



POWERING INNOVATION THAT DRIVES HUMAN ADVANCEMENT

© 2025 ANSYS, Inc. or its affiliated companies
Unauthorized use, distribution, or duplication is prohibited.

HFSS™ 3D Layout Scripting Guide



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

Release 2025 R1
January 2025

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

Copyright and Trademark Information

© 1986-2025 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. FLEXlm 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 exporting 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-Party Software

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

Table of Contents	Contents-1
1 - Introduction to Scripting	1-1
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	1-5
Arithmetic Operators	1-6
String Concatenation Operator	1-7
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	1-8
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	1-9
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	1-11
Sample HFSS Script	1-12
Sample Q3D Extractor Script	1-14
Introduction to IronPython	1-15
Scope	1-16
Python compatibility	1-16
Advantages of IronPython	1-16
IronPython Mini Cookbook	1-17
Comments	1-17
Assigning/Creating Variables	1-18
Create Lists/Arrays	1-18
Create Dictionaries/Maps	1-19
Boolean Values	1-19
Converting Numbers to Strings and Vice Versa	1-19
String Formatting/Concatenation	1-20
Looping over Lists	1-20
Looping over a Range	1-20
Indentation in IronPython	1-21
Indenting Functions	1-21
Indenting If Conditions	1-21
Methods in IronPython	1-22
Finding Methods	1-22
Help	1-22
Translating Script Commands from VBScript to IronPython	1-22
Script Method Argument	1-22

Primitive Types	1-23
Named Arrays	1-23
Named Functions	1-23
VBScript Method Call Types	1-23
Converting VBScript Function calls to IronPython Syntax	1-24
Return Values	1-24
Primitive Method Arguments	1-24
Named Array Arguments	1-25
Named Array Values with All Key Value Pairs	1-25
Named Arrays with Nested Named Arrays	1-25
Function Blocks	1-26
Scripting Using Iron Python	1-27
Translating a script in VBScript to IronPython	1-27
Writing an IronPython script from scratch	1-27
IronPython Script Execution Environment	1-28
Script Argument for IronPython	1-28
Scripting using Embedded VBScript or JavaScript	1-29
Scripting with IronPython	1-32
Standalone IronPython	1-33
Running Standalone IronPython	1-33
Using a Recorded Script	1-33
Creating an External Script	1-33
Example Script	1-34
IronPython Samples	1-35
Change property	1-36
Create a Cone using IronPython	1-37
Creating User Defined Primitives and User Defined Models in Python Scripts	1-41
Advantages Compared to C++	1-41
Changes compared to C	1-41
Structures	1-41

Return Values for UDM and UDP Functions	1-41
Constants	1-41
Methods	1-42
Output Parameters	1-42
Comparison with C function:	1-43
'List Size' Parameters	1-43
Comparison with C function:	1-44
Added Parameters	1-44
Developing a UDM/UDP	1-45
Creation	1-45
Location	1-45
Organize	1-46
Edit/Reload	1-46
UDPExtension	1-46
Import	1-46
Main class: UDPExtension	1-46
IUDPExtension methods	1-47
Mandatory methods.	1-47
GetLengthParameterUnits()	1-47
GetPrimitiveTypeInfo()	1-47
GetPrimitiveParametersDefinition2()	1-47
AreParameterValuesValid2(errorMsg, udpParams)	1-47
CreatePrimitive2(funcLib, udpParams)	1-47
Optional methods	1-47
GetPrimitiveParameters()	1-47
GetRegisteredFaceNames()	1-48
GetRegisteredEdgeNames()	1-48
GetRegisteredVertexNames()	1-48
MapParametersDefinitionVersions2(oldVersion, oldUDPParams)	1-48
GetOldPrimitiveParametersDefinition2(version)	1-48

Example UDP	1-48
UDMExtension	1-49
Import	1-49
Main class: UDMExtension	1-49
IUDMExtension methods	1-49
Mandatory methods.	1-49
GetInfo()	1-49
IsAttachedToExternalEditor()	1-49
CreateInstance(funcLib)	1-50
GetUnits(instanceld)	1-50
ReleaseInstance(instanceld)	1-50
GetAttribNameForEntityId()	1-50
GetAttribNameForPartId()	1-51
Optional methods	1-51
GetInstanceSourceInfo(instanceld)	1-51
ShouldAttachDefinitionFilesToProject()	1-52
Example UDM	1-52
UDMFunctionLibrary	1-52
Functions list:	1-53
UDM/UDP Functions	1-54
Return Values for Each UDM and UDP Function	1-54
UDP/UDM Structures and Constants	1-56
UDP/UDM Structures	1-57
List of structures	1-58
UDP/UDM Constants	1-64
Enum constants:	1-64
UDP Python Example	1-66
Introduction to CPython	1-71
Creating an External Script	1-71
Start ansyedt as GRPC server	1-72

Connect Functions:	1-72
Example:	1-73
Launching Electronics Desktop	1-73
Connecting with a Running Instance of Electronics Desktop	1-73
Closing Electronics Desktop/Ending the Script	1-74
-grpcsrv Flag	1-74
Ansys Electronics Desktop Scripting	1-76
Overview of Electronics Desktop Scripting Objects	1-76
oAnsoftApp	1-77
oDesktop	1-77
oProject	1-77
oDesign	1-77
oEditor	1-78
oModule	1-78
Example Script Opening	1-79
GetActiveProject and GetActiveDesign for Wider Use	1-79
Running a Script	1-80
Within Electronics Desktop	1-80
From the Command Line	1-81
Direct Launch	1-81
Recording a Script	1-82
Recording a Script to File	1-82
Recording a Script to a Project	1-83
Working with Project Scripts	1-83
Executing a Script from Within a Script	1-84
Electronics Desktop Scripting Conventions	1-85
Named Arguments	1-85
VBscript Example	1-86
IronPython Example	1-86
Setting Numerical Values	1-87

Layout Scripts and the Active Layer	1-88
Scripts and Locked Layers	1-88
Event Callback Scripting	1-89
PyAEDT (Beta)	1-90
2 - Object-Oriented Property Scripting	2-1
Object-Oriented Scripting	2-1
Material Properties and Examples	2-9
Body Properties and Modification	2-11
Retrieving Variables	2-11
Retrieve Datasets and Values	2-12
GetSolutionData API	2-14
Summary	2-14
Additional Details Specific to AEDT Solvers	2-18
Additional details on Boundaries/Excitations	2-19
3D component encapsulation	2-21
Additional details on Solve setup	2-21
Materials Scripting Support	2-22
Object Oriented Scripting for Materials	2-26
Examples showing change to material property type:	2-27
Examples showing change to a vector component value	2-27
Change choice property value	2-28
Change choice property value	2-29
Property Script Commands	2-30
Object Script Property Function Summary	2-32
Object Path	2-32
Property Object	2-33
Project Object	2-34
Design Object	2-35
3D Modeler Object	2-36
Variable Object	2-36

Optimetrics Module Object:	2-37
Optimetrics Setup Object	2-38
ReportSetup(Results) Module Object:	2-39
ReportSetup(Results) Module Child Objects:	2-39
Radiation Module Object:	2-40
Radiation Module Child Objects:	2-41
Conventions Used in this Chapter	2-41
Callback Scripting Using PropHost Object	2-44
ChangeProperty (Schematic Editor and Layout Editor)	2-47
PropHost Functions	2-56
AddMenuProp	2-56
AddMenuProp2	2-58
AddProp	2-59
AddProp2	2-63
ExecuteScript	2-65
GetApplication	2-66
GetCallback	2-67
GetChangedProperty	2-67
GetDescription	2-68
GetDesign	2-69
GetEditor	2-69
GetEvaluatedText	2-70
GetFileName	2-71
GetHidden	2-71
GetProgress	2-72
GetPropHost	2-72
GetPropServers	2-73
GetPropTabType	2-74
GetReadOnly	2-75
GetRunStatus	2-75

GetTabTypeName	2-76
GetText	2-76
GetValue	2-77
IsValueConstant	2-78
PropertyExists	2-79
RemoveProp	2-79
SetCallback	2-80
SetDescription	2-81
SetHidden	2-81
SetReadOnly	2-82
SetText	2-83
SetValue	2-84
GetArrayVariables	2-94
GetProperties	2-95
GetPropertyValue	2-96
GetVariables	2-98
GetVariableValue	2-98
SetPropertyValue	2-99
SetVariableValue	2-101
Example Use of Record Script and Edit Properties	2-101
Additional Property Scripting Examples	2-102
3 - Application Object Script Commands	3-1
GetAppDesktop	3-2
4 - Desktop Object Script Commands	4-1
AddMessage	4-6
AreThereSimulationsRunning	4-7
ClearMessages	4-7
CloseAllWindows	4-11
CloseProject	4-12
CloseProjectNoForce	4-13

Count	4-14
DeleteProject	4-15
DownloadJobResults	4-16
DeleteRegistryEntry	4-17
DoesRegistryValueExist	4-18
EnableAutoSave	4-19
ExportOptionsFiles	4-19
GetActiveProject	4-20
GetActiveScheduler	4-21
GetActiveSchedulerInfo	4-22
GetAutoSaveEnabled	4-23
GetBuildDateTimeString	4-23
GetCustomMenuSet	4-24
GetDefaultUnit	4-25
GetDesktopConfiguration	4-27
GetDistributedAnalysisMachines	4-28
GetDistributedAnalysisMachinesForDesignType	4-28
GetExeDir	4-29
GetGDIObjectCount	4-30
GetLibraryDirectory	4-31
GetLocalizationHelper	4-32
GetMessages	4-33
GetMonitorData	4-34
GetPersonalLibDirectory	4-35
GetProcessID	4-36
GetProjectDirectory	4-36
GetProjectList	4-37
GetProjects	4-38
GetRegistryInt	4-38
GetRegistryString	4-39

GetRunningInstancesMgr	4-40
GetSchematicEnvironment	4-41
GetScriptingToolsHelper	4-42
GetSysLibDirectory	4-42
GetTempDirectory	4-43
GetUserLibDirectory	4-44
GetVersion	4-44
IsFeatureEnabled	4-45
KeepDesktopResponsive	4-47
LaunchJobMonitor	4-47
NewProject	4-48
OpenAndConvertProject	4-49
OpenMultipleProjects	4-50
OpenProject	4-50
OpenProjectWithConversion	4-51
PageSetup (Layout Editor)	4-52
PauseRecording	4-52
PauseScript	4-53
Print	4-54
QuitApplication	4-55
RefreshJobMonitor	4-55
ResetLogging	4-56
RestoreProjectArchive	4-57
RestoreWindow	4-58
ResumeRecording	4-59
RunACTWizardScript	4-59
RunProgram	4-60
RunScript	4-61
RunScriptWithArguments	4-62
SelectScheduler	4-64

SetActiveProject	4-65
SetActiveProjectByPath	4-66
SetCustomMenuSet	4-67
SetDesktopConfiguration	4-68
SetLibraryDirectory	4-69
SetProjectDirectory	4-69
SetRegistryFromFile	4-70
SetRegistryInt	4-71
SetRegistryString	4-72
SetSchematicEnvironment	4-73
SetTempDirectory	4-73
ShowDockingWindow	4-74
Sleep	4-75
StopSimulations	4-76
SubmitJob	4-76
TileWindows	4-78
Desktop Commands For Registry Values	4-78
DeleteRegistryEntry	4-79
DoesRegistryValueExist	4-80
GetRegistryInt	4-81
GetRegistryString	4-82
SetRegistryFromFile	4-83
SetRegistryInt	4-83
SetRegistryString	4-84
ImportExport Tool Commands	4-85
ImportANF	4-86
ImportANFv2	4-87
ImportAutoCAD	4-89
ImportAWRMicrowaveOffice	4-90
ImportEDB	4-92

ImportExtracta	4-93
ImportGDSII	4-94
ImportGerber	4-95
ImportIDF	4-96
ImportIDFandMerge	4-99
ImportIDX	4-100
ImportIPC	4-103
ImportODB	4-104
ImportXFL	4-105
5 - Running Instances Manager Script Commands	5-1
GetAllRunningInstances	5-1
GetRunningInstanceByProcessID	5-2
GetRunningInstancesMgr	5-2
6 - Project Object Script Commands	6-1
AddDataset	6-4
AddMaterial	6-7
AnalyzeAll [project]	6-11
ChangeProperty	6-12
ClearMessages	6-16
CloneMaterial	6-17
Close	6-18
CopyDesign	6-19
CutDesign	6-20
DeleteDataset	6-21
DeleteDesign	6-21
DeleteToolObject	6-22
EditDataset	6-23
EditMaterial	6-25
ExportDataset	6-34
ExportMaterial	6-35

GetActiveDesign	6-36
GetArrayVariables	6-37
GetChildNames [Project]	6-38
GetChildObject [Project]	6-38
GetChildTypes [Project]	6-40
GetConfigurableData (Project)	6-40
GetDefinitionManager	6-41
GetDependentFiles	6-41
GetDesign	6-42
GetDesigns	6-43
GetEDBHandle	6-44
GetLegacyName	6-44
GetName [Project]	6-45
GetObjPath [Project]	6-46
GetPath	6-47
GetPropEvaluatedValue	6-47
GetPropNames [Project]	6-48
GetPropSIValue	6-49
GetPropValue [Project]	6-50
GetProperties	6-51
GetPropertyValue	6-52
GetTopDesignList	6-54
GetVariableValue	6-54
GetVariables	6-55
ImportDataset	6-56
Note About File Types:	6-57
InsertDesign	6-58
InsertDesign (Layout Editor)	6-59
InsertDesignWithWorkflow	6-60
InsertToolObject	6-61

Paste (Project Object)	6-62
Redo [Project Level]	6-62
RemoveAllUnusedDefinitions	6-63
RemoveMaterial	6-64
RemoveUnusedDefinitions	6-65
Rename	6-66
RunToolkit	6-67
Save	6-68
SaveAs	6-68
SaveAsStandAloneProject	6-71
SaveProjectArchive	6-72
SetActiveDefinitionEditor	6-73
SetActiveDesign	6-74
SetPropValue [Project]	6-74
SetPropertyValue	6-75
SetVariableValue	6-77
SimulateAll	6-78
Undo [Project]	6-78
UpdateDefinitions	6-79
7 - Design Object Script Commands	7-1
AddDataset	7-7
AddDesignVariablesForDynamicLink	7-10
AddDynamicLink	7-11
AddModelingProperties	7-13
Analyze	7-13
AnalyzeAll [design]	7-14
AnalyzeAllNominal	7-15
AnalyzeDistributed	7-16
ApplyMeshOps	7-17
ChangeProperty[ReportSetup]	7-18

CopyItemCommand	7-27
CreateReport	7-27
DeleteDataset	7-37
DeleteDesignInstance	7-38
DeleteFieldVariation	7-38
DeleteFullVariation	7-39
DeleteOutputVariable	7-40
DeleteSolutionVariation	7-41
DistributeAnalysis	7-42
EditCoSimulationOptions	7-43
EditDataset	7-44
EditImportData	7-46
EditInfiniteArray	7-47
Edit (Layout Editor)	7-47
EditLayoutForLayoutComponent	7-49
EditNotes	7-50
EditNotes (Layout Editor)	7-50
EditOptions (Layout Editor)	7-51
EditOutputVariable	7-51
EMDesignOptions	7-53
ExportCircuit (Layout Editor)	7-55
ExportConvergence	7-60
ExportDataset	7-61
ExportForHSpice	7-61
ExportForHSpice (Layout Editor)	7-63
ExportForSpice (Layout Editor)	7-65
ExportMeshStats	7-66
ExportNetworkData	7-67
ExportNetworkData	7-70
ExportNMFData	7-72

ExportProfile	7-74
ExportReport	7-75
ExportRLGCData	7-77
ExportRLGCGen	7-79
Generate Mesh	7-81
GetActiveEditor (Layout Editor)	7-82
GetChildNames [Design]	7-82
GetChildObject [Design]	7-83
GetChildTypes [Design]	7-84
GetConfigurableData	7-85
GetData	7-86
GetDesignID	7-86
GetDesignName	7-87
GetDesignType	7-87
GetDesignValidationInfo	7-88
GetEditor	7-89
GetEditor (Layout Editor)	7-89
GetGeometryIdsForNetLayerCombinations	7-90
GetGeometryIdsForAllNetLayerCombinations	7-91
GetManagedFilePath	7-93
GetModule	7-93
GetModule (Layout Editor)	7-94
GetModule (RadiationSetupMgr)	7-96
GetName	7-97
GetName (Layout Editor)	7-98
GetNoteText	7-98
GetObjPath [Design]	7-99
GetOutputVariableValue	7-99
GetOutputVariables	7-101
GetProject	7-102

GetPropHost	7-102
GetProperties	7-103
GetPropertyValue	7-104
GetPropNames [Design]	7-106
GetPropValue [Design]	7-106
GetVariables	7-107
GetVariableValue	7-108
ImportDataset	7-109
Note About File Types:	7-110
InsertDesign	7-111
OverlayCurrents (Layout Editor)	7-112
OverlayFarField (Layout Editor)	7-113
OverlayMesh (Layout Editor)	7-113
OverlayNearField (Layout Editor)	7-114
PasteDesign (Design Object)	7-114
PasteItemCommand (Layout Editor)	7-115
PushExcitations	7-116
Redo [Design]	7-119
Redo (Layout Editor)	7-120
RemoveImportData (Layout Editor)	7-120
RemoveModelingProperties (Layout Editor)	7-121
RenameDesignInstance	7-121
RenameDesignInstance (Layout Editor)	7-122
RenameImportData (Layout Editor)	7-122
RenameSource [Interface Source]	7-123
ReportTemplates (Layout Editor)	7-123
RunToolkit	7-123
SetActiveEditor	7-124
SetActiveEditor (Layout Editor)	7-125
SetBendGraphicsMode	7-126

SetDesignMode	7-126
SetDoMeshAssembly	7-127
SetFastTransformationForLayoutComponent	7-132
SetObjectAttributesForLayoutComponent	7-132
SetPropValue [Design]	7-135
SetPropertyValue	7-136
SetShowLayoutForLayoutComponent	7-137
SetVariableValue	7-138
Solve	7-139
StartAnalysis (Layout Editor)	7-139
StopSimLink	7-140
Undo [Design]	7-141
Undo (Layout Editor)	7-141
ValidateCircuit (Layout Editor)	7-142
ValidateLink	7-142
8 - 3D Modeler Editor Script Commands	8-1
Conventions Used in this Chapter:	8-1
<AttributesArray>	8-1
<SelectionsArray>	8-2
Draw Menu Commands	8-4
Create3DComponent	8-6
CreateBondwire	8-11
CreateBox	8-15
CreateCircle	8-17
CreateCone	8-20
CreateCutplane	8-23
CreateCylinder	8-25
CreateEllipse	8-27
CreateEquationCurve	8-30
CreateEquationSurface	8-34

CreateHelix	8-37
CreatePoint	8-39
CreatePolyline	8-41
CreateRectangle	8-46
CreateRegion	8-49
CreateRegularPolygon	8-52
CreateRegularPolyhedron	8-55
CreateSphere	8-58
CreateSpiral	8-61
CreateTorus	8-63
CreateUserDefinedPart	8-65
Edit3DComponent	8-73
Edit3DComponentDefinition	8-76
EditPolyline	8-78
Get3DComponentDefinitionNames	8-82
Get3DComponentInstanceNames	8-83
Get3DComponentMaterialNames	8-84
Get3DComponentMaterialProperties	8-84
Get3DComponentParameters	8-85
Get3DComponentPartNames	8-86
Insert3DComponent	8-86
InsertComponent	8-87
InsertPolylineSegment	8-89
SweepAlongPath	8-91
SweepAlongVector	8-93
SweepAroundAxis	8-95
SweepFacesAlongNormal	8-97
SweepFacesAlongNormalWithAttributes	8-99
UpdateComponentDefinition	8-102
Edit Menu Commands	8-103

Copy	8-104
DeletePolylinePoint	8-105
DuplicateAlongLine	8-106
DuplicateAroundAxis	8-108
DuplicateMirror	8-111
Mirror	8-114
Move	8-116
OffsetFaces	8-118
Paste (Model Editor)	8-119
Rotate	8-120
Scale	8-121
Modeler Menu Commands	8-123
AlignFaces	8-125
AssignMaterial	8-126
Chamfer	8-129
CleanUpModel	8-131
Connect	8-132
CoverLines	8-133
CoverSurfaces	8-134
CreateEntityList	8-135
CreateFaceCS	8-136
CreateGroup	8-141
CreateObjectCS	8-142
CreateObjectFromEdges	8-149
CreateObjectFromFace	8-151
CreateObjectFromFaces	8-153
CreateRelativeCS	8-154
DeleteEmptyGroups	8-156
DeleteLastOperation	8-157
DeleteOperation	8-158

DetachEdges	8-160
DetachFaces	8-161
EditFaceCS	8-163
EditObjectCS	8-169
EditRelativeCS	8-176
Export	8-178
ExportModelImageToFile	8-180
ExportModelMeshToFile	8-183
Fillet	8-184
FlattenGroup	8-186
GenerateHistory	8-187
GetActiveCoordinateSystem	8-188
GetActiveCoordinateSystemTransform	8-189
GetCoordinateSystems	8-189
HealObject	8-190
Import	8-194
ImportDXF [Modeler]	8-198
ImportFromClipboard	8-202
ImportGDSII [Modeler]	8-203
Imprint	8-206
ImprintProjection	8-208
Intersect	8-210
MoveCStoEnd	8-211
MoveEntityToGroup	8-212
MoveFaces	8-213
ProjectSheet	8-215
PurgeHistory	8-216
ReplaceWith3DComponent	8-217
Section	8-220
SeparateBody	8-222

SetModelUnits	8-223
SetWCS	8-224
ShowWindow	8-225
Simplify	8-226
Split	8-229
Stitch	8-231
Subtract	8-233
SweepFacesAlongNormal	8-234
ThickenSheet	8-236
UncoverFaces	8-238
Unite	8-239
Ungroup	8-241
WrapSheet	8-242
Other oEditor Commands	8-243
AddDefinitionFromLibFile	8-245
AssignSurfaceMaterial	8-250
BreakUDMConnection	8-252
ChangeProperty	8-253
CloseAllWindows[Editor]	8-258
Delete	8-258
FitAll	8-259
GenerateAllUserDefinedModels	8-260
GenerateUserDefinedModel	8-260
GeometryCheckAndAutofix	8-261
GetBodyNamesByPosition	8-263
GetChildNames [Modeler]	8-265
GetChildObject [Modeler]	8-268
GetChildTypes [Modeler]	8-271
GetEdgeByPosition	8-273
GetEdgeIdsFromFace	8-275

GetEdgeIDsFromObject	8-275
GetEdgeLength	8-276
GetEntityListIDByName	8-277
GetFaceArea	8-277
GetFaceCenter	8-278
GetFaceByPosition	8-279
GetFaceIDs	8-280
GetFaceIDsOfSheet	8-281
GetMatchedObjectName	8-281
GetModelBoundingBox	8-282
GetModelUnits	8-283
GetNumObjects	8-283
GetObjectIDByName	8-284
GetObjectName	8-285
GetObjectNameByEdgeID	8-285
GetObjectNameByFaceID	8-286
GetObjectNameByID	8-287
GetObjectNameByVertexID	8-287
GetObjectsByMaterial	8-288
GetObjectShapeType	8-289
GetObjectsInGroup	8-290
GetObjectVolume	8-291
GetObjPath [Editor]	8-291
GetPartsForUserDefinedModel	8-292
GetPoints [3D Modeler Editor]	8-293
GetPropEvaluatedValue	8-293
GetPropertyValue	8-294
GetPropNames [Modeler]	8-296
GetPropSIValue	8-297
GetPropValue [Modeler]	8-298

