB Syntax</b>	<code>SetVisibleLevels &lt;/levels&gt;</code>
<b>VB Example</b>	<pre> Dim oSymbolEditor  Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", -     "Simplorer Elements\basic Elements\Circuit\Passive     Elements:R")  oldLevel = oSymbolEditor.SetVisibleLevels(Array(3, 4)) </pre>

## **ToggleLevel (Symbol Editor)**

To toggle display of the graphics level in the symbol editor.

*Command:*"Visible" checkbox for a level in the **View>Levels** dialog.

*Syntax:* `ToggleLevel <level>`

*Return Value:* None.

*Parameters:* <level>

Type: Integer

The level to turn on or off.

*Example:* Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _  
    "Simplorer Elements\basic Elements\Circuit\Passive Elements:R")  
oSymbolEditor.ToggleLevel (2)
```

## **Undo (Symbol Editor)**

Undo the last operation

*Command:* Edit>Undo

*Syntax:* Undo

*Return Value:* None

*Parameters:* None

*VB Example:* oDesign.Undo

<b>Python Syntax</b>	Undo()
<b>Python Example</b>	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _     "Simplorer Elements\basic Elements\Circuit\PassiveElements:R")</pre>

```
oSymbolEditor.Undo()
```

## ZoomArea (Symbol Editor)

To change the current graphics view area.

**Command:** View>Zoom Area

**Syntax:** ZoomArea Array(<xpt>, <ypt>), Array(<xdelta>, <ydelta>)

**Return Value:** None.

*Parameters:* <xpt>

Type: Real number

The X value of the zoom area box, in meters.

<ypt>

Type: Real number

The X value of the zoom area box, in meters.

<xdelta>

Type: Real number

The width of the zoom area box, in meters.

<ydelta>

Type: Real number

The height of the zoom area box, in meters.

*Example:* Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _  
"Simplorer Elements\basic Elements\Circuit\Passive Elements:R")
```

```
oSymbolEditor.ZoomArea Array(0, 0), Array(0.004, 0.004)
```

## **ZoomIn (Symbol Editor)**

To zoom in a standard amount to the current graphics view area.

*Command:* View>Zoom In

*Syntax:* ZoomIn

*Return Value:* None.

*Parameters:* None.

*Example:* Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor",
    "Simplorer Elements\basic Elements\Circuit\Passive Elements:R")
oSymbolEditor.ZoomIn
```

## **ZoomOut (Symbol Editor)**

To zoom out a standard amount to the current graphics view area.

*Command:* View>Zoom Out

*Syntax:* ZoomOut

*Return Value:* None.

*Parameters:* None.

*Example:* Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor",
    "Simplorer Elements\basic Elements\Circuit\Passive Elements:R")
```

```
oSymbolEditor.ZoomOut
```

## **ZoomPrevious (Symbol Editor)**

To zoom the current graphics view area to the previous extents.

*Command:* View>Zoom Previous

*Syntax:* ZoomPrevious

*Return Value:* None.

*Parameters:* None.

*Example:* Dim oSymbolEditor

```
Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor",
    "Simplorer Elements\basic Elements\Circuit\Passive Elements:R")
oSymbolEditor.ZoomPrevious
```

## **ZoomToFit (Symbol Editor)**

Set the current symbol zoom to fit the contents of the currently visible page

*Command:* None

*Syntax:* ZoomToFit()

*Return Value:* None

<b>Python Syntax</b>	ZoomToFit()
<b>Python Example</b>	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor",     "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSymbolEditor.ZoomToFit()</pre>

This page intentionally  
left blank.

# 29 - Core Global Script Context Commands

To run these commands:

```
import CoreGlobalScriptContextFunctions
CoreGlobalScriptContextFunctions.[CommandName]
```

The following are general script commands recognized by the **CoreGlobalScriptContextFunctions** object:

- [AddErrorMessage](#)
- [AddFatalMessage](#)
- [AddInfoMessage](#)
- [AddWarningMessage](#)
- [LogDebug](#)
- [LogError](#)

## AddErrorMessage

Adds an error message to the **Message Manager** window. AddErrorMessage is a function of CoreGlobalScriptContextFunctions.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;message&gt;</i>	Type String	Description Error message.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	AddErrorMessage( <i>&lt;message&gt;</i> )
<b>Python Example</b>	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.AddErrorMessage('My error message.')</pre>

<b>VB Syntax</b>	N/A
<b>VB Example</b>	N/A. Script cannot be run in VBScript.

## AddFatalMessage

Adds a fatal error message to the **Message Manager** window. AddFatalMessage is a function of CoreGlobalScriptContextFunctions.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;message&gt;</i>	Type String	Description Error message.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	AddFatalMessage( <i>&lt;message&gt;</i> )
<b>Python Example</b>	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.AddFatalMessage('My fatal error message.')</pre>

<b>VB Syntax</b>	N/A
<b>VB Example</b>	N/A. Script cannot be run in VBScript.

## AddInfoMessage

Adds an informational message to the **Message Manager** window. AddInfoMessage is a function of CoreGlobalScriptContextFunctions.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <i>&lt;message&gt;</i>	Type String	Description Informational message.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	AddInfoMessage( <i>&lt;message&gt;</i> )
<b>Python Example</b>	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.AddInfoMessage('My info.')</pre>

<b>VB Syntax</b>	N/A
<b>VB Example</b>	N/A. Script cannot be run in VBScript.

## AddWarningMessage

Adds a warning message to the **Message Manager** window. AddWarningMessage is a function of CoreGlobalScriptContextFunctions.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;message&gt;</td> <td>String</td> <td>Warning message.</td> </tr> </table>			Name	Type	Description	<message>	String	Warning message.
Name	Type	Description							
<message>	String	Warning message.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	AddWarningMessge(<message>)
<b>Python Example</b>	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.AddWarningMessage('My warning.')</pre>

<b>VB Syntax</b>	N/A
<b>VB Example</b>	N/A. Script cannot be run in VBScript.

## LogDebug

Adds a debug line to the log specified at **Tools > Debug Logging**. LogDebug is a function of CoreGlobalScriptContextFunctions.

<b>UI Access</b>	N/A								
<b>Parameters</b>	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> <tr> <td>&lt;message&gt;</td> <td>String</td> <td>Debug message.</td> </tr> </table>			Name	Type	Description	<message>	String	Debug message.
Name	Type	Description							
<message>	String	Debug message.							
<b>Return Value</b>	None.								

<b>Python Syntax</b>	LogDebug(<message>)
----------------------	---------------------

<b>Python Example</b>	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.LogDebug('My debug message.')</pre>
-----------------------	-------------------------------------------------------------------------------------------------------------------

<b>VB Syntax</b>	N/A
<b>VB Example</b>	N/A. Script cannot be run in VBScript.

## .LogError

Adds an error line to the log specified at **Tools > Debug Logging**. LogError is a function of CoreGlobalScriptContextFunctions.

<b>UI Access</b>	N/A		
<b>Parameters</b>	Name <error>	Type String	Description Error to log.
<b>Return Value</b>	None.		

<b>Python Syntax</b>	.LogError(<error>)
<b>Python Example</b>	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.LogError('My error.')</pre>

<b>VB Syntax</b>	N/A
<b>VB Example</b>	N/A. Script cannot be run in VBScript.

# 30 - Example Scripts

This section contains [VBScript](#) and [IronPython](#) example scripts.

## VBScript Example Scripts

VBScript Examples:

- [Variable Helix Script](#)
- [HFSS Data Export Script](#)
- [ExampleGetLayeredImpedanceBoundaryPropertyNamesandValues.htm](#)

### Variable Helix Script

This sample script creates a tapered helix. Tapering helices is not supported from the interface.