GetRelativeCoordinateSystems	8-299
GetSelections [Model Editor]	8-300
GetSubGroupsInGroup	8-300
GetUserPosition	8-301
GetVertexIDFromNameForFirstOperation	8-302
GetVertexIDsFromEdge	8-303
GetVertexIDsFromFace	8-303
GetVertexIDsFromObject	8-304
GetVertexPosition	8-305
GetWireBodyNames	8-305
PageSetup	8-306
RemoveBadEdges	8-308
RemoveBadFaces	8-308
RemoveBadVertices	8-309
RenamePart	8-310
SetPropValue [Modeler]	8-311
SetTopDownViewDirectionForActiveView	8-312
SetTopDownViewDirectionForAllViews	8-313
UpdatePriorityList	8-313
UpgradeVersion	8-314
Validate3DComponent	8-315
WriteHistoryTreeLayoutForTest	8-316
Cable Modeling Commands	8-318
AddCableToBundle	8-319
CreateCableBundle	8-320
CreateCableHarness	8-322
CreateClockSource	8-324
CreatePWLSource	8-326
CreateStraightWireCable	8-328
CreateTwistedPairCable	8-329

ExportCableLibrary	8-331
ImportCableLibrary	8-332
RemoveCable	8-332
UpdateCableHarness	8-333
9 - Output Variable Script Commands	9-1
CreateOutputVariable	9-1
DeleteOutputVariable	9-3
DoesOutputVariableExist	9-4
EditOutputVariable	9-5
ExportOutputVariables	9-6
GetOutputVariables	9-7
GetOutputVariableValue	9-8
ImportOutputVariables	9-9
SimValueContext	9-10
10 - Reporter Editor Script Commands	10-1
AddAllEyeMeasurements	10-6
AddCartesianLimitLine	10-6
AddCartesianLimitLineFromCurve	10-8
AddCartesianLimitLineFromEquation	10-10
AddCartesianXMarker	10-12
AddCartesianYMarker	10-12
AddCartesianYMarkerToStack	10-13
AddDeltaMarker	10-14
AddMarker	10-16
AddNote	10-17
AddTraceCharacteristics	10-19
AddTraces	10-20
ApplyReportTemplate	10-23
ChangeProperty[ReportSetup]	10-23
ClearAllMarkers	10-33

ClearAllTraceCharacteristics	10-33
CloneReportsFromDatasetSolution	10-34
CopyPlotSettings	10-35
CopyReportDefinitions	10-35
CopyReportsData	10-36
CopyTraceDefinitions	10-37
CopyTracesData	10-38
CreateReport	10-39
CreateReportFromTemplate	10-48
CreateReportOfAllQuantities	10-49
DeleteMarker	10-50
DeleteAllReports	10-51
DeleteReports	10-52
DeleteTraces	10-52
DoesSupportTraceCharacteristics	10-53
EditCartesianXMarker	10-54
EditCartesianYMarker	10-55
EditMarker	10-56
ExportEyeMaskViolation	10-57
ExportImageToFile [Reporter]	10-58
ExportModelImageToFile	10-62
ExportModelMeshToFile	10-65
ExportPlot3DToFile [Reporter]	10-66
ExportReport	10-67
ExportReportDataToFile	10-69
ExportTableToFile	10-70
ExportToFile [Reporter]	10-70
ExportUniformPointsToFile	10-72
GetAllCategories	10-73
GetAllQuantities	10-74

GetAllReportNames	10-75
GetAvailableDisplayTypes	10-76
GetAvailableReportTypes	10-77
GetAvailableSolutions	10-77
GetChildNames [Report Setup]	10-78
GetChildObject [Report Setup]	10-79
GetChildTypes [ReportSetup]	10-80
GetCurvePropServerName	10-80
GetDisplayType	10-81
GetDynLinkIntrinsicVariables	10-82
GetDynLinkQtyValueState	10-82
GetDynLinkTraces	10-83
GetDynLinkVariableValues	10-84
GetName	10-85
GetObjPath [Design]	10-86
GetPropertyValue	10-86
GetPropNames [Reporter]	10-88
GetPropValue [Report Setup]	10-89
GetQtyExpressionsForSourceTrace	10-89
GetReportTraceNames	10-90
GetReportSummaryForRegressionTesting	10-91
GetSolutionContexts	10-91
GetSolutionDataPerVariation	10-92
GetDataExpressions	10-95
GetDataUnits	10-96
GetDesignVariableNames	10-96
GetDesignVariableUnits	10-97
GetDesignVariableValue	10-98
GetDesignVariationKey	10-99
GetImagDataValues	10-100

GetPerQuantityPrimarySweepValues	10-101
GetRealDataValues	10-102
GetSweepNames	10-103
GetSweepUnits	10-104
GetSweepValues	10-105
IsDataComplex	10-106
IsPerQuantityPrimarySweep	10-107
Release Data	10-108
GroupPlotCurvesByGroupingStrategy	10-109
ImportIntoReport	10-110
ImportReportDataIntoReport	10-111
MovePlotCurvesToGroup	10-112
MovePlotCurvesToNewGroup	10-113
OpenWindowForAllReports	10-113
OpenWindowForReports	10-114
PastePlotSettings	10-115
PasteReports	10-116
PasteReportsWithLegacyNames	10-116
PasteTraces	10-117
PasteTracesWithLegacyNames	10-118
RenameReport	10-118
RenameTrace	10-119
ResetPlotSettings	10-120
SavePlotSettingsAsDefault	10-121
SetLinkOutputTraces	10-121
SetPropValue [Report Setup]	10-122
UnGroupPlotCurvesInGroup	10-123
UpdateAllReports	10-124
UpdateReports	10-125
UpdateTraces	10-125

UpdateTracesContextAndSweeps	10-128
11 - Optimetrics Module Script Commands	11-1
CopySetup	11-7
DeleteSetups [Optimetrics]	11-8
DistributedAnalyzeSetup	11-9
EnableSetup	11-9
ExportDXConfigFile	11-10
ExportOptimetricsProfile	11-11
ExportOptimetricsResult	11-12
ExportParametricResults	11-13
ExportParametricSetupTable	11-14
ExportRespSurfaceMinMaxTable	11-15
ExportRespSurfaceRefinePoints	11-16
ExportRespSurfaceResponsePoints	11-17
ExportRespSurfaceVerificationPoints	11-17
GenerateVariationData [Parametric]	11-18
GetChildNames [Optimetrics]	11-19
GetChildObject [Optimetrics]	11-20
GetChildTypes [Optimetrics]	11-20
GetName	11-21
GetObjPath [Design]	11-22
GetOptimetricResult	11-22
GetOptimetricsResult	11-24
GetPropNames [Optimetrics]	11-25
GetPropValue [Optimetrics]	11-25
GetSetupNames [Optimetrics]	11-26
GetSetupNamesByType [Optimetrics]	11-27
ImportSetup	11-28
PasteSetup [Optimetrics]	11-29
RenameSetup [Optimetrics]	11-30

SetPropValue [Optimetrics]	11-30
SolveAllSetup	11-31
SolveSetup [Optimetrics]	11-32
ValidateSetup [Parametric]	11-33
General Commands Recognized by the Optimetrics Module	11-34
CopySetup	11-35
DeleteSetups [Optimetrics]	11-36
DistributedAnalyzeSetup	11-37
EnableSetup	11-38
ExportDXConfigFile	11-38
ExportOptimetricsProfile	11-39
ExportOptimetricsResult	11-40
ExportParametricResults	11-41
ExportParametricSetupTable	11-42
ExportRespSurfaceMinMaxTable	11-43
ExportRespSurfaceRefinePoints	11-44
ExportRespSurfaceResponsePoints	11-45
ExportRespSurfaceVerificationPoints	11-46
ExportDOEResponseCurve	11-46
ExportDOEResponseCurveSlices	11-47
ExportDOEResponseSurface	11-48
ExportDOELocalSensitivity	11-49
ExportDOELocalSensitivityCurve	11-50
GenerateVariationData [Parametric]	11-51
GetChildNames [Optimetrics]	11-52
GetChildObject [Optimetrics]	11-52
GetChildTypes [Optimetrics]	11-53
GetName	11-53
GetObjPath [Design]	11-54
GetOptimetricResult	11-55

GetPropNames [Optimetrics]	11-56
GetPropValue [Optimetrics]	11-57
GetSetupNames [Optimetrics]	11-58
GetSetupNamesByType [Optimetrics]	11-58
ImportSetup	11-59
PasteSetup [Optimetrics]	11-61
RenameSetup [Optimetrics]	11-61
SetPropValue [Optimetrics]	11-62
SolveAllSetup	11-63
SolveSetup [Optimetrics]	11-64
ValidateSetup [Parametric]	11-64
Parametric Script Commands	11-65
EditSetup [Parametric]	11-65
ExportParametricSetupTable	11-66
GenerateVariationData [Parametric]	11-67
InsertSetup [Parametric]	11-67
ValidateSetup [Parametric]	11-74
Optimization Script Commands	11-75
EditSetup [Optimization]	11-75
InsertSetup [Optimization]	11-80
Sensitivity Script Commands	11-88
EditSetup [Sensitivity]	11-88
Statistical Script Commands	11-95
EditSetup [Statistical]	11-95
InsertSetup [Statistical]	11-98
12 - Solutions Module Script Commands	12-1
DeleteImportData	12-1
DeleteSolutionVariation	12-2
EditQ3DSources (Layout Editor)	12-3
EditSources (Layout Editor)	12-7

ExportEigenmodes	12-11
ExportForHSpice	12-12
ExportNetworkData	12-14
ExportNMFData	12-16
FFTONReport	12-18
GetAdaptiveFreq	12-19
GetAvailableVariations	12-19
GetExcitationScaling	12-20
GetSolutionVersionID	12-21
GetSolveRangeInfo	12-21
GetSourceContexts	12-22
GetSourceData (Layout Editor)	12-23
GetValidSolutionList	12-23
HasFields	12-24
HasMatrixData	12-24
HasMesh	12-25
ImportSolution	12-26
ImportTable	12-27
IsFieldAvailableAt	12-28
ListMatchingVariations	12-29
ListValuesOfVariable	12-30
SetQ3DSATNetChanges (Layout Editor)	12-31
13 - Field Overlays Module Script Commands	13-1
AddMarkerToPlot	13-4
AddNamedExpression	13-5
CalcOp	13-6
CalcStack	13-7
CalculatorRead	13-7
CalculatorWrite	13-8
ChangeProperty	13-9

ChangeGeomSettings	13-14
ClcEval	13-14
ClcMaterial	13-15
ClearAllNamedExpr	13-16
CopyNamedExprToStack	13-17
CreateFieldPlot	13-18
DeleteFieldPlot	13-29
DeleteMarker[Fields Reporter]	13-30
DeleteNamedExpr	13-31
DeleteUneditablePlot	13-32
DoesNamedExpressionExists	13-32
EditSurfaceMeshSummaryData	13-33
EnterComplex	13-37
EnterComplexVector	13-38
EnterCoord	13-39
EnterEdge	13-40
EnterLine	13-40
EnterOutputVar	13-41
EnterPoint	13-42
EnterQty	13-43
EnterScalar	13-43
EnterScalarFunc	13-44
EnterSurf	13-45
EnterVector	13-45
EnterVectorFunc	13-46
EnterVol	13-47
ExportFieldPlot	13-47
ExportMarkerTable	13-48
ExportOnGrid [Field Overlays]	13-49
ExportPlotImageToFile	13-52

ExportPlotImageWithViewToFile [Reporter]	13-53
ExportSurfaceMeshSummary	13-54
ExportToFile	13-56
GetFieldFolderNames	13-57
GetFieldPlotNames	13-58
GetFieldPlotQuantityName	13-58
GetMeshPlotNames	13-59
GetTopEntryValue	13-60
HideAntennaParametersOverlay	13-61
HidePolarPlot	13-61
HideRadiatedPlotOverlay	13-62
LoadNamedExpressions	13-63
ModifyFieldPlot	13-64
ReassignFieldPlot	13-66
RenameFieldPlot	13-69
RenamePlotFolder	13-70
SaveFieldsPlots	13-71
SaveNamedExpressions	13-72
SetFieldPlotSettings	13-73
SetPlotFolderSettings	13-76
ShowAntennaParameterOverlay	13-81
UpdateAllFieldsPlots	13-82
UpdateQuantityFieldsPlots	13-83
14 - Fields Calculator Script Commands	14-1
AddNamedExpression	14-2
CalcOp	14-3
CalcRead(deprecated)	14-4
CalculatorRead	14-5
CalcStack	14-6
CalculatorWrite	14-6

ChangeGeomSettings	14-7
ClcEval	14-8
ClcMaterial	14-9
ClcMaterialValue	14-10
ClearAllNamedExpr	14-10
CopyNamedExprToStack	14-11
DeleteNamedExpr	14-12
DoesNamedExpressionExists	14-12
EnterComplex	14-13
EnterComplexVector	14-14
EnterCoord	14-15
EnterEdge	14-16
EnterLine	14-16
EnterOutputVar	14-17
EnterPoint	14-18
EnterQty	14-18
EnterScalar	14-19
EnterScalarFunc	14-20
EnterSurf	14-20
EnterVector	14-21
EnterVectorFunc	14-22
EnterVol	14-22
ExportOnGrid [Fields Calculator]	14-23
ExportToFile [Fields Calculator]	14-26
GetTopEntryValue	14-29
LoadNamedExpressions	14-30
SaveNamedExpressions	14-31
15 - Radiation Module Script Commands	15-1
General Commands Recognized by the Radiation Module	15-2
CopyRadFieldSetup	15-2

DeleteFarFieldSetup	15-3
DeleteNearFieldSetup	15-3
DeleteSetup	15-4
EditRadiatedPowerCalculationMethod	15-5
GetChildNames [Radiation]	15-5
GetChildObject [Radiation]	15-6
GetChildTypes [Radiation]	15-6
GetPropNames [Radiation]	15-7
GetPropValue [Radiation]	15-7
GetSetupNames	15-8
PasteRadFieldSetup	15-8
RenameSetup [Radiation]	15-9
SetPropValue [Radiation]	15-10
Script Commands for Creating and Modifying Radiation Setups	15-11
AddAntennaOverlay	15-11
AddRadFieldSourceGroup	15-13
EditAntennaOverlay	15-14
EditBoxSetup	15-16
EditFarFieldSphereSetup	15-18
EditInfiniteSphereSetup	15-21
EditLineSetup	15-23
EditNearFieldBoxSetup	15-24
EditNearFieldLineSetup	15-26
EditNearFieldRectangleSetup	15-28
EditNearFieldSphereSetup	15-29
EditRadFieldSourceGroup	15-31
EditRectangleSetup	15-33
EditSphereSetup	15-34
InsertBoxSetup	15-37
InsertFarFieldSphereSetup	15-39

InsertInfiniteSphereSetup	15-41
InsertLineSetup	15-43
InsertNearFieldBoxSetup	15-45
InsertNearFieldLineSetup	15-47
InsertNearFieldRectangleSetup	15-48
InsertNearFieldSphereSetup	15-50
InsertRectangleSetup	15-52
InsertSphereSetup	15-53
Script Commands for Modifying Antenna Array Setups	15-55
EditAntennaArraySetup	15-56
Script Commands for Exporting Antenna Parameters and Max Field Parameters	15-61
ExportElementPatternToFile	15-62
ExportFieldsToFile	15-63
ExportParametersToFile	15-65
ExportRadiationFieldsToFile	15-67
ExportRadiationParametersToFile	15-69
16 - Radiation Setup Manager Module Script Commands	16-1
Add (Infinite Sphere)	16-1
Edit (RadiationSetupMgr for Infinite Sphere)	16-5
Add (Near Field Line)	16-8
Edit (RadiationSetupMgr for Near Field Line)	16-10
Add (Near Field Plane)	16-12
Edit (RadiationSetupMgr for Near Field Plane)	16-15
Add (Near Field Sphere)	16-18
Edit (RadiationSetupMgr for Near Field Sphere)	16-22
17 - User Defined Document Script Commands	17-1
AddDocument	17-1
DeleteAllDocuments	17-4
DeleteDocument	17-5
EditDocument	17-5

GetDocumentDefinitionNames	17-7
GetDocumentNames	17-8
RenameDocument	17-8
SaveHtmlDocumentAs	17-9
SavePdfDocumentAs	17-10
UpdateAllDocuments	17-10
UpdateDocument	17-11
ViewHtmlDocument	17-12
ViewPdfDocument	17-13
Explication of a Sample UDD Script	17-13
Example Python Script: Defining a Document	17-15
18 - User Defined Solutions Commands	18-1
CreateUserDefinedSolution	18-1
DeleteUserDefinedSolutions	18-3
EditUserDefinedSolution	18-4
GetUserDefinedSolutionNames	18-6
GetUserDefinedSolutionProperties	18-7
19 - Network Data Explorer Script Commands	19-1
AddDiffPair	19-4
Cascade (SPISim)	19-5
ClearDiffPairs	19-6
Clone	19-7
Close	19-8
Combine (SPISim)	19-9
Deembed (SPISim)	19-10
DeembedBack (SPISim)	19-11
DeembedFront (SPISim)	19-13
DisableDiffPairs	19-14
EnableDiffPairs	19-15
ExportCitiFile	19-16

ExportFullWaveSpice	19-17
ExportMatlab	19-19
ExportNMFData	19-21
ExportNetworkData	19-23
ExportSpreadsheet	19-25
ExportTouchstone	19-28
ExportTouchstone2	19-30
Extract (SPISim)	19-33
GetFrequencies	19-34
GetFrequencyCount	19-35
GetName	19-36
GetPortCount	19-36
GetPortNumber	19-37
GetPostProcSettings	19-39
GetSolutionVariation	19-40
GetVariation	19-41
HasSameData	19-42
LoadSolution	19-43
Open	19-45
Rename (SPISim)	19-45
Renormalize (SPISim)	19-47
Reorder	19-48
Reorder (SPISim)	19-49
Reset	19-50
SetAllPortImpedances	19-51
SetPortDeembedDistance	19-52
SetPortImpedance	19-53
SetPostProcSettings	19-54
Smooth	19-55
Stretch (SPISim)	19-57

Terminate	19-58
20 - ComInstance Script Commands	(multiple)
Callback Scripting Using ComInstance Object	(multiple)
ComInstance Functions	62
GetComponentName	62
GetInstanceID [Component Instance]	63
GetInstanceName [Component Instance]	64
GetParentDesign	64
GetPropHost	65
GetPropServerName	65
21 - Simulation Setup Commands	21-1
Add (Layout Editor)	21-2
AddMeshOperation	21-14
AddSweep (Layout Editor)	21-19
Analyze (Layout Editor)	21-23
AnalyzeSweep (Layout Editor)	21-23
Delete (Layout Editor)	21-23
DeleteGFD (Setup HFSS 3D Layout)	21-24
DeleteSweep (Layout Editor)	21-24
DynamicMeshOverlays (Layout Editor)	21-25
Edit (Layout Editor)	21-25
EditSweep (Layout Editor)	21-37
ExportToHFSS (Layout Editor)	21-41
ExportToQ3D (Layout Editor)	21-41
GetAllSolutionNames (Layout Editor)	21-41
GetGFDDData (Setup HFSS 3D Layout)	21-41
GetMeshOperations	21-42
GetSetupData (Layout Editor)	21-43
LayoutMeshOverlay (Layout Editor)	21-43
Layout3DMeshOverlay (Layout Editor)	21-44

ListVariations (Layout Editor)	21-44
PullExcitationsFromTransient (Layout Editor)	21-44
RefreshMeshOverlays (Layout Editor)	21-45
Rename (Layout Editor)	21-45
RenameSweep (Layout Editor)	21-46
22 - Excitations Commands	22-1
Add [Excitation HFSS 3D Layout]	22-2
AddRefPort (Layout Editor)	22-2
AddRefPortUsingEdges (Layout Editor)	22-2
ConvertPlaneToTrace (Layout Editor)	22-4
ConvertTraceToPlane (Layout Editor)	22-5
CoupleEdgePorts	22-5
CreateGroundPortComponent (Layout Editor)	22-5
CreateInterfaceGround (Layout Editor)	22-6
CreateInterfacePort (Layout Editor)	22-6
CreateInterfacePortComponent (Layout Editor)	22-7
DecoupleEdgePorts	22-7
Delete[Excitation HFSS 3D Layout]	22-7
Delete [Interface Port]	22-7
Delete[Excitation HFSS 3D Layout]	22-8
DeleteProbePortAndVia (Layout Editor)	22-8
DeleteSource [Interface Source]	22-8
Edit (Layout Editor)	22-8
EditCircuitExcitations (Layout Editor)	22-20
EditCircuitPort [Interface Port]	22-21
EditExcitations (Layout Editor)	22-21
GetAllBoundariesList (Layout Editor)	22-23
GetAllPortsList (Layout Editor)	22-24
Rename [Interface Port]	22-24
RenameSource [Interface Source]	22-24

Rename (Layout Editor)	22-24
RemoveRefPort (Layout Editor)	22-25
SelectInLayout (Layout Editor)	22-25
23 - 2.5D Via Commands	23-1
ConvertPrimitives (Layout Editor)	23-1
Delete [Via HFSS 3D Layout]	23-1
Edit [Via HFSS 3D Layout]	23-2
MultipleEdit [multiple via HFSS 3D Layout]	23-2
Rename [Via HFSS 3D Layout]	23-2
SelectInLayout [Via HFSS 3D Layout]	23-3
24 - Cavity Commands	24-1
Add [Cavity HFSS 3D Layout]	24-1
Delete [Cavity HFSS 3D Layout]	24-1
Edit [Cavity HFSS 3D Layout]	24-2
Rename [Cavity HFSS 3D Layout]	24-2
SelectInLayout [Cavity HFSS 3D Layout]	24-2
25 - Layout Scripting	25-1
Object Identifiers and Script Recording	25-2
Create Primitives	25-3
AddObject ("3D Line") (Layout Editor)	25-3
CreateCircle (Layout Editor)	25-6
CreateLine (Layout Editor)	25-8
AddObject ("Partition Line") (IC Editor)	25-10
CreatePolygon (Layout Editor)	25-14
CreateRectangle (Layout Editor)	25-15
CreateText (Layout Editor)	25-17
Create Voids in Primitives	25-20
CreateCircleVoid (Layout Editor)	25-20
CreateLineVoid (Layout Editor)	25-21
CreatePolygonVoid (Layout Editor)	25-22

CreateRectangleVoid (Layout Editor)	25-24
GetPolygonVoids (Layout Editor)	25-25
Other Creation Methods	25-25
AddPinGroupRefPort (Layout Editor)	25-26
CreateComponent (Layout Editor)	25-27
CreateCurrentSourcePort (Layout Editor)	25-28
CreateHole (Layout Editor)	25-29
CreateMeasure (Layout Editor)	25-31
CreatePin (Layout Editor)	25-33
CreatePinGroup (Layout Editor)	25-34
CreatePinGroupPort (Layout Editor)	25-35
CreatePinGroups (Layout Editor)	25-35
CreateTrace (Layout Editor)	25-36
CreateVia (Layout Editor)	25-42
CreateViaGroups (Layout Editor)	25-43
CreateVoltageProbePort (Layout Editor)	25-45
CreateVoltageSourcePort (Layout Editor)	25-46
DeletePinGroup (Layout Editor)	25-47
Object Movement and Modification Methods	25-47
Connect (Layout Editor)	25-48
Copy (Layout Editor)	25-48
Delete (Layout Editor)	25-49
Disconnect (Layout Editor)	25-49
Edit (Layout Editor)	25-49
Expand (Layout Editor)	25-51
ExpandWithPorts (Layout Editor)	25-52
FlipHorizontal (Layout Editor)	25-53
FlipVertical (Layout Editor)	25-53
Move (Layout Editor)	25-54
Paste (Layout Editor)	25-54