The script includes comment lines, which are preceded by an apostrophe ( ') and offer explanations for each subsequent line or lines.

```
Dim oAnsoftApp
Dim oDesktop
Dim oProject
Dim oDesign
Dim oEditor
Dim oModule

Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
Set oProject = oDesktop.GetActiveProject()
Set oDesign = oProject.GetActiveDesign()
Set oEditor = oDesign.SetActiveEditor("3D Modeler")

' Declare the arrays and variables needed for building the polyline.
'

Dim points(), segments()

Dim NumPoints, R(2), P(2), PointsPerTurn, Turns, Units
```

```
' Establish the constant Pi.  
  
Pi = 4*Atn(1)  
  
' Retrieve the variable helix parameters from the user.  
  
' Start with the input for unit selection.  
,  
  
Units = InputBox("Select the units:"&Chr(13)&_  
"(cm,mm,um,in,mil)", "Variable Helix","mil",50,50)  
,  
  
' Check to make sure it is a valid unit.  
,  
  
Select Case Units  
  
Case "m"  
  
Units = ""  
  
Case "cm"  
  
Case "mm"  
  
Case "um"  
  
Case "in"  
  
Case "mil"  
  
Case Else  
  
MsgBox("Invalid Units - defaults to m")  
  
Units = ""  
  
End Select  
  
,  
  
' Obtain the other user-defined parameters.  
,  
  
Turns = InputBox("Select the number of turns (must be  
integer):","Variable Helix", 2,50,50)  
  
PointsPerTurn = InputBox("Select the points per turn:", _  
"Variable Helix",16,50,50)  
  
R(0) = InputBox("Select the initial Radius: ", _
```

```
"Variable Helix",10,50,50)

R(1) = InputBox("Select the final Radius: ", _
    "Variable Helix",10,50,50)

P(0) = InputBox("Select the initial Pitch: ", _
    "Variable Helix", 4,50,50)

P(1) = InputBox("Select the final Pitch: ", _
    "Variable Helix", 4,50,50)

NumPoints = Turns*PointsPerTurn

'

' Initialize the points and segments arrays.

'

Redim points(NumPoints+1)
Redim segments(NumPoints)
points(0) = "NAME:PolylinePoints"
segments(0) = "NAME:PolylineSegments"
'

'

' Build the Point and Segment Arrays needed in the HFSS polyline call.

'

For n = 1 To (NumPoints+1)
    Angle = (n-1)*2*Pi/PointsPerTurn
    Radius = R(0) + ((n-1)/NumPoints)*(R(1)-R(0))
    Pitch = P(0) + ((n-1)/NumPoints)*(P(1)-P(0))
    Rise = (n-1)*Pitch/PointsPerTurn

    XValue = cstr(Radius*cos(Angle)) & Units
    YValue = cstr(Radius*sin(Angle)) & Units
    ZValue = cstr(Rise) & Units
    points(n) = Array("NAME:PLPoint", "X:=", XValue, "Y:=", _
        YValue, "Z:=", ZValue)
'
```

```
' Create the line segments between each of the pairs of points.  
,  
  
If n<=NumPoints Then  
  
segments(n) = Array("NAME:PLSegment", "SegmentType:=", _  
"Line", "StartIndex:=", (n-1), "NoOfPoints:=", 2)  
  
End If  
  
Next  
,  
  
' Create the polyline.  
,  
  
oEditor.CreatePolyline _  
  
Array("NAME:PolylineParameters", "IsPolylineCovered:=", true, _  
"IsPolylineClosed:=", false, points, segments), _  
Array("NAME:Attributes", "Name:=", "Line_Helix", "Flags:=", _  
"", "Color:=", "(132 132 193)", "Transparency:=", 0.4, _  
"PartCoordinateSystem:=", "Global", "MaterialName:=", _  
"vacuum", "SolveInside:=", true)  
,  
  
' Create the helix cross-section.  
,  
  
oEditor.CreateCircle _  
  
Array("NAME:CircleParameters", "IsCovered:=", true, "XCenter:=", _  
cstr(R(0))&Units, "YCenter:=", 0, "ZCenter:=", 0, "Radius:=", _  
"1"&Units, "WhichAxis:=", "Y"), _  
Array("NAME:Attributes", "Name:=", "Circle_Helix", "Flags:=", _  
"", "Color:=", "(132 132 193)", "Transparency:=", 0.4, _  
"PartCoordinateSystem:=", "Global", "MaterialName:=", "vacuum", _  
"SolveInside:=", true)  
,  
  
' Sweep the cross-section along the path.
```

```
,
```

```
oEditor.SweepAlongPath _
```

```
    Array("NAME:Selections", "Selections:=", _
```

```
        "Circle_Helix,Line_Helix"),
```

```
    Array("NAME:PathSweepParameters", "DraftAngle:=", "0deg", _
```

```
        "DraftType:=", "Round", "TwistAngle:=", "0deg")
```

## Data Export Script

The following sample script demonstrates how to export data and save it to a file. The output data in the example script is in 3 columns. The first column is frequency in GHz, the second is the Real part of S11, and the third is the Img part of S11. It uses a tab-delimited format. The output is done using output variables.

The frequency sweep data must be entered correctly. If it is incorrect, the script will request a freq point that does not exist and execution will stop.

The script includes comment lines, which are preceded by an apostrophe (') and offer explanations for each subsequent line or lines.

```
Dim oAnsoftApp
```

```
Dim oDesktop
```

```
Dim oProject
```

```
Dim oDesign
```

```
Dim oEditor
```

```
Dim oModule
```

```
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
```

```
Set oDesktop = oAnsoftApp.GetAppDesktop()
```

```
set oProject = oDesktop.GetActiveProject
```

```
set oDesign = oProject.GetActiveDesign()
```

```
Dim oFS,ofile,x,y,z,path,range,
```

```
Dim arr2,del_f,freq,cfreq,val,temp,stn,stw,i,line
```

```
'
```

```
    ' Input the desired file name.
```

```
'
```

```
path = inputbox("Input the file name" & chr(13) & _
```

```
"Note: If you do not specify a path the file will " & _
```

---

```
"be placed in the script directory", _  
"File","C:\hfss_export.txt",50,50)  
  
'  
' If the user clicks Cancel the path will be blank, in which case  
the script should just exit.  
If path <>"" then  
  
'  
' Create the file, open it for data entry, and output the column  
labels.  
  
'  
Set oFS = CreateObject("Scripting.FileSystemObject")  
Set ofile = oFS.CreateTextFile (path)  
line = "Freq" & chr(9) & "RE(S11)" & chr(9) & "IMG(S11)"  
ofile.WriteLine line  
  
'  
' Input the needed freq, solution, and sweep data and clean it  
up.  
  
'  
msgbox("For the following input make sure it matches " & _  
"the frequencies defined in your sweep")  
range = inputbox("Input the range of frequencies in GHz " & _  
"and number of points",_  
"Frequency","8,12,10",50,50)  
  
'  
' The following 2 lines define the 2 output variables.  
  
'  
oDesign.CreateOutputVariable "re_S", "re(S(port1,port1))"  
oDesign.CreateOutputVariable "im_S", "im(S(port1,port1))"  
arr = split (range, ",")  
arr(0) = Trim(arr(0))  
arr(1) = Trim(arr(1))
```

```
arr(2) = Trim(arr(2))

if cint(arr(2)) <> 1 then
    del_f = (arr(1)-arr(0))/(arr(2)-1)
else
    del_f = 0
end if

temp = InputBox("Input the Setup and Sweep number to use:"_
& chr(13) & "(e.g. input 1,2 for Setup1 and Sweep2)", _
"Solution Data","1,1",50,50)

arr2 = split(temp,",")
stn = arr2(0)
swn = arr2(1)

stn = Trim(stn)
swn = Trim(swn)

'

' Loop through the freq points.

'

for i=1 to arr(2) step 1
    freq = arr(0) + (cint(i)-1)*del_f
    x=freq
    cfreq="Freq=""" & freq & "Ghz"""

'

' Get the values of the output variables for the desired freq.

'

val = oDesign.GetOutputVariableValue("re_S","Setup" & _
stn & " :Sweep" & swn,cfreq, "")
y = val

val = oDesign.GetOutputVariableValue("im_S","Setup" & _
stn & " : Sweep" & swn,cfreq, "")
z = val
```

```
' Create the line of text to send to the file and write it to the
file.

'

line = x & chr(9) & y & chr(9) &z
ofile.WriteLine line

Next

'

' Delete the 2 output variables before finishing.

'

oDesign.DeleteOutputVariable "re_S"
oDesign.DeleteOutputVariable "im_S"

'

' Close the file.

'

ofile.close

End if
```

## IronPython Example Scripts

IronPython Examples:

- [BH Coordinates Python Script](#)
- [Equation Based Curve Python Script](#)

### BH Coordinates Python Script

This sample Python script adds a material ("Posco 35PN250") with BH coordinates from a specified file (Posco\_BH\_curve.tab):

0	0.000
10	0.050
20	0.130
23	0.152
25	0.175
28	0.202
30	0.229
33	0.257
35	0.287
38	0.318
40	0.356
43	0.395
45	0.435
48	0.477
52	0.523
56	0.574
60	0.626
66	0.683
72	0.740
78	0.795
87	0.850
97	0.906
112	0.964
130	1.024
151	1.084
175	1.140
200	1.189
225	1.228

To recreate this tab file, paste [the text below the script](#) into a text editor and save as Posco\_BH\_curve.tab.

The script itself includes comment lines, which are preceded by # and offer explanations for the subsequent line(s).

### Script Contents

```
# specify path to the file with BH coordinates
path_to_file = r"D:\Posco_BH_curve.tab"

# specify name of the material
material_name = "Posco 35PN250"

oProject = oDesktop.GetActiveProject()
oDefinitionManager = oProject.GetDefinitionManager()

""" create list with B and H points to pass to AddMaterial command
bh_coordinates list is a three dimensional array: 1D: name, 2D:
Coordinate list, 3D: BH point"""

bh_coordinates = ["NAME:BHCoordinates", ["NAME:DimUnits", "", ""]]
```

```
with open(path_to_file) as input_file:  
    for line in input_file:  
        h, b = line.split()  
  
        bh_coordinates.append(["NAME:Coordinate", [  
            "NAME:CoordPoint",  
            float(h),  
            float(b)  
        ]])  
  
    # create a new magnetic material with BH curve  
    oDefinitionManager.AddMaterial(  
        [  
            "NAME:" + material_name,  
            "CoordinateSystemType:=", "Cartesian",  
            "BulkOrSurfaceType:=", 1,  
            [  
                "NAME:PhysicsTypes",  
                "set:=", ["Electromagnetic"]  
            ],  
            [  
                "NAME:permeability",  
                "property_type:=", "nonlinear",  
                "BTypeForSingleCurve:=", "normal",  
                "HUnit:=", "A_per_meter", # unit can be specified as  
                variable  
                "BUnit:=", "tesla", # unit can be specified as variable  
                "IsTemperatureDependent:=", False,  
                bh_coordinates,  
                [  
            ]  
        ]  
    )
```

```
"NAME:Temperatures"  
]  
],  
"conductivity:=" , "1818181.82",  
[  
    "NAME:magnetic_coercivity",  
    "property_type:=" , "VectorProperty",  
    "Magnitude:=" , "0A_per_meter",  
    "DirComp1:=" , "1",  
    "DirComp2:=" , "0",  
    "DirComp3:=" , "0"  
]  
])
```

### **Posco\_BH\_Curve.tab Contents**

0	0.000
10	0.050
20	0.130
23	0.152
25	0.175
28	0.202
30	0.229
33	0.257
35	0.287
38	0.318
40	0.356
43	0.395
45	0.435
48	0.477
52	0.523
56	0.574

60	0.626
66	0.683
72	0.740
78	0.795
87	0.850
97	0.906
112	0.964
130	1.024
151	1.084
175	1.140
200	1.189
225	1.228
252	1.258
282	1.283
317	1.304
357	1.324
402	1.343
450	1.360
500	1.375
550	1.387
602	1.398
657	1.407
717	1.417
784	1.426
867	1.437
974	1.448
1117	1.461
1299	1.478
1515	1.495
1752	1.513

---

2000	1.529
2253	1.543
2521	1.557
2820	1.570
3167	1.584
3570	1.600
4021	1.617
4503	1.634
5000	1.652
5503	1.669
6021	1.685
6570	1.701
7167	1.717
7839	1.733
8667	1.749
9745	1.769
11167	1.793
12992	1.820
15146	1.851
17518	1.882
20000	1.909
22552	1.931
25417	1.948
28906	1.961
33333	1.971
38932	1.981

## Equation Based Curve Python Script

This sample Python script creates an equation based curve that produces a helix.

```
from math import pi, sin, cos
```

```
oProject = oDesktop.GetActiveProject()
oDesign = oProject.GetActiveDesign()
oEditor = oDesign.SetActiveEditor("3D Modeler")

Start_t = 0
End_t = pi*2

Npoint = 128
Nsection = Npoint-1

d_t = (End_t-Start_t)/Nsection

for n in range(1,Nsection):

    P1 = Start_t+d_t*(n-1)
    P2 = P1+d_t
    X_t1 = cos(P1*6)
    Y_t1 = sin(P1*6)
    Z_t1 = P1
    X_t2 = cos(P2*6)
    Y_t2 = sin(P2*6)
    Z_t2 = P2

oEditor.CreatePolyline(
[
    "NAME:PolylineParameters",
    "IsPolylineCovered:=" , True,
    "IsPolylineClosed:=" , False,
    [

```

```
"NAME:PolylinePoints",
[

"NAME:PLPoint",
"X:=" , '1mm*' + str(X_t1),
"Y:=" , '1mm*' + str(Y_t1),
"Z:=" , '1mm*' + str(Z_t1)

],


[

"NAME:PLPoint",
"X:=" , '1mm*' + str(X_t2),
"Y:=" , '1mm*' + str(Y_t2),
"Z:=" , '1mm*' + str(Z_t2)

]

],


[

"NAME:PolylineSegments",
[

"NAME:PLSegment",
"SegmentType:=" , "Line",
"StartIndex:=" , 0,
"NoOfPoints:=" , 2

]

],


[

"NAME:PolylineXSection",
"XSectionType:=" , "None",
"XSectionOrient:=" , "Auto",
"XSectionWidth:=" , "0mm",
"XSectionTopWidth:=" , "0mm",
"XSectionHeight:=" , "0mm",
```

```
"XSectionNumSegments:=" , "0",
"XSectionBendType:=" , "Corner"
]
],
[
"NAME:Attributes",
"Name:=" , "Polyline"+str(n),
"Flags:=" , "",
"Color:=" , "(132 132 193)",
"Transparency:=" , 0,
"PartCoordinateSystem:=", "Global",
"UDMId:=" , "",
"MaterialValue:=" , "\"vacuum\"",
"SurfaceMaterialValue:=" , "\"\"",
"SolveInside:=" , True,
"IsMaterialEditable:=" , True,
"UseMaterialAppearance:=" , False,
"IsLightweight:=" , False
])
```

# Index

**3**

3D Modeler editor commands 10-1  
AssignMaterial 10-142  
Chamfer 10-145  
ChangeProperty 10-262  
Connect 10-147  
Copy 10-120  
CoverLines 10-148  
CoverSurfaces 10-149  
CreateBondwire 10-10  
CreateBox 10-14  
CreateCircle 10-16  
CreateCone 10-19  
CreateCutplane 10-22  
CreateCylinder 10-24  
CreateEllipse 10-26  
CreateEntityList 10-150  
CreateEquationCurve 10-29  
CreateEquationSurface 10-33  
CreateFaceCS 10-152, 10-158  
CreateGroup 10-157  
CreateHelix 10-36  
CreateObjectFromEdges 10-165  
CreateObjectFromFaces 10-166  
CreatePoint 10-38

CreatePolyline 10-47  
CreateRectangle 10-53  
CreateRegion 10-55  
CreateRegularPolygon 10-62  
CreateRegularPolyhedron 10-59  
CreateRelativeCS 10-168  
CreateSphere 10-65  
CreateSpiral 10-67  
CreateTorus 10-69  
CreateUserDefinedModel 10-72  
CreateUserDefinedPart 10-40, 10-79  
Delete 10-268  
DeleteEmptyGroups 10-170  
DeleteLastOperation 10-171  
DeletePolylinePoint 10-121  
DetachFaces 10-172  
DuplicateAlongLine 10-122  
DuplicateAroundAxi 10-125  
DuplicateMirror 10-128  
Edit3dComponent 10-89  
EditEntityList 10-173  
EditFaceCS 10-175  
EditObjectCS 10-181  
EditPolyline 10-92

EditRelativeCS 10-188  
Export 10-190  
ExportModelImageToFile 10-192  
Fillet 10-195  
FlattenGroup 10-197  
GenerateHistory 10-197  
Get3DComponentParameters  
    10-97  
GetActiveCoordinateSystem 10-  
    203  
GetBodyNamesByPosition 10-  
    269  
GetCoordinateSystems 10-204  
GetEdgeByPosition 10-271  
GetEdgeIDsfromFace 10-272  
GetEdgeIDsfromObject 10-273  
GetFaceArea 10-273  
GetFaceByPosition 10-275  
GetFaceCenter 10-274  
GetFaceIDs 10-277  
GetGeometryModelerMode 10-  
    277  
GetModelBoundingBox 10-278  
GetProperties 7-26  
GetPropertyValue 7-28, 12-61  
 GetUser Position 10-285  
GetVertexIDsFromEdge 10-286  
GetVertexIDsFromFace 10-286  
GetVertexIDsFromObject 10-  
    287  
GetVertexPosition 10-288  
HealObject 10-199  
Import 10-204  
ImportDXF 10-208  
ImportGDSII 10-212  
Insert3DComponent 10-100  
InsertNativeComponent 10-101  
Intersect 10-216  
Mirror 10-131  
Move 10-133  
MoveCSToEnd 10-217  
MoveEntityToGroup 10-218  
MoveFaces 10-219  
OffsetFaces 10-134  
PageSetup 10-289  
ProjectSheet 10-221  
RenamePart 10-291  
Rotate 10-137  
Scale 10-138  
Section 10-223  
SeparateBody 10-225  
SetModelUnits 10-226  
SetWCS 10-227  
Split 10-229  
Subtract 10-232  
SweepAlongPath 10-107  
SweepAlongVector 10-109  
SweepAroundAxis 10-112  
SweepFacesAlongNormal 10-114, 10-  
    114, 10-233, 10-233

Sweep-	AddDynamicNPortData 27-51
FacesAlongNormalWithAttributes 10-116	AddLevel 28-3
ThickenSheet 10-235	AddMarker 12-9
UncoverFaces 10-237	AddMarkerToPlot 18-2
Ungroup 10-239	AddMaterial 27-2, 27-12
Unite 10-238	AddMessage 3-5
UpdateComponentDefinition 10-119	AddNamedExpr 19-3
WrapSheet 10-240	AddNote 12-10
3D Modeler editor object commands	AddNPortData 27-54
GetNumObjects 10-284	AddPinGrounds 26-48
GetObjectIDByName 10-279	AddPinIPorts 26-51
GetObjectName 10-279	AddPinPageConnectors 26-52
GetObjectNameByFaceID 10-280	AddReduceOp 21-2
	AddTerminalsToWinding 13-45
	AddTraceCharacteristics 12-12
	AddTraces 12-13
	AlignHorizontal 26-53, 28-4
	AlignVertical 26-54, 28-5
Analysis module commands	
	Analyze 9-6
	CopySetup 15-2, 16-7
	DeleteSetups 15-3
	EditSetup 15-4
	ExportIcepak 15-15
	ExportSolnData 15-15
	GetSetupCount 15-16
	GetSetups 15-17
	GetSweepCount 15-18
	GetSweeps 15-18