Rotate (Layout Editor)	25-54
Activation and Deactivation Methods	25-55
Activate (Layout Editor)	25-55
DeactivateOpen (Layout Editor)	25-55
DeactivateShort (Layout Editor)	25-56
Delete (Layout Editor)	25-56
Layout and Geometry Interrogation Methods	25-57
Layout Interrogation (Layout Editor)	25-57
CreateLineFromPolygon (Layout Editor)	25-58
CreateObjectFromPolygon (Layout Editor)	25-58
FindObjects (Layout Editor)	25-59
FilterObjectList (Layout Editor)	25-60
FindObjectsByPoint (Layout Editor)	25-61
FindObjectsByPolygon (Layout Editor)	25-61
GetBBox (Layout Editor)	25-62
GetCSObjects (Layout Editor)	25-62
GetPolygon (Layout Editor)	25-63
GetPolygonDef (Layout Editor)	25-63
Point (Layout Editor)	25-63
Polygon (Layout Editor)	25-63
Point Object (Layout Editor)	25-64
Set (Layout Editor)	25-65
SetX (Layout Editor)	25-65
GetX (Layout Editor)	25-65
SetY (Layout Editor)	25-66
GetY (Layout Editor)	25-66
SetArc (Layout Editor)	25-66
IsArc (Layout Editor)	25-67
IsEqual (Layout Editor)	25-67
Mag (Layout Editor)	25-67

Distance (Layout Editor)	25-67
Cross (Layout Editor)	25-68
Move (Layout Editor)	25-68
Rotate (Layout Editor)	25-68
Normalize (Layout Editor)	25-69
DistanceFromLine (Layout Editor)	25-69
ClosestPointOnLine (Layout Editor)	25-69
Polygon Object (Layout Editor)	25-69
AddPoint (Layout Editor)	25-71
SetClosed (Layout Editor)	25-72
IsClosed (Layout Editor)	25-72
Move (Layout Editor)	25-72
Rotate (Layout Editor)	25-73
Scale (Layout Editor)	25-73
MirrorX (Layout Editor)	25-74
GetPoints (Layout Editor)	25-74
AddHole (Layout Editor)	25-74
HasHoles (Layout Editor)	25-74
GetHoles (Layout Editor)	25-75
HasArcs (Layout Editor)	25-75
HasSelfIntersections (Layout Editor)	25-75
BBoxLL (Layout Editor)	25-76
BBoxUR (Layout Editor)	25-76
IsParametric (Layout Editor)	25-76
IsConvex (Layout Editor)	25-76
IsPoint (Layout Editor)	25-77
IsSegment (Layout Editor)	25-77
IsArc (Layout Editor)	25-77
IsBox (Layout Editor)	25-77
IsCircle (Layout Editor)	25-78

GetBoundingCircleCenter (Layout Editor)	25-78
GetBoundingCircleRadius (Layout Editor)	25-78
Area (Layout Editor)	25-78
PointInPolygon (Layout Editor)	25-79
CircleIntersectsPolygon (Layout Editor)	25-79
GetIntersectionType (Layout Editor)	25-79
GetClosestPoint (Layout Editor)	25-80
GetClosestPoints (Layout Editor)	25-80
Unite (Layout Editor)	25-80
Intersect (Layout Editor)	25-81
Subtract (Layout Editor)	25-81
Xor (Layout Editor)	25-82
Boolean Operations on Primitives	25-82
ConvertPrimitives (Layout Editor)	25-82
ConvertPrimitivesToVias (Layout Editor)	25-83
Unite (Layout Editor)	25-84
Intersect (Layout Editor)	25-84
SplitRegion (Layout Editor)	25-85
Subtract (Layout Editor)	25-85
ToggleViaPin (Symbol Editor)	25-86
Coordinate System Methods	25-86
CreateCS (Layout Editor)	25-87
ClearRelative (Layout Editor)	25-91
Create3DStructure (Layout Editor)	25-91
GetCSObjects (Layout Editor)	25-92
Group (Layout Editor)	25-92
CreateGroupSelected (Layout Editor)	25-92
PositionRelative (Layout Editor)	25-92
SetCS (Layout Editor)	25-96
Ungroup (Layout Editor)	25-96

IC Editor Script Commands (IC Mode Only)	25-96
Common to General Mode	25-98
ApplyTechnology (IC Editor)	25-103
ChangeActiveLayout (IC Editor)	25-103
CreateMeshRegion (IC Editor)	25-105
CreateMeshRegionPartitionLine (IC Editor)	25-107
CreateViaGroupsOnLayer (IC Editor)	25-112
ReconstructArcs (IC Editor)	25-114
ReconstructArcsOnLayer (IC Editor)	25-115
RemoveSmallHoles (IC Editor)	25-116
RemoveSmallLines (IC Editor)	25-117
RemoveSmallMetalIslands (IC Editor)	25-118
ResetTechnology (IC Editor)	25-119
SnapPrimitives (IC Editor)	25-119
SnapPrimitivesOnLayer (IC Editor)	25-121
SnapVias (IC Editor)	25-121
SnapViasOnLayer (IC Editor)	25-124
UnitePrimitivesOnLayer (IC Editor)	25-126
WrapGeometries (IC Editor)	25-126
WrapGeometry (IC Editor)	25-128
Layers Methods	25-128
AddObject (Layout Editor and IC Editor)	25-129
AddLayer (Layout Editor)	25-131
AddStackupLayer (Layout Editor)	25-135
ChangeLayer (Layout Editor)	25-139
ClearLayerMappings (Layout Editor)	25-144
DuplicateAcrossLysrs (Layout Editor)	25-144
ExportGDSII (Layout Editor)	25-145
ExportStackupXML (Layout Editor)	25-148
GetAllLayerNames (Layout Editor)	25-148

GetLayerInfo (Layout Editor)	25-149
GetStackupLayerNames (Layout Editor)	25-151
ImportStackupXML (Layout Editor)	25-151
RemoveLayer (Layout Editor)	25-152
General Mode	25-152
IC Mode (IC Editor)	25-153
SetLayerMapping (Layout Editor)	25-154
SyncRigidFlexSandbox (Layout Editor)	25-155
Net and NetClass Methods	25-156
CreateDifferentialPair (Layout Editor)	25-157
CreateNetClass	25-158
DeleteNets (Layout Editor)	25-159
DelNetClass	25-160
GetNetClasses	25-160
GetNetClassNets	25-160
GetNetConnections (Layout Editor)	25-161
GetNets (Layout Editor)	25-162
HighlightNet (Layout Editor)	25-162
ModifyDifferentialPair (Layout Editor)	25-164
ModifyNetClass	25-165
ResetDifferentialpairs (Layout Editor)	25-165
SelectNetConnected (Layout Editor)	25-168
SelectPhysicallyConnected (Layout Editor)	25-169
SetNetColor (Layout Editor)	25-170
SetNetVisible (Layout Editor)	25-172
ToggleNetHighlight (Layout Editor)	25-173
Port Methods	25-175
AddCircuitRefPort (Layout Editor)	25-176
AddPortsToAllNets (Layout Editor)	25-178
AddPortsToNet (Layout Editor)	25-178

AddRefPort (Layout Editor)	25-179
AlignPorts (Layout Editor)	25-181
AssignCircuitRefPort (Layout Editor)	25-182
AssignRefPort (Layout Editor)	25-184
ClearRefPort (Layout Editor)	25-186
CreateCircuitPort (Layout Editor)	25-187
CreateEdgePort (Layout Editor)	25-189
CreateNPort (Layout Editor)	25-192
CreatePortInstancePorts (Layout Editor)	25-195
CreatePortsOnComponents (Layout Editor)	25-195
CreatePortsOnComponentsByNet (Layout Editor)	25-196
GetPortInfo (Layout Editor)	25-198
RemovePort (Symbol Editor)	25-200
RemovePortsFromAllNets (Layout Editor)	25-201
RemovePortsFromNet (Layout Editor)	25-201
RemovePortsOnComponents (Layout Editor)	25-202
SetHfssExtentsVisible (Layout Editor)	25-203
SetHfssPortsVisible (Layout Editor)	25-204
Miscellaneous Methods	25-205
AlignObjects (Layout Editor)	25-207
ChangeOptions (Layout Editor)	25-208
ChangeProperty (Schematic Editor and Layout Editor)	25-213
ClipPlane (Layout Editor)	25-223
CopyToHFSS 3D Layout (Layout Editor)	25-224
CutOutSubDesign (Layout Editor)	25-224
Defeature Objects (Layout Editor)	25-229
DissolveComponents (Layout Editor)	25-230
Duplicate (Layout Editor)	25-231
Edit3DComponentDefinition	25-232
EditComponent (Layout Editor)	25-233

EditPinGroup (Layout Editor)	25-235
EnableComponents (Layout Editor)	25-237
EraseMeasurements (Layout Editor)	25-238
ExportDXF (Layout Editor)	25-238
ExportGDSII (Layout Editor)	25-240
ExportGerber (Layout Editor)	25-243
ExportNCDrill (Layout Editor)	25-245
GetActiveUnits (Layout Editor)	25-248
GetComponentInfo (Layout Editor)	25-249
GetCompInstanceFromRefDes (Layout Editor)	25-250
GetComponentPins (Layout Editor)	25-250
GetComponentPinInfo (Layout Editor)	25-251
GetEditorName (Layout Editor)	25-252
GetMaterialList (Layout Editor)	25-252
GetProperties (Layout Editor)	25-253
GetPropertyValue (Layout Editor)	25-253
GetSelections (Layout Editor)	25-254
GetSelectionsInHierarchy (Layout Editor)	25-254
Heal (Layout Editor)	25-255
PageSetup (Layout Editor)	25-257
PushExcitations	25-258
Select (Layout Editor)	25-261
SelectAll (Layout Editor)	25-262
SetActiveUnits (Layout Editor)	25-263
SetPropertyValue (Layout Editor)	25-263
StitchLines (Layout Editor)	25-264
UnselectAll (Layout Editor)	25-265
ZoomToFit (Layout Editor)	25-265
26 - Definition Manager Script Commands	26-1
Add [component manager]	26-3

AddDataset	26-22
AddDefinitionFromBlock	26-25
AddMaterial	26-30
Add [padstack manager]	26-34
Add [symbol manager]	26-40
AreMaterialPropertiesEqual	26-51
AreSurfaceMaterialPropertiesEqual	26-52
ChangeProperty (Symbol Editor)	26-53
CloneMaterial	26-66
DeleteDataset	26-67
DoesMaterialExist	26-67
Edit [component manager]	26-68
EditDataset	26-95
EditMaterial	26-97
Edit [padstack manager]	26-106
ExportDataset	26-113
Export [footprint manager]	26-114
ExportMaterial	26-115
Export [padstack manager]	26-116
ExportScript	26-117
Export [symbol manager]	26-118
GetProjextMaterialNames	26-119
GetPropertyValue	26-120
ImportDataset	26-121
Note About File Types:	26-123
Remove [component manager]	26-124
Remove [footprint manager]	26-126
RemoveMaterial	26-127
Remove [padstack manager]	26-128
RemoveScript	26-129

Remove [symbol manager]	26-130
RemoveUnusedDefinitions	26-131
SetPropertyValue	26-133
UpdateDefFromBlock	26-134
UpdateDefFromBlockEx	26-136
UpdateDefinitions	26-139
Component Manager Script Commands	26-140
Add [component manager]	26-141
AddNPortData [component manager]	26-159
AddSolverOnDemandModel	26-165
ClearSolutionCache [component manager]	26-165
Edit [component manager]	26-166
EditSolverOnDemandModel	26-193
EditWithComps [component manager]	26-193
Export [component manager]	26-205
GetData [component manager]	26-206
GetNames [component manager]	26-207
GetNPortData [component manager]	26-207
GetSolverOnDemandData	26-210
GetSolverOnDemandModelList	26-211
IsUsed [component manager]	26-211
Remove [component manager]	26-212
RemoveSolverOnDemandModel	26-213
RemoveUnused [component manager]	26-213
Update Dynamic Link [component manager]	26-214
Footprint Manager Script Commands	26-215
Add [footprint manager]	26-215
Edit [footprint manager]	26-238
EditWithComps [footprint manager]	26-238
Export [footprint manager]	26-256

GetData [footprint manager]	26-257
GetNames [footprint manager]	26-258
IsUsed [footprint manager]	26-258
Remove [footprint manager]	26-259
RemoveUnused [footprint manager]	26-260
Material Manager Script Commands	26-261
GetData [material manager]	26-261
GetNames [material manager]	26-262
GetProperties [material manager]	26-264
IsUsed [material manager]	26-264
RemoveUnused [material manager]	26-265
Model Manager Script Commands	26-266
Add [model manager]	26-266
ConvertToDynamic	26-277
ConvertToParametric	26-277
Edit [deprecated]	26-278
EditWithComps [model manager]	26-278
Export [model manager]	26-289
GetData [model manager]	26-291
GetNames [model manager]	26-292
IsUsed [model manager]	26-292
Remove [model manager]	26-293
RemoveUnused [model manager]	26-294
Network Data Explorer Manager Script Commands	26-295
ExportFullWaveSpice	26-296
ExportNetworkData	26-297
ExportNMFData	26-300
Padstack Manager Script Commands	26-302
Add [padstack manager]	26-302
Edit [padstack manager]	26-309

EditWithComps [padstack manager]	26-316
Export [padstack manager]	26-323
GetData [padstack manager]	26-324
GetNames [padstack manager]	26-325
IsUsed [padstack manager]	26-325
Remove [padstack manager]	26-326
RemoveUnused [padstack manager]	26-327
Script and Library Scripts	26-328
AddScript	26-328
EditScript	26-329
ExportScript	26-330
ModifyLibraries	26-331
RemoveScript	26-332
Symbol Manager Script Commands	26-333
Add [symbol manager]	26-334
BringToFront [symbol manager]	26-345
Edit [deprecated]	26-345
EditWithComps [symbol manager]	26-345
Export [symbol manager]	26-357
GetData [symbol manager]	26-358
GetNames [symbol manager]	26-359
IsUsed [symbol manager]	26-359
Remove [symbol manager]	26-360
RemoveUnused [symbol manager]	26-361
Technology Manager Script Commands	26-362
Add (Technology Manager)	26-362
GetData (Technology Manager)	26-364
GetNames (Technology Manager)	26-365
IsUsed (Technology Manager)	26-365
Remove (Technology Manager)	26-366

27 - Definition Editor Script Commands	27-1
Activate (Layout Editor)	27-7
AddCircuitRefPort (Layout Editor)	27-8
AddLayer (Layout Editor)	27-10
AddPortsToAllNets (Layout Editor)	27-14
AddPortsToNet (Layout Editor)	27-14
AddRefPort (Layout Editor)	27-14
AlignObjects (Layout Editor)	27-17
AlignPorts (Layout Editor)	27-18
AssignCircuitRefPort (Layout Editor)	27-19
AssignRefPort (Layout Editor)	27-21
ChangeProperty	27-24
ClearLayerMappings (Layout Editor)	27-28
ClearRefPort (Layout Editor)	27-29
ClearRelative (Layout Editor)	27-29
Connect (Layout Editor)	27-29
ConvertPlaneToTrace (Layout Editor)	27-30
ConvertTraceToPlane (Layout Editor)	27-30
Copy (Layout Editor)	27-30
CopyToHFSS 3D Layout (Layout Editor)	27-31
Create3DStructure (Layout Editor)	27-31
CreateCS (Layout Editor)	27-32
CreateCircle (Layout Editor)	27-36
CreateCircleVoid (Layout Editor)	27-37
CreateCircuitPort (Layout Editor)	27-39
CreateCircuitPortsOnComponents (Layout Editor)	27-41
CreateComponent (Layout Editor)	27-43
CreateEdgePort (Layout Editor)	27-44
CreateGroundPortComponent (Layout Editor)	27-47
CreateGroupSelected (Layout Editor)	27-47

CreateHole (Layout Editor)	27-48
CreateInterfaceGround (Layout Editor)	27-49
CreateInterfacePort (Layout Editor)	27-50
CreateInterfacePortComponent (Layout Editor)	27-50
CreateLine (Layout Editor)	27-50
CreateLineFromPolygon (Layout Editor)	27-53
CreateLineVoid (Layout Editor)	27-53
CreateMeasure (Layout Editor)	27-54
CreateNportCircuitElements	27-57
CreateNetClass	27-59
CreateObjectFromPolygon (Layout Editor)	27-59
CreatePin (Layout Editor)	27-60
CreatePinGroup (Layout Editor)	27-61
CreatePinGroupPort (Layout Editor)	27-61
CreatePinGroupPortsOnComponents (Layout Editor)	27-61
CreatePolygon (Layout Editor)	27-63
CreatePolygonVoid (Layout Editor)	27-64
CreatePortInstancePorts (Layout Editor)	27-65
CreatePortsOnComponents (Layout Editor)	27-66
CreateRectangle (Layout Editor)	27-66
CreateRectangleVoid (Layout Editor)	27-68
CreateText (Layout Editor)	27-69
CreateTrace (Layout Editor)	27-72
CreateVia (Layout Editor)	27-78
CutOutSubDesign (Layout Editor)	27-79
DeactivateOpen (Layout Editor)	27-84
DeactivateShort (Layout Editor)	27-84
Defeature Objects (Layout Editor)	27-85
DelNetClass	27-86
Delete (Layout Editor)	27-86

DeleteNets (Layout Editor)	27-86
DeletePinGroup (Layout Editor)	27-87
Disconnect (Layout Editor)	27-88
Duplicate (Layout Editor)	27-88
DuplicateAcrossLyrs (Layout Editor)	27-89
Edit (Layout Editor)	27-89
Edit3DComponentDefinition	27-91
EraseMeasurements (Layout Editor)	27-92
Expand (Layout Editor)	27-92
ExpandWithPorts (Layout Editor)	27-93
ExportDXF (Layout Editor)	27-94
ExportGDSII (Layout Editor)	27-97
ExportGerber (Layout Editor)	27-99
ExportNCDrill (Layout Editor)	27-102
ExportToHFSS (Layout Editor)	27-105
ExportToQ3D (Layout Editor)	27-105
FilterObjectList (Layout Editor)	27-105
FindObjectsByPoint (Layout Editor)	27-106
FindObjectsByPolygon (Layout Editor)	27-107
FlipHorizontal (Layout Editor)	27-107
FlipVertical (Layout Editor)	27-108
GeometryCheckAndAutofix	27-108
GetAllLayerNames (Layout Editor)	27-110
GetBBox (Layout Editor)	27-110
GetCSObjects (Layout Editor)	27-111
GetCompInstanceFromRefDes (Layout Editor)	27-111
GetComponentInfo (Layout Editor)	27-111
GetComponentPinInfo (Layout Editor)	27-112
GetComponentPins (Layout Editor)	27-113
GetEditorName (Layout Editor)	27-114

GetMaterialList (Layout Editor)	27-115
GetNetClassNets	27-115
GetNetClasses	27-115
GetNetConnections (Layout Editor)	27-116
GetPolygon (Layout Editor)	27-117
GetPolygonDef (Layout Editor)	27-117
GetPortInfo (Layout Editor)	27-117
GetProperties (Layout Editor)	27-120
GetPropertyValue (Layout Editor)	27-121
GetSelections [Design]	27-121
Group (Layout Editor)	27-121
Heal (Layout Editor)	27-121
HighlightNet (Layout Editor)	27-124
ImportDXF [Modeler]	27-125
ImportGDSII	27-129
Intersect (Layout Editor)	27-131
ModifyNetClass	27-131
Move (Layout Editor)	27-132
PageSetup (Layout Editor)	27-132
Point (Layout Editor)	27-133
Polygon (Layout Editor)	27-133
PositionRelative (Layout Editor)	27-133
PushExcitations	27-136
RemoveLayer (Layout Editor)	27-140
RemovePortsFromAllNets (Layout Editor)	27-140
RemovePortsFromNet (Layout Editor)	27-141
RemovePortsOnComponents (Layout Editor)	27-142
Rotate (Layout Editor)	27-143
SelectAll (Layout Editor)	27-143
SetCS (Layout Editor)	27-144

SetLayerMapping (Layout Editor)	27-144
SetNetVisible (Layout Editor)	27-144
StitchLines (Layout Editor)	27-146
Subtract (Layout Editor)	27-147
ToggleViaPin (Footprint Editor)	27-147
Ungroup (Layout Editor)	27-148
Unite (Layout Editor)	27-148
UnselectAll (Layout Editor)	27-149
ZoomToFit (Layout Editor)	27-149
Symbol Editor Scripts	27-149
AddLevel (Symbol Editor)	27-150
AlignHorizontal (Symbol Editor)	27-151
AlignVertical (Symbol Editor)	27-152
BringToFront (Symbol Editor)	27-153
ChangeProperty (Symbol Editor)	27-153
CloseEditor (Symbol Editor)	27-166
Copy (Symbol Editor)	27-167
CreateArc (Symbol Editor)	27-167
CreateCircle (Symbol Editor)	27-168
CreateImage (Symbol Editor)	27-169
CreateLine (Symbol Editor)	27-170
CreatePin (Symbol Editor)	27-171
CreatePolygon (Symbol Editor)	27-172
CreateRectangle (Symbol Editor)	27-173
CreateText (Symbol Editor)	27-173
Cut (Symbol Editor)	27-174
Delete (Symbol Editor)	27-175
DisplayPinNames (Symbol Editor)	27-175
ExportFile (Symbol Editor)	27-176
FlipHorizontal (Symbol Editor)	27-177

FlipVertical (Symbol Editor)	27-177
GetProperties (Symbol Editor)	27-178
GetPropertyValue (Symbol Editor)	27-178
GetVisibleLevels (Symbol Editor)	27-179
GridSetup (Symbol Editor)	27-179
Move (Symbol Editor)	27-181
Pan (Symbol Editor)	27-182
Paste (Symbol Editor)	27-182
Redo (Symbol Editor)	27-183
RemoveLevels (Symbol Editor)	27-183
RemovePort (Symbol Editor)	27-184
Rotate (Symbol Editor)	27-185
Save (Symbol Editor)	27-185
SelectAll (Symbol Editor)	27-186
SendToBack (Symbol Editor)	27-187
SetActiveLevel (Symbol Editor)	27-187
SetInitialLevels (Symbol Editor)	27-188
SetPropertyValue (Symbol Editor)	27-189
SetVisibleLevels (Symbol Editor)	27-189
ToggleLevel (Symbol Editor)	27-190
ToggleViaPin (Symbol Editor)	27-191
Undo (Symbol Editor)	27-191
ZoomArea (Symbol Editor)	27-192
ZoomIn (Symbol Editor)	27-193
ZoomOut (Symbol Editor)	27-193
ZoomPrevious (Symbol Editor)	27-194
ZoomToFit (Symbol Editor)	27-194
Footprint Editor Scripts	27-195
AddLayer (Footprint Editor)	27-197
AddStackupLayer (Footprint Editor)	27-200

ChangeLayer (Footprint Editor)	27-205
ChangeLayers (Footprint Editor)	27-209
ChangeOptions (Footprint Editor)	27-216
CloseEditor (Footprint Editor)	27-221
CreateCircle (Footprint Editor)	27-222
CreateCircleVoid (Footprint Editor)	27-222
CreateEdgePort (Footprint Editor)	27-223
CreateLine (Footprint Editor)	27-226
CreateLineVoid (Footprint Editor)	27-226
CreateMeasure (Footprint Editor)	27-227
CreatePolygonVoid (Footprint Editor)	27-230
CreatePin (Footprint Editor)	27-231
CreatePolygon (Footprint Editor)	27-231
CreateRectangle (Footprint Editor)	27-232
CreateText (Footprint Editor)	27-233
CreateVia (Footprint Editor)	27-233
Duplicate (Footprint Editor)	27-234
Edit (Footprint Editor)	27-235
EraseMeasurements (Footprint Editor)	27-235
FlipHorizontal (Footprint Editor)	27-236
FlipVertical (Footprint Editor)	27-236
GetAllLayerNames (Footprint Editor)	27-237
GetLayerInfo (Footprint Editor)	27-237
GetProperties (Footprint Editor)	27-239
GetStackupLayerNames (Footprint Editor)	27-240
Intersect (Footprint Editor)	27-240
Move (Footprint Editor)	27-240
PageSetup (Footprint Editor)	27-241
RemoveLayer (Footprint Editor)	27-242
RemovePort (Footprint Editor)	27-242

Rotate [Footprint Editor]	27-243
Save (Footprint Editor)	27-243
SetActiveDefinitionEditor (Footprint Editor)	27-244
SetPropertyValue (Footprint Editor)	27-244
Subtract (Footprint Editor)	27-245
ToggleViaPin (Footprint Editor)	27-245
Unite (Footprint Editor)	27-246
ZoomToFit (Footprint Editor)	27-246
28 - Design Verification Script Commands	28-1
AddRuleSet	28-1
AddRun	28-2
DeleteRuleSet	28-5
DeleteRun	28-5
EditRuleSet	28-5
EditRun	28-7
RenameRuleSet	28-9
RenameRun	28-10
RunAllDV	28-10
RunAllRuleSetDV	28-11
RunDV	28-11
29 - Core Global Script Context Commands	29-1
AddErrorMessage	29-1
AddFatalMessage	29-2
AddInfoMessage	29-2
AddWarningMessage	29-3
LogDebug	29-3
LogError	29-4
30 - Example Scripts	30-1
VBScript Example Scripts	30-1
Data Export Script	30-1

Variable Helix Script 30-4

GetPropNames and GetPropValues for Layered Impedance Boundary Script 30-8

IronPython Example Scripts 30-10

HFSS Waveguide Array Python Script 30-11

Index Index-1

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.]