InsertSetup [Maxwell]	15-19	AssignCurrent	13-50
PasteSetup	15-33	AssignCurrentDensity	13-51
PasteSweep	15-33	AssignCurrentDensityGroup	13-52
RenameSetup	15-34	AssignCurrentDensityTerminal	13-52
ResetAllToTimeZero	15-34	AssignCurrentDensityTerminalGroup	13-52, 13-53
ResetSetupToTimeZero	15-35	AssignCurrentGroup	13-53
ResetToTimeZero	9-42	AssignCylindricalGapOp	14-5
RevertAllToInitial	15-35	AssignCylindricalHField	13-19
RevertAllToInitialTemperature	15-36	AssignDependent	13-20
RevertSetupToInitial	15-36	AssignFloating	13-53
RevertSetupToInitialTemperature	15-37	AssignFluxTangential	13-23
Analysis Setup Module Script Commands	15-1	AssignForce	21-4
Analyze		AssignImpedance	13-23
Analysis module command	9-6	AssignIndependent	13-24
AnalyzeAll	9-6	AssignInsulating	13-26
AnalyzeDistributed	9-7	AssignLayoutForce	21-5
Ansoft Application Object commands	2-1	AssignLengthOp	14-12
GetAppDesktop	2-2	AssignMaterial	10-142
ApplyMeshOps	9-5	AssignMatrix	21-7
arithmetic operators	1-6	AssignModelResolutionOp	14-15
array variables	1-4	AssignRadiation	13-26
AssignBand C		AssignResistiveSheet	13-27
AssignCharge	13-46	AssignSink	13-54
AssignCoilGroup	13-47	AssignSkinDepthLayerSetting	14-17
AssignCoilTerminal	13-48	AssignSkinDepthOp	14-18
AssignCoilTerminalGroup	13-48	AssignSymmetry	13-28
		AssignTangentialHField	13-28
		AssignThinLayer	13-29

---

AssignTorque	21-9	AssignCurrentGroup	13-53
AssignTouching	13-31	AssignCylindricalHField	13-19
AssignTrueSurfOp	14-19	AssignDependent	13-20
AssignVectorPotential	13-32	AssignFloating	13-53
AssignVoltage	13-54	AssignImpedance	13-23
AssignVoltageAPhi	13-55	AssignIndependent	13-24
AssignVoltageDrop	13-56	AssignInsulating	13-26
AssignVoltageDropGroup	13-57	AssignRadiation	13-26
AssignVoltageGroup	13-57	AssignSink	13-54
AssignVolumeChargeDensity	13-57	AssignSymmetry	13-28
AssignWindingGroup	13-58	AssignTangentialHField	13-28
AssignZeroTangentialHField	13-33	AssignVoltage	13-54
<b>B</b>			
Boundary/Excitation module commands		AssignVoltageDrop	13-56
AddAssignmentToBoundary	13-3	AssignVoltageDropGroup	13-57
AddTerminalsToWinding	13-45	AssignVoltageGroup	13-57
AssignCharge	13-46	AssignVolumeChargeDensity	13-57
AssignCoilGroup	13-47	AssignWindingGroup	13-58
AssignCoilTerminal	13-48	AssignZeroTangentialHField	13-33
AssignCoilTerminalGroup	13-48	DeleteAllBoundaries	13-4
AssignCurrentDensity	13-51	DeleteAllExcitations	13-5
AssignCurrentDensityGroup	13-52	DeleteBoundaries	13-5
AssignCurrentDensityTerminal	13-52	EditCharge	13-59
AssignCur-		EditCoilTerminal	13-59
rentDensityTerminalGroup		EditCurrentDensity	13-61
13-52, 13-53		EditCurrentDensityTerminal	13-61
		EditCylindricalHField	13-34
		EditDependent	13-34

EditExternalCircuit 13-61  
EditFloating 13-62  
EditImpedance 13-35  
EditIndependent 13-36  
EditInsulating 13-37  
EditRadiation 13-37  
EditSink 13-62  
EditSymmetry 13-38  
EditTangentialHField 13-40  
EditVoltage 13-62  
EditVoltageDrop 13-65  
EditVolumeChargeDensity 13-65  
EditWindingGroup 13-66  
EditZeroTangentialHField 13-43  
GetBoundaries 13-7  
GetBoundariesOfType 13-7  
GetBoundaryAssignment 13-6  
GetCoreLossEffect 13-8  
GetCoreLossEffectOnField 13-9  
GetDisplacementCurrent 13-9  
GetEddyEffect 13-9  
GetExcitations 13-10  
GetExcitationsOfType 13-11  
GetNumBoundaries 13-11  
GetNumBoundariesOfType 13-12  
GetNumExcitations 13-13  
GetNumExcitationsOfType 13-13  
ReassignBoundaries 13-3, 13-14  
ReassignBoundary 13-14  
RemoveAssignmentFromBoundary 13-15  
RenameBoundary 13-16  
ReprioritizeBoundaries 13-17  
SetCoreLoss 13-66  
SetEddyEffect 13-66  
SetMagnetizationCompute 13-67  
SetMinimumTimeStep 13-69  
boundary/excitation script commands 13-2  
BreakUDMConnection 10-261  
BringToFront 27-160, 28-6

**C**

CalcOp 19-4  
CalcRead (deprecated) 19-4  
CalcStack 19-6  
CalculatorRead 19-5  
CalculatorWrite 19-7  
Cascade 24-5  
Chamfer 10-145  
ChangeGeomSettings 19-8  
ChangeProperty 10-262, 12-16, 26-105, 28-6  
ClcEval 19-9  
ClcMaterial 19-10  
ClearAllMarkers 12-20  
ClearAllNamedExpr 19-11  
ClearAllTraceCharacteristics 12-20

ClearDiffPairs 24-6  
ClearMessages 3-6, 5-4  
ClearSolutionCache 27-60  
Clone 24-7  
CloneMaterial 27-9  
Close 5-4, 24-8  
CloseAllWindows 3-7  
CloseEditor 26-57, 28-19  
CloseProject 3-8  
CloseProjectNoForce 3-8  
Combine 24-9  
comparison operators 1-7  
ComplInstance Functions 25-2  
ComplInstance Script Commands 25-1  
Component Manager Script Commands 27-31  
ComputeCoreLossCoefficients 27-10  
conditional statements  
    If...Then... Else 1-8  
    Select Case 1-8  
    types of 1-8  
Con-  
    fig-  
    ureFlu-  
    entConductivityCoupling 9-8  
Connect 10-147  
ConstructVariationString 9-9  
conventions  
    scripting help 1-1  
                converting data types 1-10  
                ConvertToDynamic 27-125  
                ConvertToParametric 27-126  
                Copy 10-120, 26-57, 28-20  
                CopyData 26-58  
                CopyDesign 5-5  
                CopyNamedExprToStack 19-11  
                CopyReportDefinition 12-23  
                CopyReportsData 12-22  
                Copyright and Trademark Information 2  
CopySetup  
    Analysis module command 15-2, 16-7  
    Optimetrics module command 15-2, 16-7  
CopySubdesign 26-59  
CopyTraceDefinitions 12-23  
CopyTracesData 12-21  
Core Global Script Context Commands 29-1  
Count 3-9  
CoverLines 10-148  
CoverSurfaces 10-149  
CPython 1-70  
Create3DComponent 10-6  
Create3DDesign 9-11  
CreateArc 28-20  
CreateBondwire 10-10  
CreateBox 10-14

CreateCircle 10-16, 28-21  
CreateCone 10-19  
CreateCutplane 10-22  
CreateCylinder 10-24  
CreateEllipse 10-26  
CreateEntityList 10-150  
CreateEquationCurve 10-29  
CreateEquationSurface 10-33  
CreateFaceCS 10-152, 10-158  
CreateFieldPlot 18-4  
CreateGroup 10-157  
CreateHelix 10-36  
CreateImage 28-22  
CreateLine 28-23  
CreateObjectFromEdges 10-165  
CreateObjectFromFaces 10-166  
CreateOutputVariable 11-1  
CreatePage 26-60  
CreatePin 28-24  
CreatePoint 10-38  
CreatePolygon 28-25  
CreatePolyline 10-47  
CreateRectangle 10-53, 28-26  
CreateRegion 10-55  
CreateRegularPolygon 10-62  
CreateRegularPolyhedron 10-59  
CreateRelativeCS 10-168  
CreateReport 12-25  
CreateReportFromFile 12-24  
CreateReportFromTemplate 12-35  
CreateReportofAllQuantities 12-36  
CreateSphere 10-65  
CreateSpiral 10-67  
CreateText 28-27  
CreateTorus 10-69  
CreateUserDefinedModel 10-72  
CreateUserDefinedPart 10-40, 10-79  
CreateUserDefinedSolution 23-1  
Cut 26-61, 28-27  
CutDesign 5-6

**D**

dataset commands  
    AddDataset 8-1  
    DeleteDataset 8-4  
    EditDataset 8-5  
    ImportDataset 8-7  
Dataset Script Commands 8-1  
DeactivateOpen 26-62  
DeactivateShort 26-63  
Deembed 24-10  
DeembedBack 24-11  
DeembedFront 24-13  
Definition Editor Script Commands 28-1  
Definition Manager commands  
    AddDefinitionFromBlock 10-244  
    AddDefinitionFromLibFile 10-249  
    GetProjectMaterialNames 27-27

Definition Manager Script Commands	27-1	Optimetrics module command	16-8
Delete	10-268, 26-64, 28-28	DeleteSolutionVariation	17-1
DeleteAllBoundaries	13-4	Deletetraces	12-39
DeleteAllExcitations	13-5	DeleteUserDefinedSolutions	23-3
DeleteAllParameters	21-10	DeleteVariation	9-14
DeleteAllReports	12-38	Design object commands	12-49, 12-50, 12-51, 12-52, 12-52, 12-65
DeleteBoundaries	13-5	AddCartesianLimitLine	12-4
DeleteDataset	8-4	AddCartesianXMarker	12-6
DeleteDesign	5-7	AddCartesianYMarker	12-7
DeleteEmptyGroups	10-170	AddDeltaMarker	12-8
DeleteFieldPlot	18-35	AddMarker	12-9
DeleteFieldVariation	9-12	AddMarkerToPlot	18-2
DeleteFullVariation	9-13	AddNote	12-10
DeleteLastOperation	10-171	AddTraceCharacteristics	12-12
DeleteLinkedDataVariation	9-13	AddTraces	12-13
DeleteMarker	12-37	AnalyzeAll	9-6
DeleteMotionSetup	A	AnalyzeDistributed	9-7
DeleteNamedExpr	19-12	ApplyMeshOps	9-5
DeleteOp	14-2	CalculatorRead	19-5
DeleteOutputVariable	11-3	ClearAllMarkers	12-20
DeleteParameters	21-10	ClearAllTraceCharacteristics	12-20
DeletePolylinePoint	10-121	CloseAllWindows	3-7
DeleteProject	3-11	CopyReportDefinition	12-23
DeleteReduceMatrix	21-11	CreateOutputVariable	11-1
DeleteReduceOp	21-11	CreateReport	12-25
DeleteReport	12-39	CreateReportFromTemplate	12-35
DeleteSetups	15-3	DeleteAllReports	12-38
Analysis module command			
15-3			

DeleteOutputVariable	11-3	GetRegistryString	3-71
DeleteReport	12-39	GetSelections	10-284
DeleteTraces	12-39	GetSolutionType	9-38
DoesOutputVariableExist	11-3	GetTempDirectory	3-33
EditOutputVariable	11-4	GetVariables	7-30
ExportConvergence	9-25	GetVariableValue	7-30
ExportElementBasedHarmonicForce	9-26	GetVariationVariableValue	9-38
ExportMeshStats	9-28	GetVersion	3-35
ExportPlotImageToFile	12-41	ImportIntoReport	12-71
ExportPlotImageWithViewToFile	18-40	Is2D	9-40
ExportProfile	9-29	Is3D	9-40
ExportReport	12-43	PasteReports	12-76
ExportToFile	12-46	Redo	9-40
GetDisplayType	12-59	RenameDesignInstance	9-41
GetLibraryDirectory	3-22	RenameReport	12-78
GetMatchedObjectName	10-282	RenameTraces	12-79
GetModelUnits	10-283	RunToolkit	9-62
GetModule	9-34	SavePlotSettingsAsDefault	12-80
GetName	9-35, 24-30	SetActiveEditor	9-42
GetObjectsByMaterial	10-281	SetConductivityThreshold	9-43
GetObjectsInGroup	10-281	SetDesignSettings	9-44
GetOutputVariableValue	11-6	SetLibraryDirectory	3-62
GetProjectDirectory	3-29	SetMPIVendor	2-2
GetProperties	7-26	SetProjectDirectoryVBCommand>	3-62
GetPropertyValues	7-28, 12-61	SetPropertyValue	7-31
GetRegistryInt	3-70	SetSolutionType	9-60
		SetTempDirectory	3-64
		SetupYConnection	13-69

SetVariableValue	7-33	GetName	3-26
ShowWindow	10-228	GetProjectList	3-30
Solve	9-60, 9-63	GetProjects	3-25, 3-29
StopSimLink	9-61	GetSchematicEnvironment	3-31
UpdateAllReports	12-81	LaunchJobMonitor	3-40
ValidateDesign	9-64	NewProject	3-40
Design Object Script Commands	9-1	OpenMultipleProjects	3-42
Desktop Commands For Registry Values	3-68	OpenProject	3-43
Desktop object commands		PauseRecording	3-45
AddMessage	3-5	PauseScript	3-46
ClearMessages	3-6, 5-4	Print	3-47
CloseProject	3-8	QuitApplication	3-47
CloseProjectNoForce	3-8	RefreshJobMonitor	3-48
Count	3-9	ResetLogging	3-49
DeleteProject	3-11	RestoreWindow	3-51
EnableAutosave	3-11	ResumeRecording	3-51
ExportOptionsFiles	3-12	RunProgram	3-53
GetActiveProject	3-13	RunScript	3-54
GetAutosaveEnabled	3-14	SelectScheduler	3-56
GetBuildTimeDateString	3-14	SetActiveProject	3-58
GetCustomMenuSet	3-15	SetActiveProjectByPath	3-59
GetDesigns	3-18	SetCustomMenuSet	3-59
GetDesktopConfiguration	3-19	SetDesktopConfiguration	3-61
GetDistributedAnalysisMachines		SetSchematicEnvironment	3-63
3-20		Sleep	3-65
GetDis-		SubmitJob	3-66
trib-		TileWindows	3-67
utedAna-		Desktop Object Script Commands	3-1
lysisMachinesForDesignType		Desktop Scripting Conventions	1-82
3-20			

DetachFaces 10-172	EditFluxTangential_ 13-35
DisableDiffPairs 24-14	EditForce 21-12
DisplayPinNames 28-29	EditImpedance 13-35
DistributedAnalyzeSetup, Optimetrics module command 16-8	EditIndependent 13-36
DoesOutputVariableExist 11-3	EditInsulating 13-37
Draw Menu commands 10-4	EditLayoutForce 21-12
DuplicateAlongLine 10-122	EditLengthOp 14-27
DuplicateAroundAxis 10-125	EditMaterial 27-13
DuplicateMirror 10-128	EditMatrix 21-15
<b>E</b>	EditModelResolutionOp 14-30
Edit 27-60, 27-126, 27-160, 27-199, 27-228	EditMotionSetup E
Edit Menu Commands 10-120	EditNotes 9-14
Edit3DComponent 10-87, 10-89	EditObjectCS 10-181
EditCharge 13-59	Editor object commands
EditCoilTerminal 13-59	SetPropertyValue 7-31
EditCurrent 13-59	EditScripting IDs 26-4
EditCurrentDensity 13-61	EditOutputVariable 11-4
EditCurrentDensityTerminal 13-61	EditPolyline 10-92
EditCylindricalGapOp 14-22	EditRadiation 13-37
EditCylindricalHField 13-34	EditReduceOp 21-15
EditDataset 8-5	EditRelativeCS 10-188
EditDependent 13-34	EditResistiveSheet 13-37
EditEntityList 10-173	EditSetup
EditExternalCircuit 13-61	Analysis module command 15-4
EditFaceCS 10-175	optimization command 16-26
EditFloating 13-62	parametric command 16-19
	sensitivity command 16-42
	statistical command 16-54

EditSink 13-62	EnterQty 19-16
EditSkinOp 14-31	EnterScalar 19-16
EditSolverOnDemandModel 27-87	EnterScalarFunc 19-17
EditSurfaceMeshSummaryData 18-36	EnterSurf 19-18
EditSymmetry 13-38	EnterVector 19-18
EditTangentialHField 13-40	EnterVectorFunc 19-19
EditThinLayer 13-40	EnterVol 19-20
EditTorque 21-16	Event Callback Scripting 1-86
EditTouching 13-41	Example Scripts 30-1
EditTrueSurfOp 14-33	Export 10-190, 27-99, 27-138, 27-172, 27-217, 27-243
EditUserDefinedSolution 23-4	ExportCircuit 15-10
EditVectorPotential 13-42	ExportCitiFile 24-16
EditVoltage 13-62	ExportConvergence 9-25
EditVoltageAPhi 13-64	ExportDXConfigFile, Optimetrics module command 16-9
EditVoltageDrop 13-65	ExportFile 28-29
EditVolumeChargeDensity 13-65	ExportFullWaveSpice 27-144
EditWindingGroup 13-66	ExportHarmonicTransientForce 9-27
EditWithComps 27-87, 27-126, 27-160, 27-199, 27-235	ExportIcepak, Analysis module command 15-14
EditZeroTangentialHField 13-43	ExportImageToFile 12-40
ElectricRuleCheck 26-66	ExportMarkerTable 12-47
EnableAutoSave 3-11	ExportMaterial 27-27
EnableDiffPairs 24-15	ExportMatlab 24-18
EnableHarmonicForceCalculation 9-15	ExportMesh Stats 9-28
EnterComplex 19-13	ExportNetlist 26-68
EnterComplexVector 19-13	ExportOptimetricsProfile, Optimetrics module command 16-10
EnterLine 19-14	ExportOptimetricsResults, Opti-
EnterPoint 19-15	