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

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

VB Syntax	[Correct syntax for the command in VBscript. Arguments are enclosed in angle brackets (<>)]
VB Example	[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" 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](#).

String Concatenation Operator

The ampersand symbol (&) is used to perform string concatenation. This operator appends the second string to the first string.

Example:

```
"Hello," & " isn't it a lovely day?"
```

produces the following resultant string:

```
"Hello, isn't it a lovely day?"
```

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:

CStr	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".
CBool	Syntax: CBool(variablename). Converts variablename to a boolean. If variablename is 0 or "0", CBool returns False. Otherwise it returns True.
CDbl	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.
CInt	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.

Sample HFSS Script

Following is an example of an Ansys Electronics Desktop script. It includes comment lines, which are preceded by either an apostrophe (') or the word REM, that offer explanations for each preceding line or lines. VBScript keywords appear in bold font.

```
' -----
```

```
' Script Recorded by Ansoft HFSS Version 10.0
```

```
' 11:03 AM May 3, 2005
```

```
' -----
```

```
Dim oDesign
```

```
Dim oEditor
```

```
Dim oModule
```

REM **Dim** is used to declare variables. **Dim** means dimension. In VBScript you can use **Dim**, **REMPublic**, or **Private** to declare variables. As VBScript has no built-in data types (like REMinteger, string, etc.), all variables are treated as variants, which can store any type of REMinformation. In this example, the three variables will be used as objects. When REMrecording scripts in HFSS, variants that will be used as objects always begin with o.

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

```
' You can use Set to assign an object reference to a variable. A copy of the object is not  
' created for that variable. Here CreateObject is a function that takes a string as input  
' and returns an object. The object is assigned to the variable oAnsoftApp.
```

```
Set oDesktop = oAnsoftApp.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.
```

```
oDesktop.NewProject
```

```
' In VBScript, a Sub procedure is a procedure that is called by name, can receive arguments,  
' and can perform a specific task with a group of statements. Here the Sub procedure
```

' NewProject of the object oDesktop is called. This Sub does not take an input.

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

```
oProject.InsertDesign "Hfss", "HFSSDesign1", "DrivenModal", ""
```

' In a Sub or Function procedure call, you can group the input parameters inside

' parentheses or without parentheses. Here the four strings are the input parameters of

' the Sub procedure InsertDesign of the object oProject.

```
Set oDesign = oProject.SetActiveDesign("HFSSDesign1")
```

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

```
oEditor.CreateBox Array("NAME:BoxParameters", "XPosition:=", _  
    "0mm", "YPosition:=", "0mm", "ZPosition:=", "0mm", _  
    "XSize:=", "1.6mm", "YSize:=", "1.2mm", "ZSize:=", _  
    "0.8mm"), Array("NAME:Attributes", "Name:=", "Box1", "Flags:=", _  
    "", "Color:=", "(132 132 193)", "Transparency:=", _  
    0.400000005960464, "PartCoordinateSystem:=", _  
    "Global", "MaterialName:=", "vacuum", "SolveInside:=", true)
```

' oEditor.CreateBox is a Sub procedure that takes two array variables as input. The

' first array is for the box's geometric parameters and the second array is for the box's

' attributes. You can modify the italicized entries to create a different box. In VBScript,

' **Array** is a function that returns a variant containing an array. The underscore

' character (_) here indicates that the statement continues to the next line. The

' underscore character must be placed outside of string constants, or else VBScript will

' recognize the character as part of the string constant rather than an indication that the

' string continues on the next line. Following is an example of proper use of the underscore

' character:

```
' MsgBox("Please include units when creating variables " & _  
' "that require dimensions."
```

' Following is an example of improper use of the underscore character:

```
' MsgBox("Please include units when creating variables _
```

```
' that require dimensions."
```

For additional Ansys Electronics Desktop script examples, see [Example Scripts](#).

Sample Q3D Extractor Script

Following is an example Q3D Extractor script. It includes comment lines, preceded by either an apostrophe or the word REM, that offer explanations for each preceding line or lines. VBScript keywords appear in bold.

```
Dim oAnsoftApp
```

```
Dim oDesktop
```

```
Dim oProject
```

```
Dim oDesign
```

```
Dim oEditor
```

```
Dim oModule
```

REM **Dim** is used to declare variables and means dimension. In VBScript you can use **Dim**, **REMPublic**, or **Private** to declare variables. As VBScript has no built-in data types (like REMinteger, string, etc.), all variables are treated as variants, which can store any type of REM information. In this example, the three variables will be used as objects. When REM recording scripts in Q3D Extractor, variants that will be used as objects always begin with o.

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

' You can use **Set** to assign an object reference to a variable. A copy of the object is not ' created for that variable. Here CreateObject is a function that takes a string as input ' and returns an object. The object is assigned to the variable oAnsoftApp.

```
Set oDesktop = oAnsoftApp.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.

```
oDesktop.NewProject
```

' In VBScript, a Sub procedure is a procedure that is called by name, can receive arguments, ' and can perform a specific task with a group of statements. Here the Sub procedure ' NewProject of the object oDesktop is called. This Sub does not take an input.

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

```
oProject.InsertDesign "Q3D Extractor", "Q3DDesign1", "", ""
```

' In a Sub or Function procedure call, you can group the input parameters inside ' parentheses or without parentheses. Here the four strings are the input parameters of ' the Sub procedure InsertDesign of the object oProject.

```
Set oDesign = oProject.SetActiveDesign("Q3DDesign1")
Set oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.CreateBox Array("NAME:BoxParameters", "XPosition:=", _
    "0mm", "YPosition:=", "0mm", "ZPosition:=", "0mm", _
    "XSize:=", "1.6mm", "YSize:=", "1.2mm", "ZSize:=", _
    "0.8mm"), Array("NAME:Attributes", "Name:=", "Box1", "Flags:=", _
    "", "Color:=", "(132 132 193)", "Transparency:=", _
    0.400000005960464, "PartCoordinateSystem:=", _
    "Global", "MaterialName:=", "vacuum", "SolveInside:=", true)
```

' oEditor.CreateBox is a Sub procedure that takes two array variables as input. The ' first array is for the box's geometric parameters, and the second array is for the box's ' attributes. You can modify the italicized entries to create a different box. In VBScript, ' **Array** is a function that returns a variant containing an array. The underscore ' character (_) here indicates that the statement continues to the next line. The ' underscore character must be placed outside of string constants, or else VBScript ' recognizes the character as part of the string constant rather than an indication that the ' string continues on the next line. Following is an example of proper use of the underscore ' character:

```
' MsgBox("Please include units when creating variables " & _
' "that require dimensions."
```

' Following is an example of improper use of the underscore character:

```
' MsgBox("Please include units when creating variables _
' that require dimensions."
```

For additional Ansys Electronics Desktop script examples, see [Example 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.

[\(http://code.google.com/p/ironclad/\)](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**.



```

=====
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: Dim doc Assignment then needs a Set instruction: Set doc = app.GetActiveProject()	No Set syntax. Simply create and assign: doc = app.GetActiveProject()

Create Lists/Arrays

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

Create Dictionaries/Maps

VBScript	IronPython
Declare with a Dim: Dim dict Use the CreateObject function with ProgID Scripting.Dictionary: Set dict = CreateObject _ ("Scripting.Dictionary") Add items using the object, key, item syntax: dict.Add key, item	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> Delimiters are curly braces. Use a colon between the key and the value. Separate key value pairs with a comma: <pre>myDict = { "a" : 1, "b" : "hello there", "c" : [1, 2, "abc"] }</pre>

Boolean Values

VBScript	IronPython
Boolean literals are in lower case: true false	The first letter is capitalized: True False

Converting Numbers to Strings and Vice Versa

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

String Formatting/Concatenation

VBScript	IronPython
String concatenation uses the & operator: <pre>Dim allStr, str1 str1 = " how are you" allStr = "Hello " & " There" & str1</pre> There seems to be no direct string formatting function in VBScript. Using string concatenation or using Replace are the two built-in options: <pre>Dim fmt = "{1} climbs stalk {2}" Dim str = Replace(fmt, "{1}", "jack") str = Replace(str, "{2}", 10)</pre>	If you have two strings, you can always concatenate them using the '+' operator: <pre>str1 = "hello" str2 = "world" str12 = str1 + " " + str2</pre> 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,): <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 " -> " process(i)</pre>

Looping over a Range

VBScript	IronPython
To loop over a range, specify start, end, and step: <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 `__somenam__` 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("Ouin")), 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 main the user-entered order. Every item in a function block is expect 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](#)

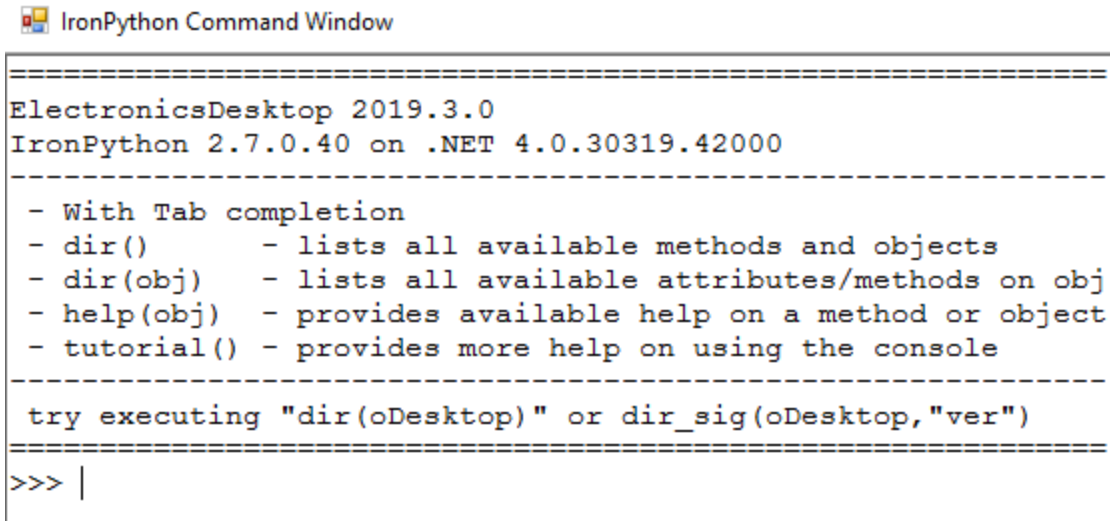
[Translating Script commands from VBScript to IronPython](#)

[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.
 - HFSS -runscript someScript.py to keep HFSS GUI open after completing script execution.
 - HFSS -features=beta -ng -runscriptandexit someScript.py to run HFSS in a non-graphical mode and exit after script completion. Note that this is a beta feature supported for the HFSS and 3D Layout design types.
- 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 subtopics:

[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.

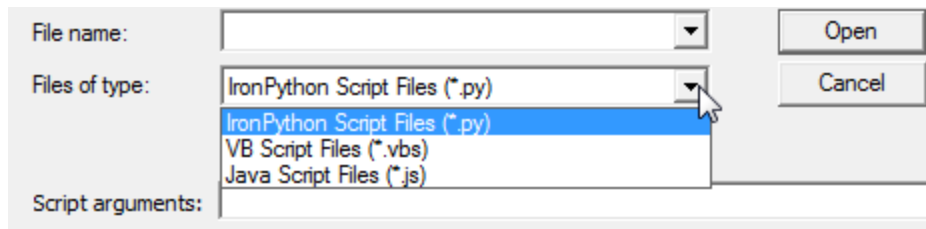


Figure 1: Run Script dialog and script arguments

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\ANSYS Inc\v251\AnsysEM\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\ANSYS Inc\v251\AnsysEM")

sys.path.append(r"C:\Program Files\ANSYS Inc\v251\An-
sysEM\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 Kises", "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 [UDMFunctionLibrary](#) 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 tupe, in order**

```
(lRet, partIdsList) = udpFunctionLibrary.DetachFaces(nPartIds, faceIdsList)
```

Two output parameter; return value must be preserved

the return tuple is (returnVal, output1, output2)

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


Comparison with C function:

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

'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
<pre>bool getAllFaces(long partId, long* numFaces, long** facelds, void* callbackData);</pre> where numFaces and facelds are output parameters and numFaces is the size of faceld.	<pre>facelds = udmFunctionLibrary.GetAllFaces(partId)</pre> (ignore numFaces as redundant: folded into facelds, return value is omitted: folded into the facelds is None check callbackData is omitted)
<pre>long unite(long numObjects, long* objectIds, void* callbackData);</pre> where numObjects and objectIds are input parameters and numObjects is the size of objectIds.	<pre>lret = udpFunctionLibrary.Unite(objectIds)</pre> (ignore numObjects as redundant: folded into objectIds callbackData is omitted)
<pre>long detachFaces(long nSize, long* facelds, long* partIds, void* callbackData);</pre> where partIds is and output list and nSize is an input parameters and nSize is the size of partIds.	<pre>(lret, partIdList) = udpFunctionLibrary.DetachFaces(nSize, facelds)</pre> (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
<pre>long duplicateAlongLine(long partId, UDPVector transVector, int nClones, long* nClones, void* callbackData);</pre>	<pre>(lret, cloneIds) = udmFunctionLibrary.DuplicateAlongLine(partId, transVec, nClones, cloneIdsSize) (callbackData is omitted cloneIdsSize is a new parameter)</pre>
<pre>long duplicateAroundAxis(long partId, UDPCoordinateSystemAxis axis, double angle, int nClones, long* nClones, void* callbackData);</pre>	<pre>(lret, cloneIds) = udmFunctionLibrary.DuplicateAroundAxis (partId, axis, angle, nClones, cloneIdsSize) (callbackData is omitted cloneIdsSize is a new parameter)</pre>

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.

IUDPEExtension 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 UDPEExtension 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 UDPParam
- returns True if udpParams are valid, False otherwise.

CreatePrimitive2(funcLib, udpParams)

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

Optional methods

These methods, which have default implementations, can be implemented as methods of UDPEExtension 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(currentUDPParams, projectedUDPParams)

- currentUDPParams is a list of UDPPParam
- projectedUDPParams is a list of UDPPParam
- returns True on success, False on failure.

MapParametersDefinitionVersions2(oldVersion, oldUDPParams)

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

GetOldPrimitiveParametersDefinition2(version)

- version is a string
- returns a list of UDPPPrimitiveParameterDefinition2 or None on failure.

Example UDP

```
import sys

class UDPEExtension(IUDPEExtension):

    def GetLengthParameterUnits(self):
        return "mm"

    def GetPrimitiveTypeInfo(self)
        typeInfo = UDPPPrimitiveTypeInfo(
            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 UDMPParameters.

GetUnits(instanceld)

- instanceld 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 UDMPParameters 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 udmlnParams 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(instanceld)

- instanceld 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. DII 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. DII 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(instanceId)

- instanceId 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:** faceIds = **GetAllFaces(int: partId)**
10. **List_of_int:** edgeIds = **GetAllEdges(int: partId)**
11. **List_of_int:** vertexIds = **GetAllVertices(int: partId)**
12. **bool:** bret = **SetFaceAttribs(List_of_int: faceIds, List_of_string: attribs)**
13. **bool:** bret = **SetEdgeAttribs(List_of_int: edgeIds, List_of_string: attribs)**
14. **bool:** bret = **SetVertexAttribs(List_of_int: vertexIds, List_of_string: attribs)**
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 = **GetFacelDFromPosition(UDPPosition:** pointOnFace)
6. **int:** ID = **GetEdgeIDFromPosition(UDPPosition:** pointOnEdge)
7. **int:** ID = **CreatePolyline(UDPPolylineDefinition:** polylineDefinition)
8. **int:** ID = **CreateRectangle(CoordinateSystemPlane:** whichPlane, **UDPPosition:** center-Point, **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:** center-Point, **double:** fRadius, **int:** isCovered)
11. **int:** ID = **CreateEllipse(CoordinateSystemPlane:** whichPlane, **UDPPosition:** center-Point, **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:** center-Point, **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_UDPParam**: 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**(**UDPPProgress**: 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,  
        UDPParam(ParamDataType.Int, numCubes),  
        ParamPropType.Number,  
        ParamPropFlag.MustBeInt]
```

- **UDPParam**

- **UDPParamData**

Data field in UDPParam 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 UDPParam.DataType when setting UDPParam.Data.

Example:

```
nCubesParam = UDPParam(ParamDataType.Int, numCubes)
```

```
nCubes = nCubesParam.Data
```

```
distanceParam = UDPParam()
```

```
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, translationVector)
```

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 Name string Purpose

Structure	Construction	Members
	string purpose, string company, string date, string version)	string Company string Date string Version
UDPPrim- itiveParameterDefinition	UDPPrim- itiveParameterDefinition(string name, string description, UnitType unitType, double defaultValue)	string Name string Description UnitType UnitType double DefaultValue
UDPPParam	UDPPParam() UDPPParam(ParamDataType dataType, object data) object can be int, double, string, bool or UDPPosition methods: setInt(int val) setBool(bool val) setString(string val) setDouble(double val) setPosition(UDPPosition val)	ParamDataType DataType object Data object can be int, double , string, bool or UDPPosition
UDPPrim- itiveParameterDefinition2	UDPPrim- itiveParameterDefinition2(string name, string description, UnitType unitType,	string Name string Description UnitType UnitType ParamPropType PropType ParamPropFlag PropFlag

Structure	Construction	Members
	ParamPropType propType, ParamPropFlag propFlag, UDPParam defaultValue)	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)
UDPPolylineDefinition	UDPPolylineDefinition() UDPPolylineDefinition(List_of_UDPPosition positions, List_of_UDPPolylineSegmentDefinition segDefs, int closed,	int IsClosed int IsCovered List_of_UDPPosition ArrayOfPosition List_of_UDPPolylineSegmentDefinition ArrayOfSegmentDefinition

Structure	Construction	Members
	int covered)	
UDPEqua- tionBasedCurveDefinition	UDPEqua- tionBasedCurveDefinition(string functionXt, string functionYt, string functionZt, double tStart, double tEnd, int numOfPointsOnCurve)	string FunctionXt string FunctionYt string FunctionZt double TStart double TEnd int NumOfPointsOnCurve
UDPEqua- tionBasedSurfaceDefinition	UDPEqua- tionBasedSurfaceDefinition(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 = UDPEqua- tionBasedSur- faceDefinition ("u", "v", "1", 0, 1, 0, 1, 0, 0)
UDPHelixDefinition	UDPHelixDefinition(int profileID, UDPPosition ptOnAxis, UDPPosition axisDir, double noOfTurns, bool isRightHanded, double radi- usChangePerTurn,	int ProfileID UDPPosition PtOnAxis UDPPosition AxisDir double NoOfTurns bool IsRightHanded double RadiusChangePerTurn double Pitch

Structure	Construction	Members
	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 suppressFillet, BLNDFilletRadiusLaw filletRadiusLaw, double filletStartRadius, double filletEndRadius, bool followSmoothEdgeSequence, BLNDFilletType filletType, double setbackDistance,	bool SuppressFillet /* Reserved for future */ BLNDFilletRadiusLaw FilletRadiusLaw double FilletStartRadius double FilletEndRadius bool FollowSmoothEdgeSequence /* Reserved for future */ BLNDFilletType FilletType double SetbackDistance

Structure	Construction	Members
	double bulgeFactor)	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, UDParam value, ParamPropType propType, ParamPropFlag propFlag)	string DefName UDPParam DefValue ParamPropType PropType ParamPropFlag PropFlag
UDMOption	UDMOption() UDMOption(string name, UDParam value,	string OptName UDPParam OptValue ParamPropType PropType ParamPropFlag PropFlag

Structure	Construction	Members
	ParamPropType propType, ParamPropFlag propFlag)	
UDMPParameter	UDMPParameter() UDMPParameter(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 Double String Bool Position Unknown
ParamPropType	Text Menu Number

enum Constant	Parameters
	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 SplitKeepPositiveOnly SplitKeepNegativeOnly
PolylineSegmentType	LineSegment ArcSegment SplineSegment AngularArcSegment
MessageSeverity	WarningMessage ErrorMessage

enum Constant	Parameters
	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 suc-
        cessful 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 defin-
    ition 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

An Ansys Electronics Desktop 2024R1 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.

CPython Script Engine and ansysedt process are communicated using GRPC, the ansysedt process started as GRPC server, the Script Script Engine acted as the GRPC client.

Important:

(Supported on both Windows and Linux in 24R1)

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.

Connect() to an opened project in remote machine needs to use a share network path.

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.

Start ansyedt as GRPC server

But default ansyedt process will be started as GRPC server. When there is no argument provided ansyedt.exe it start listening on the first available port from 50051.

-grpcsrv Flag

ansyedt -grpcsrv <optional port number or port range>.

ansyedt -grpcsrv

If no Port Number specified, same as the default like no grpcsrv flag

ansyedt -grpcsrv portNumber

ansyedt process will be listening on the port number, but report error as the port is used by other application.

ansyedt -grpcsrv FirstPortNumber:LastPortNumber

ansyedt process will be listening on the first available port within the port range, but report error if all ports in the range used.

ansyedt -grpcsrv FirstPortNumber:NumberOfPorts

ansyedt process will be listening on the first available port within the port range, but report error if all ports in the range used.

Note: If the last number is smaller than the first number the last number will be treated as the number of ports. (50051: 50250 has the same range as 50051:200)

Connect Functions:

1. Connect(projectPath)
 - a. Connect to the opened project.
 - b. Launch ansyedt.exe, open the project, then connect to it.
 - c. Error if the project does not exist.
2. Connect(portNumber) Connect to running ansyedt process in the local machine. Error if no ansyedt process listening on this port.
3. Connect(machine, projectPath) connect to an open project in remote Machine. projectPath must be a network path. Error if no ansyedt process opened the project.

4. `Connect(machine, portNumber)` connect to a running ansyedt process on the remote machine. Error if no ansyedt process listening on this port.