- metrics module command 16-11  
ExportOptionsFiles 3-12  
ExportParametricResults, Opti-metrics module command 16-12  
ExportPlotImageToFile 12-41  
ExportPlotImageWithViewToFile 18-40  
ExportProfile 9-26, 9-29  
ExportReport 12-43  
ExportScript 27-250  
ExportSolnData, Analysis module command 15-15  
ExportSpreadsheet 24-20  
ExportSurfaceMeshSummary 18-41  
ExportToFile 12-44  
ExportTouchstone 24-22  
ExportTouchstone2 24-24  
Extract 24-27
- F**
- FFTONReport 12-47  
Field Overlay module commands, GetFieldPlotNames 18-42  
field overlay script commands 18-1  
Field Overlays module commands  
    AddNamedExpr 19-3  
    CalcOp 19-4  
    CalcStack 19-6  
    ChangeGeomSettings 19-8
- ClcMaterial 19-10  
ClearAllNamedExpr 19-11  
CopyNamedExprToStack 19-11  
CreateFieldPlot 18-4  
DeleteFieldPlot 18-35  
DeleteNamedExpr 19-12  
EditSurfaceMeshSummaryData 18-36  
EnterComplex 19-13  
EnterComplexVector 19-13  
EnterLine 19-14  
EnterPoint 19-15  
EnterQty 19-16  
EnterScalar 19-16  
EnterScalarFunc 19-17  
EnterSurf 19-18  
EnterVector 19-18  
EnterVectorFunc 19-19  
EnteVol 19-20  
ExportSurfaceMeshSummary 18-41  
ModifyFieldPlot 18-43  
RenameFieldPlot 18-47  
RenamePlotFolder 18-48  
SetFieldPlotSettings 18-48  
SetPlotFolderSettings 18-51  
SetPlotsViewSolutionContext 18-57  
UpdateAllFieldsPlots 18-58  
UpdateQuantityFieldsPlots 18-58

Fields Calculator commands	FindElements 26-69
AddNamedExpr 19-2, 19-3	FitToBorder 26-71
CalcOp 19-4	FlattenGroup 10-197
CalcStack 19-6	FlipHorizontal 26-75, 28-30
CalculatorWrite 19-7	FlipVertical 26-76, 28-31
ChangeGeomSettings 19-8	For...Next loop 1-9
ClcEval 19-9	functions
ClcMaterial 19-10	VBScript procedures 1-10
ClearAllNamedExpr 19-11	
CopyNamedExprToStack 19-11	<b>G</b>
DeleteNamedExpr 19-12	general method script commands 26-45
EnterComplex 19-13	GenerateHistory 10-197
EnterComplexVector 19-13	GenerateVariationData Parametric, parametric command 16-20
EnterLine 19-14	Get3DComponentDefinitionNames 10-97
EnterPoint 19-15	Get3DComponentInstanceNames 10-98
EnterQty 19-16	Get3DComponentMaterialNames 10-99
EnterScalar 19-16	Get3DComponentMaterialProperties 10-99
EnterScalarFunc 19-17	Get3DComponentParameters 10-97
EnterSurf 19-18	GetActiveCoordinateSystem 10-203
EnterVector 19-18	GetActiveDesign 5-8
EnterVectorFunc 19-19	GetActiveProject 3-13
EnterVol 19-20	GetAllCategories 12-49
ExportOnGrid 19-20	GetAllQuantities 12-50
ExportToFile 19-23	GetAllReportNames 12-48
Fields Calculator Script Commands 19-1	GetArrayVariables 7-26
Fields reporter module commands	
ExportMarkerTable 12-47	
Fillet 10-195	

GetAutoSaveEnabled 3-14  
GetAvailableDisplayTypes 12-51  
GetAvailableReportTypes 12-52  
GetAvailableSolutions 12-52  
GetBodyNamesByPosition 10-269  
GetBoundaries 13-7  
GetBoundariesOfType 13-7  
GetBoundaryAssignment 13-6  
GetBuildDateTimeString 3-14  
GetComponentName 25-2  
GetComponentPinLocation 26-118  
GetCoordinateSystems 10-204  
GetCoreLossEffect 13-8  
GetCoreLossEffectOnField 13-9  
GetData 27-100, 27-110, 27-139,  
27-173, 27-218, 27-244  
GetDataExpressions 12-53  
GetDataUnits 12-54  
GetDefinitionManager 5-9  
GetDependentFiles 5-10  
GetDesign 5-11, 5-11  
GetDesigns 3-18  
GetDesignValidationInfo 9-31  
GetDesignVariableNames 12-55  
GetDesignVariableUnits 12-56  
GetDesignVariableValue 12-57  
GetDesignVariationKey 12-58  
GetDisplacementCurrent 13-9  
GetDisplayType 12-59  
GetDistributedAnalysisMachines 3-20  
GetDis-  
trib-  
utedAna-  
lysisMachinesForDesignType 3-20  
GetDocumentNames 22-8  
GetEddyEffect 13-9  
GetEdgeByPosition 10-271  
GetEdgeIDsFromFace 10-272  
GetEdgeIDsFromObject 10-273  
GetEditorName [Schematic] 26-119  
GetEvaluatedPropertyValue 26-106  
GetExcitations 13-10  
GetExcitationsOfType 13-11  
GetExeDir 3-21  
GetFaceArea 10-273  
GetFaceByPosition 10-275  
GetFaceCenter 10-274  
GetFaceIDs 10-277  
GetFieldPlotNames, Field Overlay module  
command 18-42  
GetFrequencies 24-28  
GetFrequencyCount 24-29  
GetGeometryMode 9-32  
GetGeometryModelerMode 10-277  
GetImagDataValues 12-59  
GetInstanceID 25-3  
GetInstanceName 25-4  
GetLibraryDirectory 3-22  
GetMatchedObjectName 10-282

GetModelBoundingBox 10-278  
GetModelUnits 10-283  
GetModule 9-34  
GetMotionSetupNames G  
GetName 3-26, 5-13, 9-35, 24-30  
GetNames 27-101, 27-110, 27-140,  
27-174, 27-219, 27-245  
GetNetConnections 26-120  
GetNPortData 27-104  
GetNumBoundaries 13-11  
GetNumBoundariesOfType 13-12  
GetNumExcitations 13-13  
GetNumExcitationsOfType 13-13  
GetNumObjects 10-284  
GetNumPages 26-121  
GetObjectIDByName 10-279  
GetObjectName 10-279  
GetObjectNameByFacID 10-280  
GetObjectsByMaterial 10-281  
GetObjectsIngroup 10-281  
GetOperationNames 14-3  
GetOutputVariableValue 11-6  
GetParentDesign 25-4  
GetPath 5-13  
GetPer-  
    QuantityPrimarySweepValues  
        12-60  
GetPersonalLibDirectory 3-27  
GetPortCount 24-30  
GetPortNumber 24-31  
GetPostProcSettings 24-32  
GetProcessID 3-28  
GetProjectDirectory 3-29  
GetProjectList 3-30  
GetProjectMaterialNames 27-27  
GetProjects 3-25, 3-29  
GetProperties 7-26, 26-107, 27-112,  
28-31  
GetPropertyValue 7-28, 12-61, 28-32  
GetPropHost 25-5, 25-5  
GetPropServerName 25-5  
GetRealDataValues 12-63  
GetRegistryInt 3-70  
GetRegistryString 3-71  
GetReportTraceNames 12-64  
GetSelections 9-37, 10-284  
GetSetupCount 15-16  
GetSetupCount, Analysis module com-  
mand 15-16  
GetSetupNames (Optimetrics), Opti-  
metrics module command 16-  
13  
GetSetupNamesByType (Opti-  
metrics), Optimetrics module  
command 16-13  
GetSetups 15-17  
    Analysis module command 15-17  
GetSolutionContexts 12-64  
GetSolutionDataPerVariation 12-65  
GetSolutionType 9-38  
GetSolverOnDemandData 27-105