Example:

```
import sys

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

import ScriptEnv

ScriptEnv.Connect(r"c:\myProjects\Project1.aedt")

oProject = oDesktop.GetActiveProject()

print(oProject.GetName())
```

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:

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

On Linux:

```
ansyedt -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](#)

This page intentionally
left blank.

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](#)

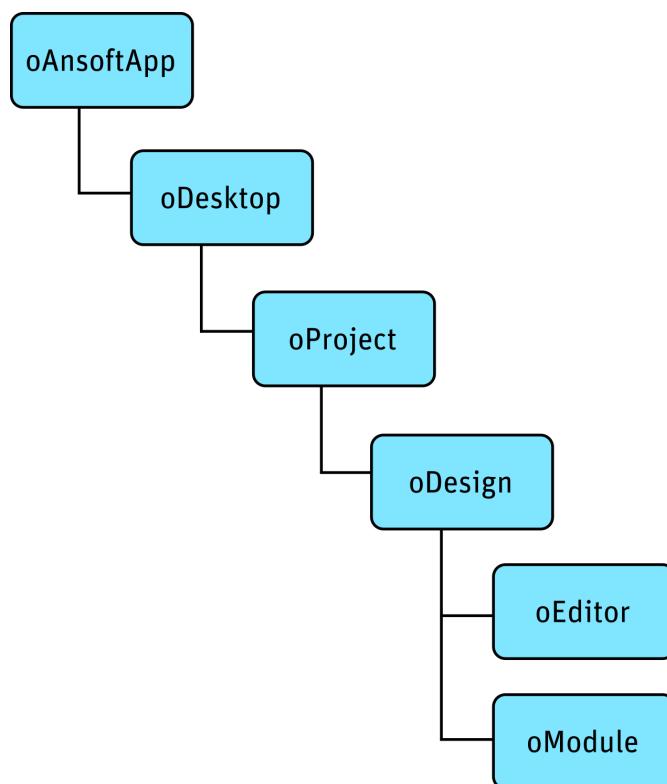
[Working with Project Scripts](#)

[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()
```

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

- [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](#)

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](#)
- [Layout 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()
```

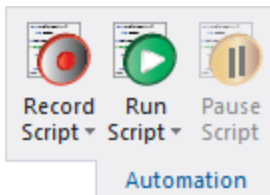
Running a Script

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

Within Electronics Desktop

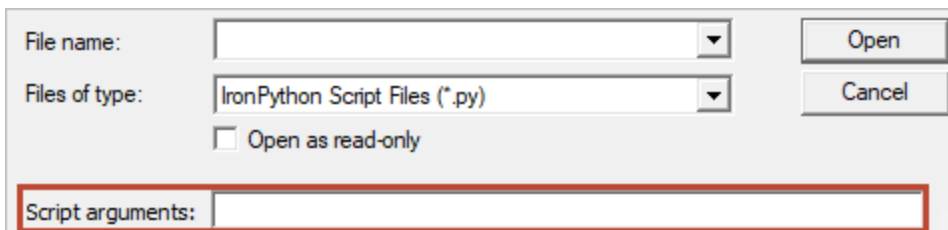
To run scripts in Electronics Desktop:

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:

```
ansyedt.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 HFSS 3D Layout help topic "Running Ansys Electronics Desktop 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 collection
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](#).

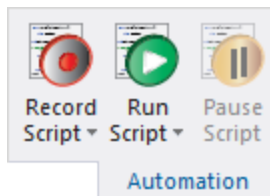
Important:

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:

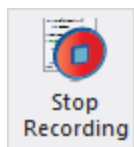
1. Click **Tools > Record Script to File**, or select the **Automation** tab and click the **Record Script** icon:



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.

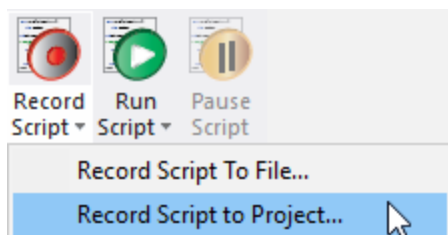
7. When you have finished recording the script, click **Stop Recording**, or select **Tools > Stop Script Recording**.

The recorded script is saved to the folder you specified.

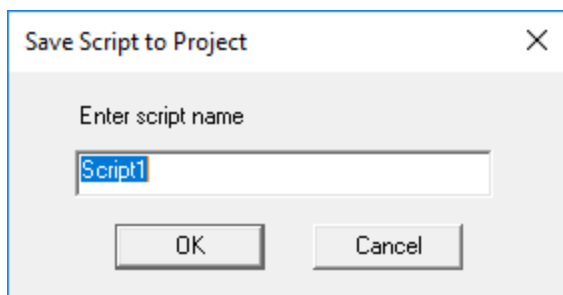
Recording a Script to a Project

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:



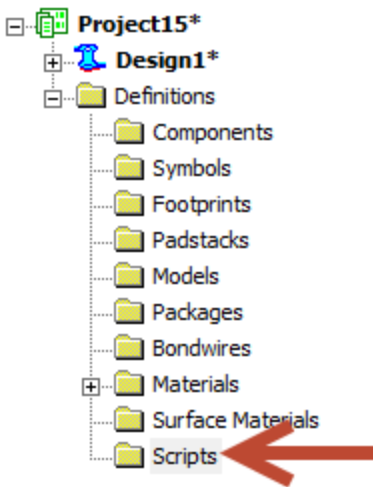
2. Enter a name for the script in the text box, then click **OK**.
3. Perform the steps you want to record.
4. When you have finished, click **Stop Recording**, or select **Tools > Stop Script Recording**.

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](#)
- [Layout Scripts and the Active Layer](#)
- [Scripts and Locked Layers](#)
- [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:=", "50Ohm",  
    "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)
```

Layout Scripts and the Active Layer

A design's active layer is the layer that is used for object creation and placement during adding operations in the user interface. Adding operations include paste and placement of instances, as well as object creation. Usually there is an active layer, but it is not required and cannot be assumed. Adding operations are responsible for ensuring that the active layer exists and meets any particular requirements (such as layer type) for the operation. Adding operations may change the active layer to a different layer that meets requirements. If the active layer is changed, Electronics Desktop generates an alert. If no layer is available to be active, the operation is not done.

The active layer is not used during script adding operations. Script adding operations are responsible for ensuring that the specified layer exists and meets the particular requirements (such as layer type) for the operation. If there is a problem with using the specified layer, the operation is not done. The active layer is always visible and selectable. These attributes are reset, if needed, when a layer is made active. The current active layer is indicated by a combo box display in the toolbar. The list for the combo box contains all layers that may be set active.

The active text style is related to the active layer. If there is no active layer, there is no active text style. Objects on the active layer have priority during snapping.

Scripts and Locked Layers

The locked attribute of a layer is defined to mean that you may not edit, delete, or add objects on the layer, either directly or with scripts (i.e., scripts run on layout or footprint definitions). This includes not being able to change properties of objects on the layer. Note, however, that parameter changes can alter objects on locked layers.

The locked attribute of a layer is configurable using script commands and is user-editable via the **Edit Layers Dialog** in the Layout Editor.

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 Compliance 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	Compliance	Compliance.GetParentDesign() — Returns a <code>oDesign</code> item that can be used to call Design functions. Compliance.GetPropservername() — Returns a Compliance identification name that can be used in <code>oEditor</code> property-method scripts, such as <code>GetPropertyvalue()</code>, <code>SetPropertyvalue()</code>, etc. Compliance.GetComponentName() — Returns the component name, e.g. "MS_TRL".
Simulate Component	Compliance	Compliance.GetDesign() — Returns the interface to the specified design simulation. Compliance.GetProgress() — Returns the completion percentage (from 0 to 100) of the specified design simulation. Compliance.GetRunStatus() — Returns the status number of the specified design simulation. Compliance.Abort() — 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, **ComplInstance** is visible from the invoked sub-script, ScriptName, and ComplInstance's functions can be called from ScriptName.

PyAEDT (Beta)

PyAEDT is a Python library that interacts with the AEDT API to make scripting simpler for the end user. It supports all AEDT 3D products (HFSS, Icepak, Maxwell 3D, and Q3D Extractor), 2D tools, Ansys Mechanical, EMIT, Circuit tools like Nexxim, system simulation tools like Twin Builder, and layout tools like HFSS 3D Layout and EDB. Additionally, it enables the end user to have a CPython interface with AEDT. Its class and method structures simplify operation for the end user, enabling more Pythonic code while reusing information as much as possible across the various APIs.

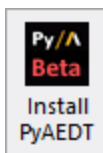
Documentation for PyAEDT can be found online at: <https://aedt.docs.pyansys.com/version/stable/>

Installing PyAEDT adds three items to the **Tools > Toolkit > PersonalLib** menu and to the **Automation** tab:

- **Console** – launches the PyAEDT console.
- **Jupyter Notebook** – launches Jupyter Notebook (a computational notebook) in an internet browser.
- **Run PyAEDT Script** – launches a file browser allowing you to select a Python script to run via PyAEDT.

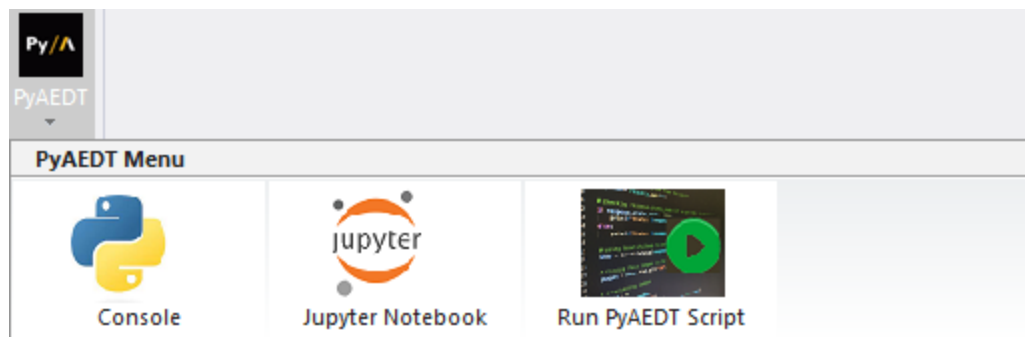
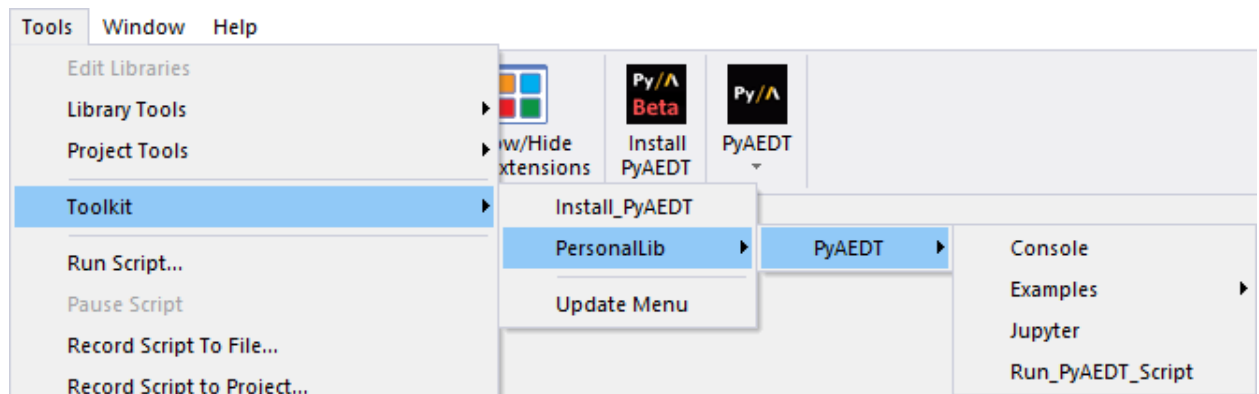
To install PyAEDT:

- From the **Automation** tab, click **Install PyAEDT**.



A web browser launches, and takes you to detailed installation instructions.

When installation is complete, the **Tools > Toolkit > PersonalLib** menu and the **Automation** tab update to display PyAEDT menu options:



2 - Object-Oriented Property Scripting

Object-oriented scripting enhances scripting in AEDT by allowing object-oriented access to retrieve or modify object properties. The primary gain is the ease with which properties of various existing objects in an AEDT project or design can be read, modified, and set. This feature also allows for much less code to be written to access object properties and enables much more readable code for users, avoiding complex array input.

This topic covers the following:

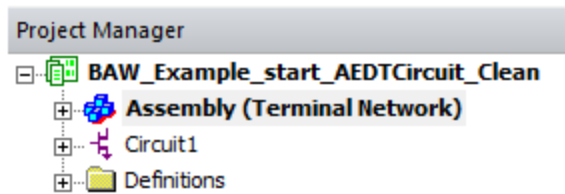
- Logic/syntax
- Basic attributes of project and design properties
- Example Scripts

Object-Oriented Scripting

There are five basic functions in object-oriented scripting for retrieving and setting 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()
```

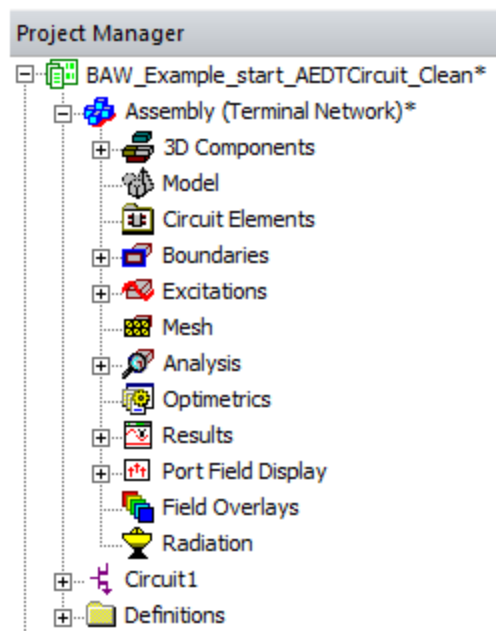
Once the objects have been defined, you can use GetChildNames() to learn what object instances exist for each. As an example, observe the Child Names of oProject, and you see a list of the Designs in the AEDT Project, per the GUI.

```
>>> 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']
```

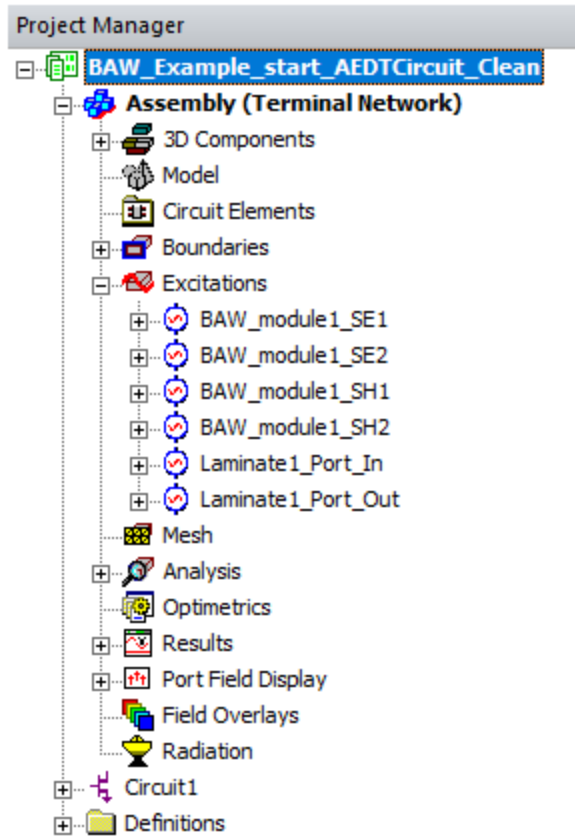
These names are what you would expect based on the Project Manager Layout:



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')
```

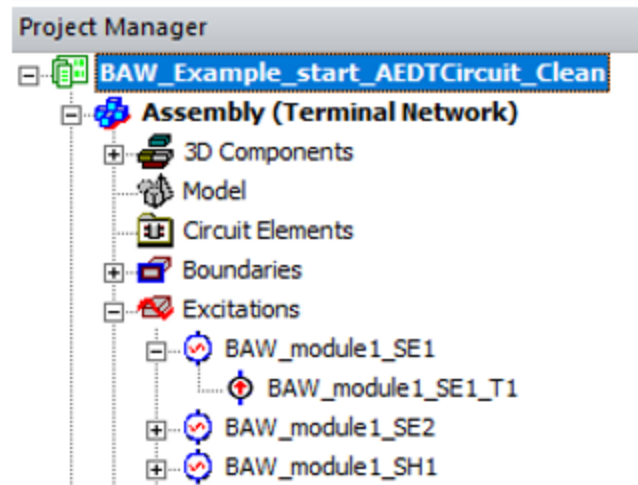
You now have an object, oExcitations defined to be the oDesign Child Object 'Excitations'. What does this mean? If you expand the Excitations dialogue in the Project Manager, you expect that the Child Names of this object would be the names of the Excitations as defined, in this case six ports:



```
>>> oExcitations.GetChildNames()  
['Laminat1_Port_In', 'Laminat1_Port_Out', 'BAW_module1_SE1', 'BAW_module1_SE2', 'BAW_module1_SH1',  
'BAW module1 SH2']
```

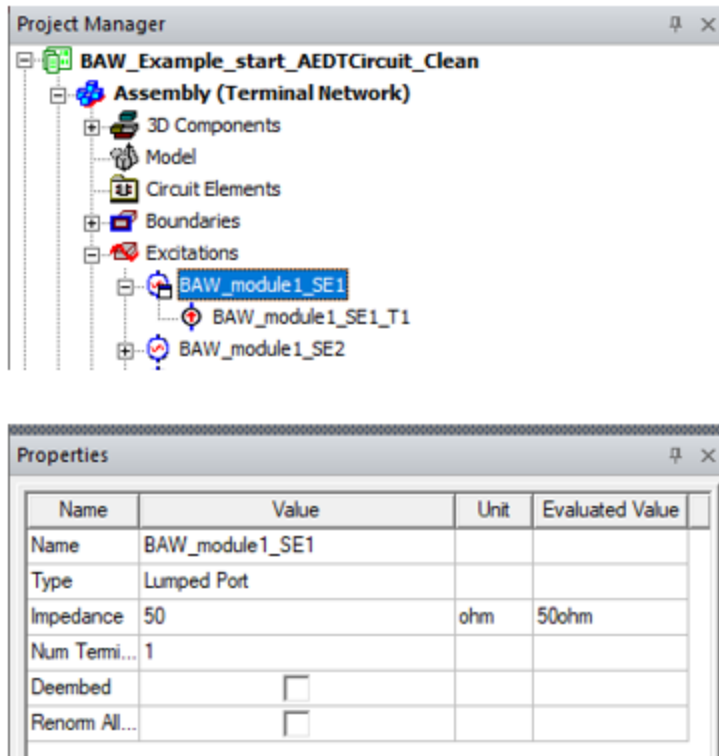
There is clear logic to the Object Child Names as the children of the oExcitations object as the ports that have been defined in HFSS. Looking at the Project tree can help you to conceive and retrieve desired information.

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']
```




```
>>> oPort.GetPropNames()  
['Name', 'Type', 'Impedance', 'Num Terminals', 'Deembed', 'Renorm All Terminals']
```

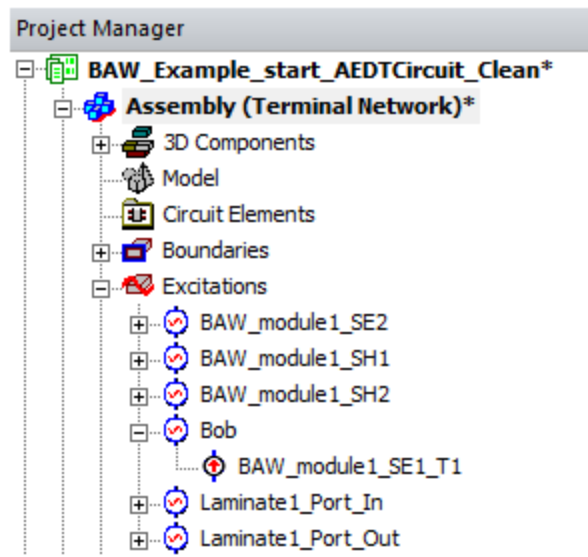
Once you identify the desired object and you know the desired property, you can access the value via `GetPropValue()`. For example, if you want to retrieve the name of the object `oPort`:

```
>>> 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', 'Magnetostriiction Type', 'Inverse Magnetostriiction 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, 0.0040000000000000001], [0.002, 0.0050000000000000001], [0.0030000000000000001,
0.0060000000000000001]]
>>>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  ..... "Domain:=",.Sweep"
12  ..... ],
13  [
14  ..... "Freq:=",.["All"]
15  ..... ],
16  [
17  ..... "dB(St(Terminal_1,Terminal_1))"
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 of 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', 'Optimetrics',
'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.

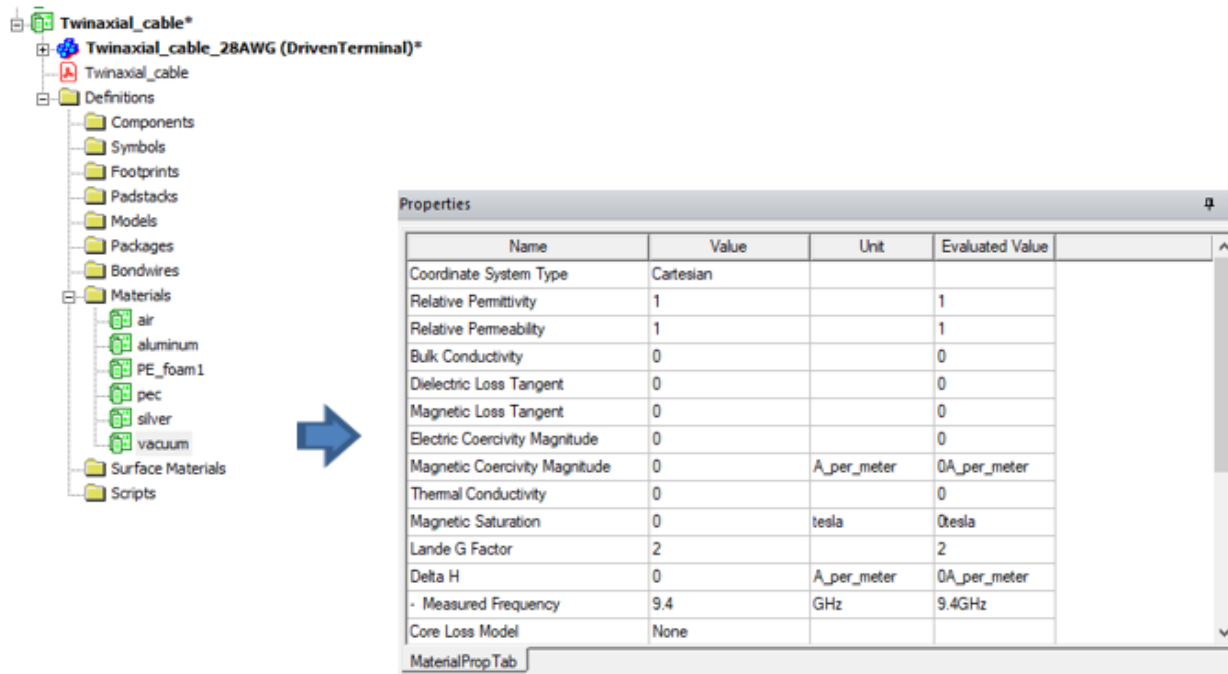
```
>>> setup.GetChildNames()  
  
['Sweep', 'Sweep1', 'Sweep2']  
  
>>> setup.GetChildTypes()  
  
['Discrete', 'Interpolating']  
  