GetSolverOnDemandModelList 27-  
105

GetSweepCount 15-18

GetSweepCount, Analysis module  
command 15-18

GetSweepNames 12-68

GetSweeps 15-18

Analysis module command 15-  
18

GetSweepUnits 12-69

GetSweepValues 12-70

GetSymmetryMultiplier H

GetSysLibDirectory 3-33

GetTempDirectory 3-33

GetTopDesignList 5-14

GetTopEntryValue 19-26

GetUserLibDirectory 3-34

GetUserPosition 10-285

GetValidISolutionList 17-2

GetVariables 7-30

GetVariableValue 7-30

GetVariation 24-33

GetVariationVariableValue 9-38

GetVersion 3-35

GetVertexIDsFromEdge 10-286

GetVertexIDsFromFace 10-286

GetVertexIDsFromObject 10-287

GetVertexPosition 10-288

GetVisibleLevels 28-32

GridSetup 26-72, 28-33

GroupPlotCurvesByGroupingStrategy 12-  
71

**H**

HasSameData 24-34

HealObject 10-199

hierarchy of variables in HFSS 1-73

**I**

If...Then... Else statement 1-8

Import (3D Modeler command) 10-204

ImportANF 3-35

ImportAutoCAD 3-37

ImportDataset 8-7

ImportDXF 10-208

ImportGDSII 3-38, 10-212

ImportIntoReport 12-71

ImportODB 3-39

ImportSetup, Optimetrics module com-  
mand 16-14

include files

scripts 1-10

indentation, IronPython 1-19

Information Method List 26-112

InitialMeshSettings 14-35

InputBox function 1-11

Insert3DComponent 10-100

InsertDesign 5-15

InsertDesignWithWorkflow 5-16

InsertNativeComponent 10-101

InsertSetup	AddMaterial 27-2, 27-12
Analysis module command 15-19	CloneMaterial 27-9
optimization command 16-31	EditMaterial 27-13
parametric command 16-21	ExportMaterial 27-27
sensitivity command 16-49	RemoveMaterial 27-28
statistical command 16-57	Materials Scripting Support 6-23
Intersect 10-216	Maxwell boundary condition commands
IronPython 1-12, 1-13	AssignCurrent 13-50
indentation in 1-19	AssignFluxTangential_1 13-23
Is2D 9-40	AssignResistiveSheet 13-27
Is3D 9-40	AssignThinLayer 13-29
IsDataComplex 12-72	AssignTouching 13-31
IsPerQuantityPrimarySweep 12-73	AssignVectorPotential 13-32
IsUsed 27-105, 27-113, 27-141, 27-174, 27-219, 27-245	AssignVoltageAPhi_ 13-55
<b>L</b>	EditCurrent 13-59
LaunchJobMonitor 3-40	EditFluxTangential 13-35
LoadNamedExpressions 19-27	EditResistiveSheet 13-37
LoadSolution 24-36	EditThinLayer 13-40
logical operators 1-7	EditTouching 13-41
looping through code	EditVectorPotential 13-42
Do ... Loop 1-9	EditVoltageAPhi 13-64
For ... Next 1-9	Maxwell definition manager commands
<b>M</b>	ComputeCoreLossCoefficients 27-10
material commands	Maxwell Draw commands
AddDefinitionFromBlock 10-244	Create3DComponent 10-6
AddDefinitionFromLibFile 10-249	Maxwell parameter setup commands
	AddReduceOp 21-2

EditReduceOp 21-15  
Mesh Operations module commands  
AssignCylindricalGapOp 14-5  
AssignLengthOp 14-12  
AssignModelResolutionOp 14-15  
AssignSkinDepthLayerSetting 14-17  
AssignSkinDepthOp 14-18  
AssignTrueSurfOp 14-19  
DeleteOp 14-2  
EditCylindricalGapOp 14-22  
EditLengthOp 14-27  
EditModelResolutionOp 14-30  
EditSkinOp 14-31  
EditTrueSurfOp 14-33  
GetOperationNames 14-3  
InitialMeshSettings 14-35  
RenameOp 14-3  
Mesh Operations Module Script Commands 14-1  
mesh operations script commands 14-2  
method format, create schematic objects 26-2  
Microsoft  
VBScript user's guide 1-12  
Mirror 10-131  
Model Manager Script Commands 27-114

Modeler Menu Commands 10-140  
ModifyFieldPlot 18-43  
ModifyInceptionParameters 18-46  
ModifyLibraries 27-252  
module script commands  
boundary/excitation 13-2  
field overlay 18-1  
general method 26-45  
mesh operations 14-2  
Module Setup commands  
GetSymmetryMultiplier H  
SetSymmetryMultiplier H  
modules in HFSS scripting 1-73  
Motion Setup commands  
AssignBand C  
DeleteMotionSetup A  
EditMotionSetup E  
GetMotionSetupNames G  
ReassignMoving B  
Move 10-133, 28-34  
MoveCSToEnd 10-217  
MoveEntityToGroup 10-218  
MoveFaces 10-219  
MovePlotCurvestoGroup 12-74  
MovePlotCurvestoNewGroup 12-75  
MsgBox function 1-11

**N**

Named Arguments 1-83

NameNets 26-79  
NdExplorer  
    Script Commands 24-1  
Network Data Explorer Manager  
    Commands 27-144  
NewProject 3-40

**O**

oAnsoftApp object 1-73  
Object Oriented Property  
    Scripting 6-1  
oDesign object 1-73  
oDesktop object 1-73  
oEditor object 1-73  
OffsetFaces 10-134  
oModule object 1-73  
Open 24-37  
OpenAndConvertProject 3-41  
OpenExternalEditor 10-288  
OpenMultipleProjects 3-42  
OpenProject 3-43  
operators  
    arithmetic 1-6  
    categories in VBScript 1-5  
    comparison 1-7  
    logical 1-7  
    precedence of 1-5  
oProject object 1-73  
Optimetrics module commands  
    DeleteSetups 16-8

DistributeAnalyzeSetup 16-8  
ExportDXConfigFile 16-9  
ExportOptimetricsProfile 16-10  
ExportOptimetricsResults 16-11  
ExportParametricResults 16-12  
GetSetupNames 16-13  
GetSetupNamesByType 16-13  
ImportSetup 16-14  
RenameSetup 16-16  
SolveSetup 15-2, 16-7, 16-17, 16-  
    18  
Optimetrics Module Script  
    Commands 16-1  
optimization commands  
    EditSetup 16-26  
    InsertSetup 16-31  
Optimization Script Commands 16-26  
Other oEditor Commands 10-241  
output variable commands  
    CreateOutputVariable 11-1  
    DeleteOutputVariable 11-3  
    DoesOutputVariableExist 11-3  
    EditOutputVariable 11-4  
    GetOutputVariableValue 11-6  
Output Variable Script  
    Commands 11-1

**P**

PageBorders 26-81  
PageSetup 10-289

Pan 26-81, 28-35	PasteSetup
Parameter setup commands	Analysis module command 15-32
DeleteReduceMatrix 21-11	Optimetrics module command 16-15
DeleteReduceOp 21-11	PasteSweep 15-33
RenameReduceMatrix 21-17	PasteTraces 12-77
RenameReduceOp 21-18	PauseRecording 3-45
Parameter Setup commands	PauseScript 3-46
AssignMatrix 21-7	Print 3-47
AssignTorque 21-9	Project object commands
DeleteAllParameters 21-10	AddDataset 8-1
EditForce 21-12	AddMaterial 27-2, 27-12
EditMatrix 21-15	ChangeProperty 28-6
EditTorque 21-16	CloneMaterial 27-9
ReassignParameter 21-16	Close 5-4
RenameParameter 21-17	CopyDesign 5-5
parametric commands	CutDesign 5-6
EditSetup 16-19	DeleteDataset 8-4
GenerateVariationData Para- metric 16-20	DeleteDesign 5-7
InsertSetup 16-21	EditDataset 8-5, 8-7
Parametric Script Commands 16- 18	EditMaterial 27-13
Paste 3-44, 3-45, 5-17, 26-83, 28- 35	ExportMaterial 27-27
Paste (Model Editor) 10-136	GetActiveDesign 5-8
Paste (Project Object) 3-45, 5-17	GetArrayVariables 7-26
PasteData 26-84	GetDependentFiles 5-10
PasteDesign (Schematic Editor) 26-84	GetDesign 5-11
PasteReports 12-76	GetName 5-13
	GetPath 5-13
	GetProjectMaterialNames 27-27

GetProperties 7-26, 28-31  
GetPropertyValue 7-28, 12-61,  
28-32  
GetTopDesignList 5-14  
GetTopEntryValue 19-26  
GetVariables 7-30  
GetVariableValue 7-30  
InsertDesign 5-15  
LoadSolution 24-36  
Paste 3-44, 3-45, 3-45, 5-17, 5-  
17  
Redo 5-18  
RemoveMaterial 27-28  
RemoveUnusedDefinitions 27-  
29  
Rename 5-19  
RestoreProjectArchive 3-50, 5-  
20  
Save 5-21  
SaveAs 5-21  
SaveAsStandAloneProject 5-24  
SaveProjectArchive 5-24  
SetActiveDesign 5-26  
SetPropertyValue 7-31  
SetVariableValue 7-33  
SimulateAll 5-3, 5-27  
Undo 5-28  
UpdateDefinitions 5-28  
Project Object Script Commands 5-  
1  
ProjectSheet 10-221

property commands  
ChangeProperty 28-6  
GetArrayVariables 7-26  
GetProjectMaterialNames 27-27  
GetProperties 7-26, 28-31  
GetPropertyValue 7-28, 12-61, 28-  
32  
GetTopEntryValue 19-26  
GetVariables 7-30  
GetVariableValue 7-30  
SetPropertyValue 7-31  
SetVariableValue 7-33  
Property Method List 26-105  
Property Script Commands 7-1  
Conventions uSed in the  
Chapter 7-13  
Object Script Property Function  
Summary 7-3  
PurgeHistory 10-222  
PushExcitations 26-87  
Layout Editor 26-87