>>> sweep1 = setup.GetChildObject("sweep")
```

Related Topics

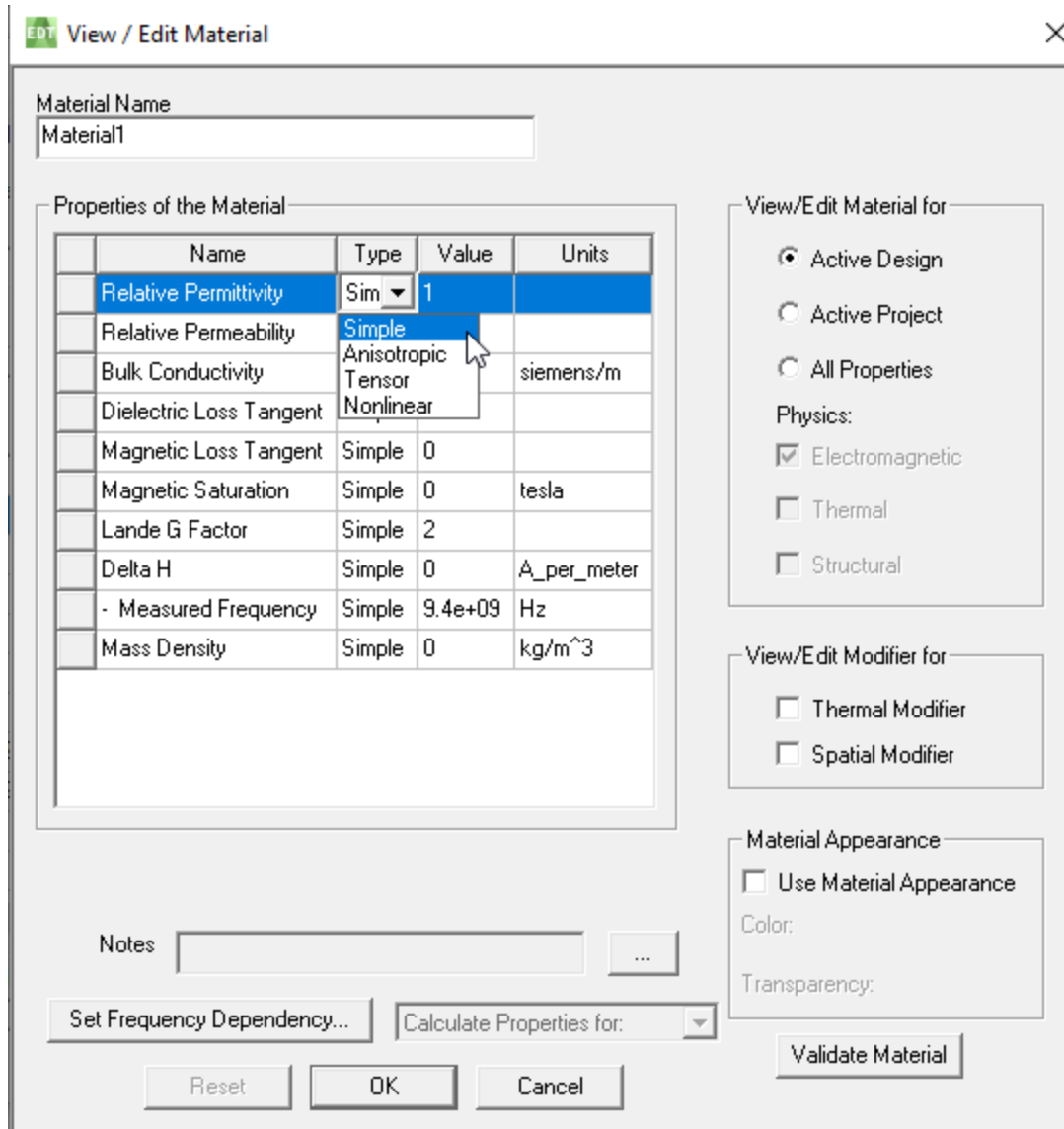
[Example: GetPropNames and GetPropValues for Layered Impedance Boundary Script](#)

Materials Scripting Support

Supported material properties are shown in a Property window for the material item.



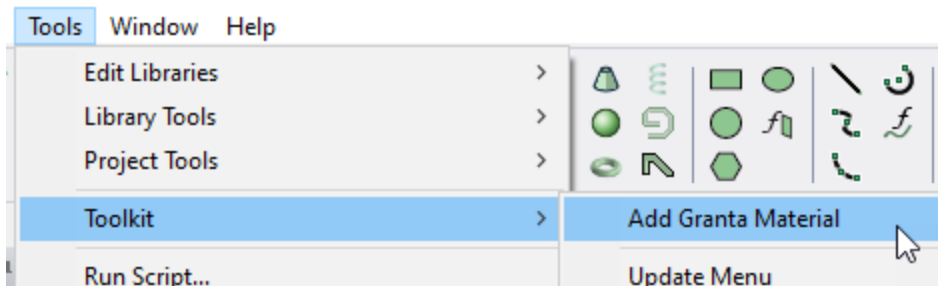
Some Material Properties like Relatively Permittivity may have values assigned as BH Curves or Tensors, as discussed in the Assigning Materials chapter of the online help.

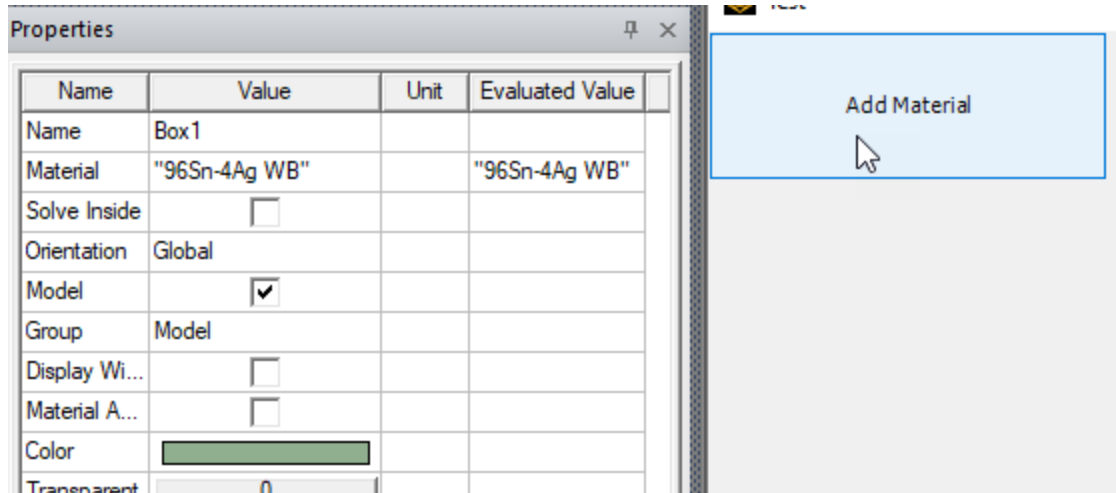


With this feature enabled you can then:

- 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](#)
- [AddDefiniitonFromLibFile](#)
- [GetExtendedDefinitionObect](#)

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 Per-  
meability/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',  
'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 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 Per-  
meability/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 Per-
meability/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 Per-
meability/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', 'Mag-  
netic Coercivity Components', 'Magnetic Coercivity Components/Component1', 'Magnetic Coer-  
civity Components/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', 'Pois-  
son's Ratio Type/Choices', 'Poisson's Ratio', 'Poisson's Ratio/SIValue', 'Poisson's Ratio/E-  
valuatedValue']
```

```
>>>
```

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:

[Object Script Property Function Summary](#)

[Conventions Used in this Chapter](#)

[ChangeProperty](#)

[GetArrayVariables](#)

[GetProperties](#)

[GetPropertyValue](#)

[GetVariables](#)

[GetVariableValue](#)

[SetPropertyValue](#)

[SetVariableValue](#)

[Example Use of Record Script and Edit Properties](#)

[Additional Property Scripting Example](#)

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 "/" . In the HFSS-Optimtee example, the Optimization Max value of variable "offset" is represented by the path "TeeModel/offset/Optimization/max" or "TeeModel/Variable[offset]/Optimization[max]"
- 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(<*blIncludeReadOnly*>);
 - <*blIncludeReadOnly*> – 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(bIncludeReadOnly) 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)
GetChildNames() 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("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", "NotAnOptimizerName");` 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 depend 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 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:

Property	A single item that can be modified in the Properties window or in the modal Properties pop-up window.
<PropServer>	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.
<PropTab>	Corresponds to one tab in the Properties window, the one under which properties are being edited.
<PropName>	The name of a single property.

The following tables list specific <PropServer> and <PropTab> values for different property types.

For Project Variables:

<PropServer>	"ProjectVariables"
<PropTab>	"ProjectVariableTab"

For Local Variables:

<PropServer>	"LocalVariables"
<PropTab>	"LocalVariableTab"

For Passed Parameters:

<PropServer>	"Instance:<Name of Circuit Instance>"
<PropTab>	"PassedParameter Tab"

For Definition Parameters:

<PropServer>	"DefinitionParameters"
<PropTab>	"DefinitionParameters"

For Modules and Editors:

<PropServer>	<ModuleName>:<ItemName> where <ItemName> is the boundary name, solution setup name, etc. For example, "BoundarySetup:PerfE1"
<PropTab>	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:

<PropServer>	Name of the object. For example, "Box1".
<PropTab>	"Geometry3DAttributeTab"

For 3D Model Editor operations:

<PropServer>	<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.
<PropTab>	"Geometry3DCmdTab"

For Reporter operations on Report properties:

<PropServer>	<ReportSetup>
<ChangeProperty>	Array. For example, to set the company name in a plot header to "My Company": <pre>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"))))</pre>

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.

Callback Scripting Using PropHost 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 PropHost. The PropHost represents the PropServer (owner of properties) that contains the Property that is calling the Callback script. Therefore, the Callback script can use the PropHost's functions to query or set other properties in the same PropServer.

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

<propType>	<propTypeName>	Property Description
0	TextProp	Text
1	MenuProp	Menu
2	CheckboxProp	Checkbox
3	VariableProp	Variable
4	VPointProp	VPoint
5	V3DPointProp	V3DPoint
6	NumberProp	Number
7	ColorProp	Color
8	PointProp	Point
9	ValueProp	Value
10	ButtonProp	Button
11	SeparatorProp	Separator
12	NetlistProp	Netlist
13	FileNameProp	FileName

<tabType>	Description	Objects with this tab type
0	DefaultTab	
1	PassedParameterTab	Instances of designs, components, geometric objects
2	DefinitionParameterTab	Definitions of designs, components
3	LocalVariableTab	Definitions of designs, components
4	ProjectVariableTab	Projects
5	ConstantsTab	Projects
6	BaseElementTab	Geometric objects

7	ComponentTab	Designs, components
8	PropertyTab	
9	CircuitTab	
10	SystemTab	
11	HFSS3DLayoutTab	
12	HfssTab	HFSS objects
13	OptimetricsTab	Optimetrics data
14	AltraSimTab	
15	Report3DTab	Report3d
16	FieldsPostProcessorTab	Fieldspostprocessor
17	MeshSetupTab	Manual meshing setup
18	RadFieldSetupTab	Radiation field geometry setup
19	Geometry3DAttributeTab	Geometry3D
20	Geometry3DCmdTab	Geometry3D
21	Geometry3DPolylineTab	Geometry3D
22	Geometry3DCSTab	Geometry3D
23	Geometry3DPlaneTab	Geometry3D
24	Geometry3DPointTab	Geometry3D
25	Geometry3DListTab	Geometry3D
26	StandardPropTab	
27	PropDisplayPropTab	
28	CustomPropTab	
39512	Component	Designs, components

The topics for this section include:

[ChangeProperty for HFSS and Maxwell](#)

[ChangeProperty for Schematic and Layout](#)

ChangeProperty (Schematic Editor and Layout 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 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>,  
      <NewPropArray>,  
      <ChangedPropsArray>,  
      <DeletedPropsArray>)
```

<PropServersArray>

```
Array("Name:PropServers", <PropServer>,
```



```
<PropServer>, ...)
```

```
<NewPropArray>
```

```
Array("Name:NewProp", <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 **RMxpert>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"))))
```

PropHost Functions

The following commands can be used to manipulate properties from a Property script.

[GetEditor](#)

[GetPropHost](#)

AddMenuProp

Use: Creates a new Menu property in tabType with name specified; choices are set to the values in choices; initial selection is initialChoice.

UI Access	NA		
Parameters	Name	Type	Description
	<tabType>	Integer	One of the following, where tab titles are shown in parentheses: 1 for PassedParameterTab ("Parameter Values") 2 for DefinitionParameterTab (Parameter Defaults") 3 for LocalVariableTab ("Variables" or "Local Variables") 4 for ProjectVariableTab ("Project variables") 5 for ConstantsTab ("Constants") 6 for BaseElementTab ("Symbol" or "Footprint") 7 for ComponentTab ("General") 28 for CustomTab ("Intrinsic Variables") 39500 for Quantities ("Quantities") 39506 for Signals ("Signals") 39512 for Component ("Component")
	<propName>	String	Name of the new menu property
	<choices>	String	A comma-separated list of menu choice strings.
	<initialChoice>	String	One of the strings in the list of choices.
Return Value	None		

Python Syntax	AddMenuProp(<tabType>, <propName>, <choices>, <initialChoice>)
Python Example	<pre>PropHost.AddProp(2, "ResChoices", "inline,upfront,parallel,series", "parallel")</pre>

VB Syntax	AddMenuProp(<tabType>, <propName>, <choices>, <initialChoice>)
VB Example	<pre>PropHost.AddProp(2, "ResChoices", "inline,upfront,parallel,series", "parallel")</pre>

AddMenuProp2

Creates a new Menu property in tabTypeName with name specified; choices are set to the values in choices; initial selection is initialChoice.

UI Access	None		
Parameters	Name	Type	Description
	<tabType>	String	One of the following, where tab titles are shown in parentheses: PassedParameterTab ("Parameter Values") DefinitionParameterTab (Parameter Defaults") LocalVariableTab ("Variables" or "Local Variables") ProjectVariableTab ("Project variables") ConstantsTab ("Constants") BaseElementTab ("Symbol" or "Footprint") ComponentTab ("General") Component("Component")

			CustomTab ("Intrinsic Variables")
			Quantities ("Quantities")
			Signals ("Signals")
	<propName>	String	Name of the new menu property.
	<choices>	String	A comma-separated list of menu choice strings.
	<initialChoice>	String	One of the strings in the list of choices.
Return Value	None		

Python Syntax	AddMenuProp2(<tabTypeName>, <propName>, <choices>, <initialChoice>)		
Python Example	<pre>PropHost.AddMenuProp2("DefinitionParameterTab", "ResChoices", "inline,upfront,parallel,series", "parallel")</pre>		

VB Syntax	AddMenuProp2(<tabTypeName>, <propName>, <choices>, <initialChoice>)		
VB Example	<pre>PropHost.AddMenuProp2("DefinitionParameterTab", "ResChoices", "inline,upfront,parallel,series", "parallel")</pre>		

AddProp

Use: Creates a new propType property in tabType with name and value specified.

Command: None

Syntax: AddProp(<tabType>, <propType>, <propName>, <valueText>)

Return Value: None.

VB Example: PropHost.AddProp(2, 3, "W1", "10mm"); creates a new VariableProp in the DefinitionParameters tab named W1 with value 10mm.

UI Access	NA		
Parameters			
	Name	Type	Description
	<tabType>	Integer	One of the following, where tab titles are shown in parentheses: 1 for PassedParameterTab ("Parameter Values") 2 for DefinitionParameterTab (Parameter Defaults") 3 for LocalVariableTab ("Variables" or "Local Variables") 4 for ProjectVariableTab ("Project variables") 5 for ConstantsTab ("Constants") 6 for BaseElementTab ("Symbol" or "Footprint") 7 for ComponentTab ("General") 28 for CustomTab ("Intrinsic Variables") 39500 for Quantities ("Quantities") 39506 for Signals ("Signals") 39512 for Component ("Component")
	<propType>	Integer	One of the following to add new definition properties (e.g., to component, design or project): 0 for TextProp (any string) 1 for Menu (one,two,three) 2 for CheckboxProp ("true" or "false" values)

			3 for VariableProp (useful for Project and Local Variables, expression) 4 for VPointProp (x and y values, which can be expressions) 5 for V3DPointProp (x, y and z values, which can be expressions) 6 for NumberProp (simple number, no expression) 7 for ColorProp (takes an RGB value) 8 for PointProp (x and y values, no expressions) 11 for SeparatorProp (used for property list organization, the value is the separator "title") 13 for FileNameProp (special button property for filenames) 39506 for GenericProp (for generics in Twin Builder, SML and VHDL) 39500 for QuantityProp (for Twin Builder quantities, SML and VHDL) 39504 for SignalProp (for VHDL signals in Twin Builder) 39502 for VariablePropNU (like a VariableProp, but provides access to setting NetlistUnits in Component definition dialog for Designer; for Twin Builder, use GenericProp in DefinitionParametersTab, QuantityProp in Quantities, and signalProp in Signals)
	<propName>	String	Name of the new menu property.
	<valueText>	String	The property value
Return Value	None		

Python Syntax	AddProp(<tabType>, <propType>, <propName>, <valueText>)
Python Example	PropHost.AddProp(2, 3, "W1", "10mm")

VB Syntax	AddProp(<tabType>, <propType>, <propName>, <valueText>)
VB Example	PropHost.AddProp(2, 3, "W1", "10mm")

Note:

AddProp adds a new property using the <propName> preceded by a separator "--". For instance, a new property added with the <propName> "xxx" will be added with the name "--xxx". Consequently, all subsequent script functions that wish to access the added property must now use the name "--xxx".

For example:

```
PropHost.AddProp 1, 11, "Time_Domain_Options", ""
```

```
Prophost.SetHidden "--Time_Domain_Options", 1
```

propType	Property Type	Value Example
0	Text	"hello"
1	Menu	"one,two,three"
2	Checkbox	"true" or "false"
3	Variable	"34mm" or "2*val1"
4	VPoint	"x1,y1"
5	V3DPoint	"x1,y1,z1"
6	Number	"17" or "22pF"
7	Color	

8	Point	"34,27"
9	Value	"34mm" or "2*val1"
11	Separator	"colors"
13	Filename	"c:\temp\ne22.s2p"

AddProp2

Creates a new propTypeName property in tabTypeName with name and value specified.

UI Access	NA		
Parameters	Name	Type	Description
	<tabType>	Type	One of the following, where tab titles are shown in parentheses: PassedParameterTab ("Parameter Values") DefinitionParameterTab (Parameter Defaults) LocalVariableTab ("Variables" or "Local Variables") ProjectVariableTab ("Project variables") ConstantsTab ("Constants") BaseElementTab ("Symbol" or "Footprint") ComponentTab ("General") Component("Component") CustomTab ("Intrinsic Variables") Quantities ("Quantities") Signals ("Signals")
	<propType>	String	One of the following to add new definition properties (e.g., to component,

			design or project): TextProp (any string) CheckboxProp ("true" or "false" values) VariableProp (useful for Project and Local Variables, expression) VPointProp (x and y values, which can be expressions) V3DPointProp (x, y and z values, which can be expressions) NumberProp (simple number, no expression) ColorProp (takes an RGB value) PointProp (x and y values, no expressions) SeparatorProp (used for property list organization, the value is the separator "title") FileNameProp (special button property for filenames) GenericProp (for generics in Twin Builder, SML and VHDL) QuantityProp (for Twin Builder quantities, SML and VHDL) SignalProp (for VHDL signals in Twin Builder) VariablePropNU (like a <code>VariableProp</code>, but provides access to setting <code>Net-listUnits</code> in Component definition dialog for Designer; for Twin Builder, use <code>GenericProp</code> in <code>DefinitionParametersTab</code>, <code>QuantityProp</code> in <code>Quantities</code>, and <code>signalProp</code> in <code>Signals</code>)
	<propName>	String	Name of the new menu property.
	<valueText>	String	The property value
Return Value	None		

Python Syntax	AddProp2(<tabTypeName>, <propTypeName>, <propName>, <valueText>)
Python Example	<pre>PropHost.AddProp2("DefinitionParameterTab", "VariableProp", "W1", "10mm")</pre>

VB Syntax	AddProp2(<tabTypeName>, <propTypeName>, <propName>, <valueText>)
VB Example	<pre>PropHost.AddProp2("DefinitionParameterTab", "VariableProp", "W1", "10mm")</pre>

ExecuteScript

Finds the named script in the Definitions/Scripts folder and runs that script; the script being run can also use the PropHost object.

UI Access	NA		
Parameters	Name	Type	Description
	<scriptName>	String	Name of a script stored in a script library
Return Value	None		

Python Syntax	ExecuteScript(<scriptName>)
Python Example	<pre>PropHost.ExecuteScript("PropChangeScript")</pre>

VB Syntax	ExecuteScript(<scriptName>)
VB Example	PropHost.ExecuteScript("PropChangeScript")

GetApplication

Returns the application object currently running the script.

UI Access	NA		
Parameters	Name	Type	Description
	None		
Return Value	Object The Application		

Python Syntax	GetApplication()
Python Example	oAnsoftApp2 = prophost.GetApplication()

VB Syntax	GetApplication
VB Example	Dim oAnsoftApp2 oAnsoftApp2 = prophost.GetApplication

GetCallback

Finds named property and returns name of Callback script.

UI Access	NA		
Parameters	Name	Type	Description
	<propName>	String	Name of the property
Return Value	String		

Python Syntax	GetCallback(<propName>)		
Python Example	<pre>a = PropHost.GetCallback("W1")</pre>		

VB Syntax	GetCallback(<propName>)		
VB Example	<pre>a = PropHost.GetCallback("W1")</pre>		

GetChangedProperty

Use: If the script was called by a Callback associated with a property, this function returns the name of that property.

UI Access	NA		
Parameters	Name	Type	Description
	None		
Return Value	String		

	Returns "C" if the script was a Callback associated with the property named "C" and the script was called in response to the property "C" changing value.

Python Syntax	GetChangedProperty()
Python Example	<pre>pn = PropHost.GetChangedProperty()</pre>

VB Syntax	GetChangedProperty()
VB Example	<pre>pn = PropHost.GetChangedProperty()</pre>

GetDescription

Finds named property and returns description string

UI Access	NA		
Parameters	Name	Type	Description
	<propName>	String	Name of the property
Return Value	String		

Python Syntax	GetDescription(<propName>)

Python Example	<pre>a = PropHost.GetDescription("W1")</pre>

VB Syntax	GetDescription(<propName>)
VB Example	<pre>a = PropHost.GetDescription("W1")</pre>

GetDesign

Use: Returns the interface to the specified design simulation.

Command: None

Syntax: GetDesign <DesignName>

Return Value: Interface to the specified design simulation.

Parameters: <DesignName>

Type: <string>

VB Example: Set oDesign = oPropHost.GetDesign ("DesignerModel1")

GetEditor

Use: Returns an interface to the editor requested IF the PropServer behind the PropHost is contained within that type of editor.

Command: None

Syntax: GetEditor(<editorName>)

Return Value: String

VB Example: Set oLayout2 = PropHost.GetEditor("Layout"); returns the interface to the layout containing a selected component. This interface can be used to call Layout Scripting functions.

Python Syntax	GetEditor(<editorName>)
Python Example	<code>oEditor = GetEditor("SchematicEditor")</code>

GetEvaluatedText

Returns the evaluated value of the property, useful when the value is an expression.

UI Access	NA		
Parameters	Name	Type	Description
	<propname>	String	The name of the property
Return Value	String The Evaluated Value		

Python Syntax	GetEvaluatedText (<propname>)
Python Example	<code>ret = prophost.GetEvaluatedText ("R")</code>

VB Syntax	GetEvaluatedText (<propname>)
VB Example	<code>ret = prophost.GetEvaluatedText ("R")</code>

GetFileName

Finds the full path name to propName.

UI Access	NA		
Parameters	Name	Type	Description
	<propName>	String	Name of the property
Return Value	Full path file name or empty string.		

Note:

Directory variables can be used in the property's value (e.g. \$projectdir, \$userlib, \$syslib, \$personallib) and these will be expanded to the correct path. GetFileName always returns a path string; if propName actually contains a variable expression, that expression is evaluated to a constant string before returning.

Python Syntax	GetFileName(<propName>)
Python Example	<code>a = PropHost.GetFileName("SubstrateFile")</code>

VB Syntax	GetFileName(<propName>)
VB Example	<code>a = PropHost.GetFileName("SubstrateFile")</code>

GetHidden

Finds named property and returns its Hidden flag.

UI Access	NA		
Parameters	Name	Type	Description
	<propName>	String	Name of the property
Return Value	Returns 1 if property is hidden and 0 if it is not.		

Python Syntax	GetHidden(<propName>)
Python Example	<pre>a= PropHost.GetHidden("W1")</pre>

VB Syntax	GetHidden(<propName>)
VB Example	<pre>a = PropHost.GetHidden("W1")</pre>

GetProgress

Use: Returns the percentage (from 0 to 100) of the simulation completed.

Command: None

Syntax: GetProgress(<simProgress>)

Return Value: String

VB Example:

```
a = PropHost.GetDesign(<simProgress> );
```

GetPropHost

Use: Returns an interface to the PropHost of the CompInstance, which gives access to its properties.

Command: None

Syntax: GetPropHost()

VB Example: `Set oPropHost2 = CompInstance.GetPropHost()`;

Returns the interface to the properties of the compInstance.

This interface can be used to call PropHost functions; for more information see [Callback Scripting Using PropHost Object](#).

Python Syntax	GetPropHost()
Python Example	<code>oPropHost2 = CompInstance.GetPropHost()</code>

GetPropServers

Returns array of objects that have properties showing on tabTypeName.

UI Access	NA		
Parameters	Name	Type	Description
	None		
Return Value	Array of propserver names.		

Python Syntax	GetPropServers(<tabTypeName>)
Python Example	<pre>objects = PropHost.GetPropServers("PassedParameterTab")</pre> Returns array containing PropServers that have properties shown on PassedParameterTab; this would include only components and designs; individual properties can be accessed using standard notation, e.g. objects(0) might contain "CompInst@CAP_;1".

VB Syntax	GetPropServers(<tabTypeName>)
VB Example	<pre>objects = PropHost.GetPropServers("PassedParameterTab")</pre> Returns array containing PropServers that have properties shown on PassedParameterTab; this would include only components and designs; individual properties can be accessed using standard notation, e.g. objects(0) might contain "Complnst@CAP_1".

GetPropTabType

Finds named property and returns the id of the tab it is in.

UI Access	NA		
Parameters	Name	Type	Description
	<propName>	String	Name of the property
Return Value	Returns string		

Python Syntax	GetPropTabType(<propName>)
Python Example	<pre>a = PropHost.GetPropTabType("W1")</pre>

VB Syntax	GetPropTabType(<propName>)
VB Example	<pre>a = PropHost.GetPropTabType("W1")</pre>

GetReadOnly

Finds named property and returns its ReadOnly flag.

UI Access	NA		
Parameters	Name	Type	Description
	<propName>	String	Name of the property
Return Value	Returns 1 if property is read-only and 0 if it is not.		

Python Syntax	GetReadOnly(<propName>)		
Python Example	<pre>a = PropHost.GetReadOnly("W1"); returns 1</pre>		

VB Syntax	GetReadOnly(<propName>)		
VB Example	<pre>a = PropHost.GetReadOnly("W1"); returns 1</pre>		

GetRunStatus

Use: Returns the status number of the specified design simulation.

Command: None

Syntax: GetRunStatus(<statusNumber>)

Return Value: Returns string.

VB Example: `a = PropHost.GetRunStatus(<statusNumber>);`

GetTabTypeName

Finds named property and returns the name of the tab it is on.

UI Access	NA		
Parameters	Name	Type	Description
	<propName>	String	Name of the property
Return Value	String		

Python Syntax	GetTabTypeName(<propName>)		
Python Example	<code>a = PropHost.GetTabTypeName ("W1")</code>		

VB Syntax	GetTabTypeName(<propName>)		
VB Example	<code>a = PropHost.GetTabTypeName ("W1")</code>		

GetText

Finds property in any tab and returns its value as a text string.

UI Access	NA		

Parameters	Name	Type	Description
	<propName>	String	Name of the property
Return Value	String		

Python Syntax	GetText(<propName>)
Python Example	<pre>a = PropHost.GetText("C")</pre>

VB Syntax	GetText(<propName>)
VB Example	<pre>a = PropHost.GetText("C")</pre>

GetValue

Use: Finds property in any tab and returns its value as a double.

Command: None

Syntax: GetValue(<propName>)

Return Value: Returns double.

VB Example:

```
a = PropHost.GetValue("C") ;
```

Note:

Values are returned in SI units. Compound SI units are, in general, not supported. Temperature values are returned in Celsius

IsValueConstant

Determines whether the specified property is a constant or not. Returns True if propName is evaluated to be a constant; returns False if propName is evaluated to be an expression.

UI Access	NA		
Parameters	Name	Type	Description
	<propname>	String	The name of the property.
Return Value	Integer; non-zero for constant, zero for variable or expression.		

Python Syntax	IsValueConstant (<propname>)
Python Example	<pre>ret = prophost.IsValueConstant("R")</pre>

VB Syntax	IsValueConstant (<propname>)

VB Example	<code>ret = prophost.IsValueConstant("R")</code>

PropertyExists

Finds named property and returns its property type.

UI Access	NA		
Parameters	Name	Type	Description
	<propName>	String	Name of the property
Return Value	Returns 1 if property exists in any tab, 0 if it does not.		

Python Syntax	<code>PropertyExists(<propName>)</code>
Python Example	<code>a = PropHost.PropertyExists("W1")</code>

VB Syntax	<code>PropertyExists(<propName>)</code>
VB Example	<code>a = PropHost.PropertyExists("W1")</code>

RemoveProp

Removes the named property from whichever tab it is found.

UI Access	NA		
Parameters	Name	Type	Description
	<propName>	String	Name of the property

Return Value	None

Python Syntax	RemoveProp(<propName>)
Python Example	<code>PropHost.RemoveProp("W1")</code>

VB Syntax	RemoveProp(<propName>)
VB Example	<code>PropHost.RemoveProp("W1")</code>

SetCallback

Finds named property and sets its Callback script.

UI Access	NA		
Parameters	Name	Type	Description
	<propName>	String	Name of the property
	<scriptName>	String	Name of a script stored in a script library.
Return Value	None		

Python Syntax	SetCallback(<propName>, <scriptName>)
Python Example	<code>PropHost.SetCallback("W1", "SynchronizeResistors")</code>

VB Syntax	<code>SetCallback(<propName>, <scriptName>)</code>
VB Example	<code>PropHost.SetCallback("W1", "SynchronizeResistors")</code>

SetDescription

Finds named property and sets its description text.

UI Access	NA		
Parameters	Name	Type	Description
	<propName>	String	Name of the property
	<description>	String	Descriptive text.
Return Value	None		

Python Syntax	<code>SetDescription(<propName>, <description>)</code>
Python Example	<code>PropHost.SetDescription("W1", "this is the width of the gate")</code>

VB Syntax	<code>SetDescription(<propName>, <description>)</code>
VB Example	<code>PropHost.SetDescription("W1", "this is the width of the gate")</code>

SetHidden

Finds named property and sets its Hidden flag.

UI Access	NA		
Parameters	Name	Type	Description
	<propName>	String	Name of the property
	<hidden>	Boolean	True or 1, false or 0.
Return Value	None		

Python Syntax	SetHidden(<propName>, <hidden>)		
Python Example	<code>PropHost.SetHidden("W1", 1)</code>		

VB Syntax	SetHidden(<propName>, <hidden>)		
VB Example	<code>PropHost.SetHidden("W1", 1)</code>		

SetReadOnly

Finds named property and sets its ReadOnly flag.

UI Access	NA		
Parameters	Name	Type	Description
	<propName>	String	Name of the property
	<hidden>	Boolean	True or 1, false or 0.
Return Value	None		

Python Syntax	SetReadOnly(<propName>, <hidden>)
Python Example	PropHost.SetReadOnly("W1", 1)

VB Syntax	SetReadOnly(<propName>, <hidden>)
VB Example	PropHost.SetReadOnly("W1", 1)

SetText

Finds property in any tab and sets its value to a text string.

UI Access	NA		
Parameters	Name	Type	Description
	<propName>	String	Name of the property
	<text>	String	The value for the property.
Return Value	None		

Python Syntax	SetText(<propName>, <text>)
Python Example	PropHost.SetText("C", "22nF")

VB Syntax	SetText(<propName>, <text>)

VB Example	<code>PropHost.SetText("C", "22nF")</code>

SetValue

Finds property in any tab and sets its value to a double.

UI Access	NA		
Parameters	Name	Type	Description
	<propName>	String	Name of the property
	<value>	Real	A real number for the property value.
Return Value	None		

Python Syntax	<code>SetValue(<propName>, <value>)</code>
Python Example	<code>PropHost.SetValue("C", 2e-9)</code>

VB Syntax	<code>SetValue(<propName>, <value>)</code>
VB Example	<code>PropHost.SetValue("C", 2e-9)</code>

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>,  
      <NewPropArray>,  
      <ChangedPropsArray>,  
      <DeletedPropsArray>)
```

<PropServersArray>

```
Array("Name:PropServers", <PropServer>,  
      <PropServer>, ...)
```

<NewPropArray>

```
Array("Name:NewProp", <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 **RMxpert>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.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing names of variables.

Python Syntax	GetArrayVariables()
Python Example	<pre> oProject.GetArrayVariables() oDesign.GetArrayVariables() </pre>

VB Syntax	GetArrayVariables

VB Example	<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.

UI Access	N/A		
Parameters	Name	Type	Description
	<PropTab>	String	One of the following, where tab titles are shown in parentheses: - PassedParameterTab ("Parameter Values") - DefinitionParameterTab (Parameter Defaults") - LocalVariableTab ("Variables" or "Local Variables") - ProjectVariableTab ("Project variables") - ConstantsTab ("Constants") - BaseElementTab ("Symbol" or "Footprint") - ComponentTab ("General") - Component("Component") - CustomTab ("Intrinsic Variables") - Quantities ("Quantities") - Signals ("Signals")
	<PropServer>	String	An object identifier, generally returned from another script method, such as <code>CompInst@R;2;3</code>
Return Value	Array of strings containing the names of the appropriate properties.		

Python Syntax	GetProperties(<PropTab>, <PropServer>)
Python Example	oEditor.GetProperties('PassedParameterTab', 'k')

VB Syntax	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.

UI Access	N/A		
Parameters	Name	Type	Description
	<PropTab>	String	One of the following, where tab titles are shown in parentheses: - PassedParameterTab ("Parameter Values") - DefinitionParameterTab (Parameter Defaults") - LocalVariableTab ("Variables" or "Local Variables") - ProjectVariableTab ("Project variables") - ConstantsTab ("Constants") - BaseElementTab ("Symbol" or "Footprint") - ComponentTab ("General") - Component("Component")

			• CustomTab ("Intrinsic Variables") • Quantities ("Quantities") • Signals ("Signals")
	<i><PropServer></i>	String	An object identifier, generally returned from another script method, such as <code>CompInst@R;2;3</code>
	<i><PropName></i>	String	Name of the property.
Return Value	String value of the property.		

Python Syntax	<code>GetPropertyValue (<PropTab>, <PropServer>, <PropName>)</code>
Python Example	<pre>selectionArray = oEditor.GetSelections() for k in selectionArray: val = oEditor.GetPropertyValue("PassedParameterTab", k, "R") ... </pre>

VB Syntax	<code>GetPropertyValue (<PropTab>, <PropServer>, <PropName>)</code>
VB Example	<pre>selectionArray = oEditor.GetSelections for k in selectionArray: val = oEditor.GetPropertyValue("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`.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing the variables.

Python Syntax	<code>GetVariables ()</code>
Python Example	<code>oProject.GetVariables ()</code> <code>oDesign.GetVariables ()</code>

VB Syntax	<code>GetVariables</code>
VB Example	<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`.

UI Access	N/A

Parameters	Name	Type	Description
	<VarName>	String	Name of the variable to access.
Return Value	String represents the value of the variable.		

Python Syntax	GetVariableValue(<VarName>)
Python Example	oProject.GetVariableValue ("var_name")

VB Syntax	GetVariableValue <VarName>
VB Example	oProject.GetVariableValue "var_name"

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.

UI Access	N/A		
Parameters	Name	Type	Description
	<propTab>	String	One of the following, where tab titles are shown in parentheses: - PassedParameterTab ("Parameter Values") - DefinitionParameterTab (Parameter Defaults") - LocalVariableTab ("Variables" or "Local Variables")

			• ProjectVariableTab ("Project variables") • ConstantsTab ("Constants") • BaseElementTab ("Symbol" or "Footprint") • ComponentTab ("General") • Component("Component") • CustomTab ("Intrinsic Variables") • Quantities ("Quantities") • Signals ("Signals")
	<i><propServer></i>	String	An object identifier, generally returned from another script method, such as <code>CompInst@R;2;3</code>
	<i><propName></i>	String	Name of the property.
	<i><propValue></i>	String	The value for the property
Return Value	None.		

Python Syntax	<code>SetPropertyValue(<propTab>, <propServer>, <propName>, <propValue>)</code>
Python Example	<code>oEditor.SetPropertyValue("PassedParameterTab", "k", "R", "2200")</code>

VB Syntax	<code>SetPropertyValue <propTab>, <propServer>, <propName>, <propValue></code>
VB Example	<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`.

UI Access	N/A		
Parameters	Name	Type	Description
	<VarName>	String	Variable name.
	<VarValue>	Value	New value for the variable.
Return Value	None.		

Python Syntax	SetVariableValue (<VarName>, <VarValue>)
Python Example	<code>oProject.SetVariableValue('\$Var1', '3mm')</code>

VB Syntax	SetVariableValue <VarName>, <VarValue>
VB Example	<code>oProject.SetVariableValue "\$Var1", "3mm"</code>

Example Use of Record Script and Edit Properties

A simple way to see how to format the string arguments for a design object or property of interest is to use the script recording command and then edit the property. Open the script file and look at the `oEditor.ChangeProperty` entry to see the string arguments.

```
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("wg_combiner")
Set oDesign = oProject.SetActiveDesign("HFSSModel1")
Set oEditor = oDesign.SetActiveEditor("3D Modeler")

oEditor.ChangeProperty Array("NAME:AllTabs", Array("NAME:Geometry3DAttributeTab", Array
("NAME:PropServers", _
    "Polyline1"), Array("NAME:ChangedProps", Array("NAME:Display Wireframe", "Value:=", true), _
    Array("NAME:Display Wireframe", "Value:=", false), Array("NAME:Transparent", "Value:=", 0.2))))
```

Additional Property Scripting Examples

The following is a sample VB script that uses the `GetPropertyValue`, `SetPropertyValue`, and `GetProperties` functions. The script gets all the properties of the first `CreateBox` command of "Box1". It then loops through the properties and for each one, shows the user the current value and asks if the value should be changed.

```
Dim all_props
Dim prop

all_props = oEditor.GetProperties("Geometry3DCmdTab", _
    "Box1>CreateBox:1")

For Each prop In all_props
```

```
val = oEditor.GetPropertyValue("Geometry3DCmdTab", _  
    "Box1:CreateBox:1", prop)  
new_val = InputBox("New Value of " + prop + ":", _  
    "Current Value of '" + prop + "' is " + val, val)  
If new_val <> val Then  
oEditor.SetPropertyValue "Geometry3DCmdTab", _  
    "Box1:CreateBox:1", prop, new_val  
val = _  
    oEditor.SetPropertyValue("Geometry3DCmdTab", _  
        "Box1:CreateBox:1", prop)  
MsgBox("Now the value of '" + prop + "' is " + val)  
End If  
Next
```

The following is a sample VB script that creates a HFSS 3D Layout design, draws a rectangle in the layout editor and uses the `GetPropertyValue`, `SetPropertyValue` and `GetProperties` functions. The script gets all properties of the rectangle. It then loops through the properties and for each one, shows the user the current value and asks if the value should be changed. Note that the last call to `GetPropertyValue` in the script will fail if you change the name of the rectangle from the script.

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



```
Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")
Set oDesktop = oAnsoftApp.GetAppDesktop()
,
oDesktop.RestoreWindow
oDesktop.NewProject
Set oProject = oDesktop.GetActiveProject
,
' CREATE A RECTANGLE IN HFSS_3D_LAYOUT
,
oProject.InsertDesign "HFSS 3D Layout", "HFSS 3D Layout1", _
"C:\Program Files\ANSYS Inc\v251\AnsysEM\syslib\PCB - SingleSided.asty", ""
Set oDesign = oProject.SetActiveDesign("HFSS 3D Layout1")
Set oEditor = oDesign.SetActiveEditor("Layout")
oEditor.CreateRectangle Array("NAME:Contents", _
"rectGeometry:=", Array("Name:=", _
"rect_1", "LayerName:=", "Top", "lw:=", _
"0mm", "Ax:=", "-22mm", "Ay:=", "20mm", "Bx:=", _
"29mm", "By:=", "-4mm", "ang:=", "0deg"))
,
' GET ALL PROPERTIES OF THE RECTANGLE
```

```
'  
  
Dim all_props  
Dim prop  
Dim val  
Dim new_val  
'  
  
all_props = oEditor.GetProperties("BaseElementTab","rect_1")  
'  
  
' LOOP OVER ALL PROPERTIES  
'  
  
For Each prop in all_props  
val = oEditor.GetPropertyValue("BaseElementTab","rect_1",prop)  
'  
  
' DISPLAY VALUE TO THE USER  
'  
  
new_val = InputBox("New Value of "+prop+":",_  
"Current Value of "+prop+" is "+val,val)  
'  
  
' CHANGE THE VALUE IF DESIRED  
'  
  
If new_val <> val Then
```

```
oEditor.SetPropertyValue _  
"BaseElementTab", "rect_1", prop, new_val  
val = _  
oEditor.GetPropertyValue("BaseElementTab", "rect_1", prop)  
MsgBox("Now the value of "+prop+" is "+val)  
End If  
,  
Next  
,
```

3 - 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.

UI Access	NA		
Parameters	Name	Type	Description
	None		
Return Value	Object		

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

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

4 - 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. See: [Object Oriented Property Scripting](#).

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

[AddMessage](#)

[AreThereSimulationsRunning](#)

[ClearMessages](#)

[CloseAllWindows](#)

[CloseProject](#)

[CloseProjectNoForce](#)

[DeleteProject](#)

[DeleteRegistryEntry](#)

[DoesRegistryValueExist](#)

[DownloadJobResults](#)

[EnableAutoSave](#)

[ExportOptionsFiles](#)

[GetActiveProject](#)

[GetActiveScheduler](#)

[GetActiveSchedulerInfo](#)

[GetAutoSaveEnabled](#)

[GetBuildDateTimeString](#)

[GetCustomMenuSet](#)

[GetDefaultUnit](#)

[GetDesktopConfiguration](#)

[GetDistributedAnalysisMachines](#)

[GetDistributedAnalysisMachinesForDesignType](#)

[GetExeDir](#)

[GetGDIObjectCount](#)

[GetLibraryDirectory](#)

[GetLocalizationHelper](#)

[GetMessages](#)

[GetMonitorData](#)

[GetPersonalLibDirectory](#)

[GetProcessID](#)

[GetProjectDirectory](#)

[GetProjectList](#)

[GetProjects](#)

[GetRegistryInt](#)

[GetRegistryString](#)

[GetRunningInstancesMgr](#)

[GetSchematicEnvironment](#)

[GetScriptingToolsHelper](#)

[GetSysLibDirectory](#)

[GetTempDirectory](#)

[GetUserLibDirectory](#)

[GetVersion](#)

[IsFeatureEnabled](#)

[LaunchJobMonitor](#)

[NewProject](#)

[OpenAndConvertProject](#)

[OpenMultipleProjects](#)

[OpenProject](#)

[OpenProjectWithConversion](#)

[PageSetup](#)

[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](#)

[TileWindows](#)

Related Topics:

[Desktop Commands For Registry Values](#)

[ImportExport Tool Commands](#)

AddMessage

Add a message with severity and context to message window.

UI Access	N/A		
Parameters	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.
Return Value	None.		

Python Syntax	AddMessage(<projectName>, <designName>, <severity>, <msg>, <category>)
Python Example	<pre>oDesktop.AddMessage("Project1", "HFSS 3D Layout", 0, "This is a test message", "")</pre>

VB Syntax	AddMessage <projectName>, <designName>, <severity>, <msg>, <category>
VB Example	oDesktop.AddMessage "Project1", "HFSS 3D Layout1", 0, "This is a test message", ""

AreThereSimulationsRunning

Returns a bool specifying whether there are simulations running, or either running or pending, depending on the inclusion of alsoPending.

UI Access	NA		
Parameters	Name	Type	Description
	<bool> <AlsoPending>	Integer	Whether to report if simulations are running, or running and pending.
Return Value	A human readable status string specifying what happened.		

Python Syntax	AreThereSimulationsRunning (<bool>, clean)
Python Example	oDesktop.AreThereSimulationsRunning(bool clean)

ClearMessages

For a specified project and design, this command clears all messages in a specified severity range. The user-specified integral severity level is interpreted as:

0 => info, 1 => warning, 2 => error, and 3 => fatal error.

The ClearMessages function accepts four input arguments. The first two specifies project and design, respectively. This function clears messages in the range specified by the third (interpreted as start severity) and fourth (interpreted as stop severity) arguments. The fourth argument has a default value of 0. The last two arguments need not be specified in any given order, i.e., they can indicate either ascending or descending trend. The function clears all messages having severity level within this specified range. The start and stop severity need not be specified in any given order (i.e., they can indicate either ascending or descending severity).

UI Access	In Message Manager , right-click and select Clear messages for [ProjectName]...		
Parameters	Name	Type	Description
	<projectName>	String	Name of the project from which to clear messages.
	<designName>	String	Name of the design under the <projectName> from which to clear messages.
	<startSeverity>	Integer	User-specified severity level at which messages start getting cleared: • 0 = informational • 1 = warning • 2 = error • 3 = fatal error
	<stopSeverity>	Integer	<i>Optional.</i> User-specified stop severity level until which messages will be cleared. • 0 = informational • 1 = warning • 2 = error • 3 = fatal error If not specified, <stopSeverity> has a default value of 0.
Return Value	None.		

Python Syntax	ClearMessages(<projectName>, <designName>, <startSeverity>, <stopSeverity>)
Python Example	Preserve Current Functionality; stopSeverity = 0 (DEFAULT VALUE) <pre># startSeverity = 0; clears just the Info messages oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 0) # startSeverity = 1; clears all the Info and Warning messages oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 1) # startSeverity = 2; clears all the Info, Warning, Error oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 2) # startSeverity = 3; clears all messages, i.e., Info, Warning, Error, Fatal oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 3)</pre> Extend the current functionality by enabling range-based deletion <pre># startSeverity = 0; clears the Info messages; oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 0) # startSeverity = 0 and stopSeverity = 0; clears all the Info messages oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 0, 0) # startSeverity = 0 and stopSeverity = 1; clears all the Info and Warning messages oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 0, 1) # startSeverity = 0 and stopSeverity = 2; clears all the Info, Warning, and Error messages oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 0,</pre>

```
2)

# startSeverity = 0 and stopSeverity = 3; clears all the Info, Warning, Error,
and Fatal messages
oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 0,
3)

# startSeverity = 1; clears all the Info and Warning messages;
oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 1)
# startSeverity = 1 and stopSeverity = 1; clears all the Warning messages
oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 1, 1)
# startSeverity = 1 and stopSeverity = 2; clears all the Warning and Error mes-
sages
oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 1, 2)
# startSeverity = 1 and stopSeverity = 3; clears all the Warning, Error, and
Fatal messages
oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 1, 3)