**Q**

QuitApplication 3-47

**R**

ReassignBoundaries 13-3, 13-14  
ReassignBoundary 13-14  
ReassignMoving B  
ReassignParameter 21-16

Redo 28-36  
design-level command 9-40  
project-level command 5-18  
references, for VBScript 1-12  
RefreshJobMonitor 3-48  
ReleaseData 12-77  
Remove 27-106, 27-141, 27-174,  
27-220, 27-246  
RemoveAssignmentFromBoundary  
13-15  
RemoveLevels 28-37  
RemoveMaterial 27-28  
RemovePort 28-37  
RemoveScript 27-251  
RemoveSolverOnDemandModel  
27-107  
RemoveUnused 27-108, 27-113,  
27-143, 27-176, 27-221, 27-  
247  
RemoveUnusedDefinitions 27-29  
Rename 5-19, 24-38  
RenameBoundary 13-16  
RenameDesignInstance 9-41  
RenameFieldPlot 18-47  
RenameOp 14-3  
RenameParameter 21-17  
RenamePart 10-291  
RenamePlotFolder 18-48  
RenameReduceMatrix 21-17  
RenameReduceOp 21-18  
RenameReport 12-78  
RenameSetup 15-34  
Analysis module command 15-33  
Optimetrics module command 16-16  
RenameTraces 12-79  
Renormalize 24-39  
Reorder 24-41, 24-42  
Repeating a Statement Until a Condition  
Becomes True 1-9  
Repeating Statements While a Condition is  
True 1-9  
Reporter editor commands  
AddCartesianLimitLine 12-4  
AddCartesianXMarker 12-6, 12-7  
AddDeltaMarker 12-8  
AddMarker 12-9  
AddMarkerToPlot 18-2  
AddNote 12-10  
AddTraceCharacteristics 12-12  
AddTraces 12-13  
ChangeProperty 12-16  
ClearAllMarkers 12-20  
ClearAllTraceCharacteristics 12-20  
CopyReportDefinition 12-23  
CopyReportsData 12-22  
CopyTraceDefinitions 12-23  
CopyTracesData 12-21  
CreateReport 12-25  
CreateReportFromFile 12-24  
CreateReportFromTemplate 12-35

CreateReportOfAllQuantities	12-36	SavePlotSettingsAsDefault	12-80
DeleteAllReports	12-38	UnGroupPlotCurvesInGroup	12-87
DeleteReport	12-39	UpdateAllReports	12-81
DeleteTraces	12-39	UpdateReports	12-82
ExportImageToFile	12-40	UpdateTraces	12-82
ExportPlotImageToFile	12-41, 18-40	UpdateTracesContextAndSweeps	12-85
ExportReport	12-43	Reporter Editor Script Commands	12-1
ExportToFile	12-46, 12-71	ReprioritizeBoundaries	13-17
FFTOnReport	12-47	Reset	24-43
GetAllCategories	12-49	ResetAllToTimeZero	15-34
GetAllQuantities	12-50	ResetLogging	3-49
GetAllReportNames	12-48	ResetPlotSettings	12-80
GetAvailableDisplayTypes	12-51	ResetSetupToTimeZero	15-35
GetAvailableReportTypes	12-52	RestoreWindow	3-51
GetAvailableSolutionss	12-52	RestToTimeZero	9-42
GetDisplayStyle	12-59	ResumeRecording	3-51
GetReportTraceNames	12-64	RevertAllToInitial	15-35
GetSolutionContexts	12-64, 12-65	RevertSetupToInitial	15-36
GroupPlotCurvesByGroupingStrategy	12-71	RevertSetupToInitialTemperature	15-37
MovePlotCurvestoGroup	12-74	Rotate	10-137, 26-90, 28-38
MovePlotCurvestoNewGroup	12-75	Running Instance Manager Script Commands	4-1
PasteReports	12-76	RunProgram	3-53
PasteTraces	12-77	RunScript	3-54
RenameReport	12-78	RunScriptWithArguments	3-55
RenameTraces	12-79	RunToolkit	9-62
ResetPlotSettings	12-80		

**S**

sample scripts

variable helix 30-13

Save 5-21, 28-39

SaveAs 5-21

SaveAsStandAloneProject 5-24

SaveNamedExpressions 19-28

SavePlotSettingsAsDefault 12-80

Scale 10-138

Schematic Scripting 26-1

Create Method List 26-6

Scope and Lifetime of Variables 1-4

Script and Library Scripts 27-248

script commands

boundary/excitation 13-2

field overlay 18-1

general method script commands 26-45

mesh operations 14-2

Script Commands for Creating and Modifying Excitations in Maxwell 13-44

Script Commands for Creating and Modifying Boundaries in Maxwell 13-18

Script Commands for Creating and Modifying Mesh Operations 14-4

scripts

CPython 1-70

IronPython 1-12, 1-13

overview 1-1

pausing 1-78

recording 1-79

running 1-77

stopping 1-78

VBScript 1-2

Section 10-223

Select Case statement 1-8

SelectAll 26-92, 28-39

SelectPage 26-93

SelectScheduler 3-56

SendToBack 26-97, 28-40

sensitivity commands

EditSetup 16-42

InsertSetup 16-50

Sensitivity Script Commands 16-42

SeparateBody 10-225

SetActiveDefinitionEditor 5-25

SetActiveDesign 5-26

SetActiveEditor 9-42

SetActiveLevel 28-40

SetActiveProject 3-58

SetActiveProjectByPath 3-59

SetAllPortImpedance 24-44

SetConductivityThreshold 9-43

SetCoreLoss 13-66

SetDesignSettings 9-44

SetEddyEffect 13-66

SetFieldPlotSettings	18-48	Solutions module commands
SetInitialLevels	28-41	ConstructVariationString 9-9
SetLibraryDirectory	3-62	DeleteFieldVariation 9-12
SetMagnetizationCompute	13-67	DeleteFullVariation 9-8, 9-13
SetMinimumTimeStep	13-69	DeleteLinkedDataVariation 9-13
SetModelUnits	10-226	DeleteSolutionVariation 17-1
SetPageData	26-94	GetValidISolutionList 17-2
SetPlotFolderSettings	18-51	Solutions Module Script Commands 17-1
SetPlotsViewSolutionContext	18-57	Solve 9-60
SetPortDeembedDistance	24-45	SolveAllSetup, Optimetrics module command 16-18
SetPortImpedance	24-46	SolveSetup, Optimetrics module command 16-17
SetPostProcSettings	24-47	SortComponents 26-98
SetProjectDirectory	3-62	Split 10-229
SetPropertyValue	7-31, 28-42	statistical commands
SetSolutionType	9-60	EditSetup 16-54
SetSymmetryMultiplier	H	InsertSetup 16-57
SetTempDirectory	3-64	Statistical Script Commands 16-54
Setting Numerical Values	1-85	StopSimLink 9-61
SetupYConnection	13-69	Stretch 24-49
SetVariableValue	7-33	Sub Procedures 1-10
SetVisibleLevels	28-42	SubmitJob 3-66
SetWCS	10-227	Subtract 10-232
SGetAppDesktop	2-2	SweepAlongPath 10-107
ShowWindow	10-228	SweepAlongVector 10-109
ShowVariableBlock	26-98	SweepAroundAxis 10-112
simple and composite names	1-3	SweepFacesAlongNormal 10-114, 10-114, 10-233, 10-233
SimulateAll	5-3, 5-27	
Sleep	3-65	
Smooth	24-48	

Sweep-	commands
FacesAlongNormalWithAttributes 10-116	GetDocumentNames 22-8
Symbol Editor Scripts 28-1	User defined solution module commands
Symbol Manager Script Commands 27-148	CreateUserDefinedSolution 23-1
<b>T</b>	DeleteUserDefinedSolutions 23-3
Terminate 24-51	EditUserDefinedSolution 23-4
ThickenSheet 10-235	User Defined Solutions Commands 23-1
ToggleLevel 28-43	Using a Do Loop 1-9
<b>U</b>	<b>V</b>
UncoverFaces 10-237	ValidateDesign 9-64
Undo 28-44	Variable Naming Conventions 1-4
project-level command 5-28	variables
Ungroup 10-239	array 1-4
UnGroupPlotCurvesInGroup 12-87	assigning information 1-3
Unite 10-238	declaring 1-3
UpdateAllFieldsPlots 18-58	hierarchy in HFSS 1-73
UpdateAllReports 12-81	used in HFSS scripts 1-73
UpdateComponentDefinition 10-119	VBScript 1-2
UpdateDefinitions, project-level command 5-28	Microsoft user's guide 1-12
UpdateDynamicLink 27-109	operators 1-5
UpdateQuantityFieldsPlots 18-58	references 1-12
UpdateReports 12-82	VBScript Procedures 1-10
UpdateTraces 12-82	<b>W</b>
UpdateTracesContextAndSweeps 12-85	Wire (Schematic Editor) 26-99
User defined documents module	WrapSheet 10-240
	<b>Z</b>
	ZoomArea 26-100, 28-45

ZoomIn 26-101, 28-46  
ZoomOut 26-102, 28-46  
ZoomPrevious 26-103, 28-47  
ZoomToFit 26-104, 28-47