# startSeverity = 2; clears all the Info, Warning, and Error messages;
oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 2)
# startSeverity = 2 and stopSeverity = 2; clears all the Error messages
oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor", "Maxwell3DDesign1", 2, 2)
# startSeverity = 2 and stopSeverity = 3; clears all the Error and Fatal messages
```

	<pre> oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor","Maxwell3DDesign1",2, 3) # startSeverity = 2 and stopSeverity = 0; clears all the Info, Warning, and Error messages oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor","Maxwell3DDesign1",2, 0) # startSeverity = 3; clears all the Info, Warning, Error, and Fatal messages; oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor","Maxwell3DDesign1",3) # startSeverity = 3 and stopSeverity = 3; clears all the Fatal error messages oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor","Maxwell3DDesign1",3, 3) # startSeverity = 3 and stopSeverity = 0; clears all the Info, Warning, Error, and Fatal messages oDesktop.ClearMessages("Ex_EC_1_Asymmetrical_Conductor","Maxwell3DDesign1",3, 0) </pre>

VB Syntax	<code>ClearMessages <projectName>, <designName>, <startSeverity>, <stopSeverity></code>
VB Example	<pre> Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop") Set oDesktop = oAnsoftApp.GetAppDesktop() oDesktop.ClearMessages "MyProject", "MyDesign", 0, 3 </pre>

CloseAllWindows

Closes all MDI child windows on the desktop.

UI Access	From main menu, Window > CloseAll .

Parameters	None.
Return Value	None.

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

VB Syntax	CloseAllWindows
VB Example	<code>oDesktop.CloseAllWindows</code>

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	Type	Description
	<ProjectName>	String	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	<code>oDesktop.CloseProject ("MyProject")</code>

VB Syntax	<code>CloseProject <ProjectName></code>
VB Example	<code>oDesktop.CloseProject "MyProject"</code>

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.

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

Python Syntax	<code>CloseProjectNoForce (<ProjectName>)</code>
Python Example	<code>oDesktop.CloseProjectNoForce ("MyProject")</code>

VB Syntax	<code>CloseProjectNoForce <ProjectName></code>

VB Example	<code>oDesktop.CloseProjectNoForce "MyProject"</code>

Count

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

UI Access	N/A
Parameters	None.
Return Value	Integer
Python Syntax	Count
Python Example	<pre>projects = oDesktop.GetProjects() numprojects = projects.Count</pre>

VB Syntax	Count
VB Example	<pre>Dim oAnsoftApp Dim oDesktop Dim oProject Dim oDesign Dim oEditor Dim oModule Dim oProjects Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop")</pre>

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

DeleteProject

Deletes a project from disk.

UI Access	Edit > Delete		
Parameters	Name	Type	Description
	<ProjectName>	String	Name of the project
Return Value	None		

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

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

DownloadJobResults

This command is for downloading results from Ansys Cloud. Before using this script command, the command [SelectScheduler\(\)](#) must be used first to select “ansys cloud” scheduler. This makes sure that current scheduler is Ansys Cloud, and user is logged in. Then, either a valid .q file or .q.completed file must be in the project folder, or, a valid job ID and a “batchinfo” folder containing the corresponding .jobid file are required in the project folder. When the download requirements are met, the command downloads results from Ansys Cloud using the specified filters to the given folder.

UI Access	Select Scheduler		
Parameters	Name	Type	Description
	<jobID>	String	Provide the job ID of the target job. The job ID must be able to be found in current .q (or .q.completed) file, or inside the “batchinfo” folder in projectPath. If the job ID is empty, the job ID in current .q (or .q.completed) file will be used.
	<projectPath>	String	A string of path to locate the project folder. The project file may be not necessary, but .q (or .q.completed) file or “batchinfo” folder with valid .jobid files are required.
	<resultPath>	String	A string giving the folder path for the download to save to.
	<Filters> (optional)	String	A string containing filters to download. The delimiter of file types is “;”. If no filter specified, the default filter “*” will be applied, which requests all files for download.

Return Value	A Boolean result about download complete or not

Python Syntax	<code>DownloadJobResults(<i>jobID</i>, <i>projectPath</i>, <i>resultsPath</i>, <i>filters</i>)</code>
Python Example	<pre>boolDownloadCompleted = oDesktop.DownloadJobResults("012345678901234567890", "C:\\projects\\basic.aedt", "C:\\projects\\DownloadResults\\", "**")</pre>

VB Syntax	<code>DownloadJobResults(<i>jobID</i>, <i>projectPath</i>, <i>resultsPath</i>, <i>filters</i>)</code>
VB Example	<pre>boolDownloadCompleted = oDesktop. DownloadJobResults ("012345678901234567890", "C:\\projects\\basic.aedt", "C:\\projects\\DownloadResults\\", "**")</pre>

DeleteRegistryEntry

Deletes a registry entry from a registry key. Returns true if deletion succeeded.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><pathToRegistry></code>	String	Path to Registry entry.
Return Value	Boolean: • True – Key has been deleted. • False – Key does not exist.		

Python Syntax	<code>DeleteRegistryEntry(<pathToRegistry>)</code>

Python Example	<pre>res = oDesktop.DeleteRegistryEntry("Desktop/ColorScheme")</pre>

VB Syntax	DeleteRegistryEntry <pathToRegistry>
VB Example	<pre>res = oDesktop.DeleteRegistryEntry "Desktop/ColorScheme"</pre>

DoesRegistryValueExist

Determines whether a registry value exists.

UI Access	N/A		
Parameters	Name	Type	Description
	<KeyName>	String	Full name of registry key, including path.
Return Value	Boolean: • True – Key exists. • False – Key does not exist.		

Python Syntax	DoesRegistryValueExist(<KeyName>)
Python Example	<pre>Exist = oDesktop.DoesRegistryValueExist('Desktop/ActiveDSOConfigurations/HFSS 3D Lay-out')</pre>

VB Syntax	DoesRegistryValueExist(<KeyName>)
VB Example	<code>bExist = oDesktop.DoesRegistryValueExist("Desktop/ActiveDSOConfigurations/HFSS 3D Layout")</code>

EnableAutoSave

Enable or disable autosave feature.

UI Access	N/A		
Parameters	Name	Type	Description
	<enable>	Boolean	True to enable autosave; False disables it.
Return Value	None.		

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

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

ExportOptionsFiles

Copies the options config files to the *DestinationDirectory*.

UI Access	Tools > Options > Export Options Files ...		
Parameters	Name	Type	Description
	<DestinationDirectory>	String	The path to the destination directory.
Return Value	None		

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

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

GetActiveProject

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

Note:

GetActiveProject returns normally if there are no active objects.

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

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

VB Syntax	GetActiveProject
VB Example	<pre>Set oProject = oDesktop.GetActiveProject</pre>

GetActiveScheduler

Obtains the name of the scheduler active in the Desktop.

Note:

GetActiveScheduler returns normally if there are no active objects.

UI Access	N/A
Parameters	None.
Return Value	Name of the scheduler that is currently active in the desktop. Gets results such as "RSM", "Remote RSM", "Ansys Cloud", "SLURM", "LSF", "PBS", "SGE", "Windows HPC", etc. (User-friendly scheduler names.)

Python Syntax	GetActiveScheduler()
Python Example	<pre>oSchedulerName = oDesktop.GetActiveScheduler()</pre>

VB Syntax	GetActiveScheduler
VB Example	<code>Set oSchedulerName = oDesktop.GetActiveScheduler</code>

GetActiveSchedulerInfo

Obtains info for the scheduler active in the Desktop.

Note:

GetActiveSchedulerInfo returns normally if there are no active objects.

UI Access	N/A
Parameters	None.
Return Value	Info for the scheduler that is currently active in the desktop or a remote scheduler. Gets results such as "RSM", "Remote RSM", "Ansys Cloud", "SLURM", "LSF", "PBS", "SGE", "Windows HPC", as well as server info used to select a server. For a local scheduler, the return format resembles '{"Selected Scheduler":"RSM","Scheduler Name":"RSM","Server":""}'. For a remote scheduler, the return format resembles '{"Selected Scheduler":"Remote RSM","Scheduler Name":"SLURM","Server":"<server>"}'

Python Syntax	GetActiveSchedulerInfo()
Python Example	<code>oSchedulerInfo = oDesktop.GetActiveSchedulerInfo()</code>

VB Syntax	GetActiveScheduler
VB Example	<pre>Set oSchedulerInfo = oDesktop.GetActiveSchedulerInfo</pre>

GetAutoSaveEnabled

Checks whether the autosave feature is enabled.

UI Access	N/A
Parameters	None.
Return Value	Integer: • 1 – Autosave is enabled. • 0 – Autosave is not enabled.

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

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

GetBuildDateTimeString

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

UI Access	N/A

Parameters	None.
Return Value	String build date and time, in the format: year-month-day hour:minute:second. Example: 2019-01-18 21:59:33

Python Syntax	GetBuildDateTimeString()
Python Example	<code>oDesktop.GetBuildDateTimeString()</code>

VB Syntax	GetBuildDateTimeString
VB Example	<code>dnt = oDesktop.GetBuildDateTimeString</code>

GetCustomMenuSet

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

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

Python Syntax	GetCustomMenuSet ()

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

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

GetDefaultUnit

Returns the default unit for a physical quantity.

UI Access	Tools > Options > General Options > Default Units. Note that this menu only displays units that can be changed, while the script can be used to view additional default units.		
Parameters	Name	Type	Description
	<type>	String	String containing a type of measurement. Valid strings are (case insensitive): "Acceleration" "Angle" "AngularAcceleration" "AngularDamping" "AngularSpeed" "Capacitance" "Conductance" "Current" "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" • "VoltageChangeRate" • "Volume" • "VolumeFlowPerPressureRoot"

	<table><tr><td></td><td></td><td>• "VolumeFlowRate"</td></tr></table>			• "VolumeFlowRate"
		• "VolumeFlowRate"		
Return Value	String containing the default unit (for example, "mm").			

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

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

GetDesktopConfiguration

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

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

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

VB Syntax	GetDesktopConfiguration
VB Example	<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.

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

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

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

GetDistributedAnalysisMachinesForDesignType

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

UI Access	NA

Parameters	Name	Type	Description
	<designTypeName>	String	The name of the type of design, such as "Twin Builder", "HFSS", "HFSS-IE", "Maxwell 3D", "Maxwell 2D", "RMxpert", "EM Design", "Circuit", "System", "Q3D Extractor", "2D Extractor"
Return Value	Object; returns a collection of machine names.		

Python Syntax	GetDistributedAnalysisMachinesForDesignType (<designTypeName>)
Python Example	<pre>machineNames =oDesktop. GetDistributedAnalysisMachinesForDesignType ("HFSS 3D Layout")</pre>

VB Syntax	GetDistributedAnalysisMachinesForDesignType <designTypeName>
VB Example	<pre>Set machineNames =oDesktop. GetDistributedAnalysisMachinesForDesignType ("HFSS 3D Layout")</pre>

GetExeDir

Returns the path where the executable is located.

UI Access	N/A
Parameters	None.
Return Value	String path where executable is located.

	Example: "C:\Program Files\ANSYS Inc\v251\AnsysEM"

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

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

GetGDIObjectCount

Note:

This command is for internal Ansys use only.

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

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

GetLibraryDirectory

Get the path to the SysLib directory.

UI Access	NA		
Parameters	Name	Type	Description
	None		
Return Value	String The path to the SysLib directory.		

Python Syntax	GetLibraryDirectory()
Python Example	AddInfoMessage (str (oDesktop.GetLibraryDirectory()))

VB Syntax	GetLibraryDirectory
VB Example	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.

UI Access	NA		
Parameters	Name	Type	Description
	None		
Return Value	Object		
	Localization helper object, such as "IDispatch(ILocalizationHelper)"		

Python Syntax	GetLocalizationHelper()
Python Example	oDesktop.GetLocalizationHelper()

VB Syntax	GetLocalizationHelper
VB Example	oDesktop.GetLocalizationHelper()

GetMessages

Get the messages from a specified project and design.

UI Access	NA		
Parameters	Name	Type	Description
	<ProjectName>	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)
	<DesignName>	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
	<Severity>	Integer	Severity is 0-3, and is tied in to info/warning/error/fatal types as follows: • 0 – info and above • 1 – warning and above • 2 – error and fatal • 3 – fatal only (rarely used)
Return Value	Array of string messages.		

Python Syntax	GetMessages (<ProjectName>, <DesignName>, <Severity>)
Python Example	Messages = oDesktop.GetMessages("MyProject", "HFSS 3D Layout1", 1)

VB Syntax	GetMessages <ProjectName>, <DesignName>, <Severity>
VB Examples	Messages = oDesktop.GetMessages "MyProject", "HFSS 3D Layout1", 1

GetMonitorData

Get monitor data. This command is requires a -monitor flag on the ansyedt.exe command line and is used only with the web client.

UI Access	NA		
Parameters	Name	Type	Description
	<Request>	String	a json string describing what monitor data is desired. This is a list of triples separated by spaces of the form: <code>type startposition maxnumber</code>. The <code>type</code> must be one of <code>convergence profile sweptvar progress variations, displaytype</code>, or a guide that was previously returned in a <code>displaytype</code> block. The <code>startposition</code> is an integer that specifies where to start in the sequence of monitor data of that type. Each of these types is available in a list, ordered by time. For example, as the simulation goes on, data points are added to the internal list of convergence items. If the caller needs to incrementally update a display, it can repeatedly call this method with the <code>startposition</code> for each of the types given as the size of the list it has previously obtained and displayed, which ensures there will be no duplication

			and the returned data will be new each time. The <code>maxnumber</code> , if greater than 0, is the limit on the number of new items returned for each type. If the caller is able to handle any number of returned values, it can be left at 0.
Return Value	a json string with the results.		

Python Syntax	<code>GetMonitorData (<Request>)</code>
Python Example	<code>Messages = oDesktop.GetMonitorData(request)</code>

GetPersonalLibDirectory

Get the path to the `PersonalLib` directory.

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

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

VB Syntax	<code>GetPersonalLibDirectory</code>

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

GetProcessID

Returns the process ID of ansysedt.exe.

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

Python Syntax	<code>GetProcessID ()</code>
Python Example	<code>oDesktop.GetProcessID()</code>

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

GetProjectDirectory

Gets the path to the Project directory.

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

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

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

GetProjectList

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

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

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

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

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

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

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

GetRegistryInt

Obtains registry key integer value.

UI Access	N/A		
Parameters	Name	Type	Description
	<KeyName>	String	Full name of registry key, including path.

Return Value	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.

Python Syntax	GetRegistryInt(<KeyName>)
Python Example	<pre>num = oDesktop.GetRegistryInt('Desktop/Settings/ProjectOptions/HFSS 3D Layout/UpdateReportsDynamicallyOnEdits')</pre>

VB Syntax	GetRegistryInt(<KeyName>)
VB Example	<pre>num = oDesktop.GetRegistryInt("Desktop/Settings/ProjectOptions/HFSS 3D Layout/UpdateReportsDynamicallyOnEdits")</pre>

GetRegistryString

Obtains registry key string value.

UI Access	N/A		
Parameters	Name	Type	Description
	<KeyName>	String	Full name of registry key, including path.
Return Value	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.		

Python Syntax	GetRegistryString(<KeyName>)
Python Example	<pre>activeDSO = oDesktop.GetRegistryString('Desktop/ActiveDSOConfigurations/HFSS 3D Layout')</pre>

VB Syntax	GetRegistryString(<KeyName>)
VB Example	<pre>activeDSO = oDesktop.GetRegistryString("Desktop/ActiveDSOConfigurations/HFSS 3D Layout")</pre>

GetRunningInstancesMgr

Returns the object of the Running Instances Manager.

UI Access	N/A		
Parameters	Name	Type	Description
	None		
Return Value	Object Running instances manager object		

Python Syntax	GetRunningInstancesMgr()
Python Example	<pre>oRunningInstances = oDesktop.GetRunningInstanceMgr()</pre>

VB Syntax	GetRunningInstancesMgr()
VB Example	Set oRunningInstances = oDesktop.GetRunningInstancesMgr()

GetSchematicEnvironment

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

UI Access	N/A
Parameters	None.
Return Value	Integer representing a schematic environment: • 0 = Circuit • 1 = Twin Builder • 2 = Maxwell Circuit

Python Syntax	GetSchematicEnvironment()
Python Example	oDesktop.GetSchematicEnvironment()

VB Syntax	GetSchematicEnvironment
VB Example	Set oProject = oDesktop.GetSchematicEnvironment

GetScriptingToolsHelper

Note:
This command is for internal Ansys use only.

Returns the object for the scripting tools helper.

UI Access	NA		
Parameters	Name	Type	Description
	None		
Return Value	Object ScriptingTools helper object		

Python Syntax	GetScriptingToolsHelper()
Python Example	oDesktop.GetScriptingToolsHelper()

VB Syntax	GetScriptingToolsHelper
VB Example	oDesktop.GetScriptingToolsHelper()

GetSysLibDirectory

Get the path to the SysLib directory.

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

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

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

GetTempDirectory

Gets the path to the `Temp` directory.

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

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

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

GetUserLibDirectory

Gets the path to the UserLib directory.

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

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

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

GetVersion

Returns a string representing the version.

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

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

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

IsFeatureEnabled

Returns a Boolean for whether a queried feature is enabled.

UI Access	N/A
Parameters	<FeatureID>.
Return Value	Boolean for named feature.

Python Syntax	IsFeatureEnabled()

Python Example	<pre>import ScriptEnv ScriptEnv.Initialize("Ansoft.ElectronicsDesktop") oDesktop.RestoreWindow() feature_strs = ["SF3519", "F195709_EXPORT_TO_EMIT", "F362235_EMIT_RESULTS_WINDOW_ IMPROVEMENTS",_ "F136736_SBR_Rough_Surface", "F353006_VOLUMETRIC_SBR", "F359673_SBR_MULTISTATE_ARRAY", "F393115_SWE_ANTENNA",_ "S196592_SBR_Directivity", "F432541_VSBR_IMPROVEMENTS", "F353007_VRT_FILTERS_ENHANCE", "S540337_SBR_REGION_LOSS_DIRECTIVITY",_ "F11941_VRT_CURRENT_DENSITY", "S544593_SBR_3D_COMPONENT_ARRAY", "F539850_SBR_GO_ BLOCKAGE", "F540275_SBR_RAYSTATS"] results = [oDesktop.IsFeatureEnabled(str) for str in feature_strs] result_txt = open("C:/Users/MyResults/Downloads/results.txt", "w") for i in range(len(feature_strs)): result_txt.write('%s : %s\n' % (feature_strs[i], results[i])) result_txt.close()</pre>

VB Syntax	GetVersion()
VB Example	results = oDesktop.IsFeatureEnabled(SF3519)

KeepDesktopResponsive

Specifies the minimum number of milliseconds to keep the desktop from showing hung.

UI Access	N/A		
Parameters	Name	Type	Description
	<MinTimeInMilliseconds>	Integer	The minimum number of milliseconds to keep the desktop window from showing hung, that is, not responding.
Return Value	Boolean True for success and to keep running; False to indicate that the calling script should shut down.		

Python Syntax	KeepDesktopResponsive (<TimeInMilliseconds>)		
Python Example	<code>oDesktop.KeepDesktopResponsive(10000)</code>		

VB Syntax	KeepDesktopResponsive <TimeInMilliseconds>		
VB Example	<code>oDesktop.KeepDesktopResponsive 10000</code>		

LaunchJobMonitor

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

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

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

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

NewProject

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

UI Access	File > New.
Parameters	None.
Return Value	Object, the project that is added.

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

VB Syntax	NewProject

VB Example	Set oProject = oDesktop.NewProject

OpenAndConvertProject

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

UI Access	Click File > Open , and choose a legacy project		
Parameters	Name	Type	Description
	<itemPath>	String	full project path of the legacy project
	<legacyChoice>	Integer	0: show conversion dialog box, (same as File > Open 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)
Return Value	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.

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

VB Syntax	OpenAndConvertProject(<i>filePath</i> , <i>legacyChoice</i>)

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

OpenMultipleProjects

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

UI Access	N/A		
Parameters	Name	Type	Description
	<directory>	String	Path to the projects.
	<file Type>	String	Type of projects to open.
Return Value	None.		

Python Syntax	OpenAndConvertProject(<filePath>, <legacyChoice>)
Python Example	<pre>oProject = oDesktop.OpenAndConvertProject("c:\files\optimtee.", "*.aedt")</pre>

VB Syntax	OpenAndConvertProject <filePath>, < legacyChoice>
VB Example	<pre>Set oProject = oDesktop.OpenAndConvertProject "c:\files\optimtee.", "*.aedt"</pre>

OpenProject

Opens a specified project.

UI Access	Click File > Open .		
Parameters	Name	Type	Description
	<FileName>	String	Full path of the project to open.
Return Value	An object reference to the newly opened project.		

Python Syntax	OpenProject(<FileName>)		
Python Example	oDesktop.OpenProject("D:/Projects/Project1.aedt")		

VB Syntax	OpenProject <FileName>		
VB Example	oDesktop.OpenProject "D:/Projects/Project1.aedt"		

OpenProjectWithConversion

Note:

This command is for internal Ansys use only.

Python Syntax	OpenProjectWithConversion()		
Python Example	oDesktop.OpenProjectWithConversion()		

PageSetup (Layout Editor)

Use: Specifies page setup for printing.

Command: **File>Page Setup**

Syntax: PageSetup <ArgArray>

Return Value: None.

Parameters: <Margins>: Page margins in implicit units of inches.

 <Border>: Integer value indicating to draw border (1) or not to draw (0).

 <DesignVars>: Integer value indicating to draw design vars (1) or not to draw (0).

VB Example: Set oProject = oDesktop.GetActiveProject()
Set oDesign = oProject.GetActiveDesign()
Set oEditor = oDesign.GetActiveEditor()
oEditor.PageSetup Array("NAME:PageSetupData", "margins:=", Array("left:=", 550, "right:=", _
550, "top:=", 500, "bottom:=", 500), "border:=", 1, "DesignVars:=", 0)

PauseRecording

Temporarily stop script recording.

UI Access	NA

Parameters	Name	Type	Description
	None		
Return Value	None		

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

VB Syntax	PauseRecording
VB Example	<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.

UI Access	Tools > Pause Script		
Parameters	Name	Type	Description
	<Message>	String	Any Text.
Return Value	None		

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

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

Print

Prints the contents of the active view window.

UI Access	File > Print.
Parameters	None.
Return Value	None.

Python Syntax	Print()
Python Example	oDesktop.Print()

VB Syntax	Print
VB Example	oDesktop.Print

QuitApplication

Exits the desktop.

UI Access	File > Exit.
Parameters	None.
Return Value	None.

Python Syntax	QuitApplication()
Python Example	<code>oDesktop.QuitApplication()</code>

VB Syntax	QuitApplication
VB Example	<code>oDesktop.QuitApplication</code>

RefreshJobMonitor

For use in monitoring a job.

UI Access	Tools > Job Management > Monitor Jobs.
Parameters	None.
Return Value	A string specifying the job state. The result can be any of the following strings:

	• "Monitor Not Visible" • "Queued" • "Running" • "Shutting Down" • "Unknown" • "Completed" • "Not Monitoring" • "Starting Monitoring"

Python Syntax	RefreshJobMonitor()
Python Example	<code>oDesktop.RefreshJobMonitor()</code>

VB Syntax	RefreshJobMonitor
VB Example	<code>oDesktop.RefreshJobMonitor</code>

ResetLogging

Redirects simulation log file to a specified directory and log level.

UI Access	N/A		
Parameters	Name	Type	Description

	<logFile>	String	Path to log file.
	<logLevel>	Integer	Specified log level.
Return Value	None.		

Python Syntax	ResetLogging(<logFile>, <logLevel>)
Python Example	oDesktop.ResetLogging("C:/Project1.aedtresults/", 1)

VB Syntax	ResetLogging <logFile>, <logLevel>
VB Examples	oDesktop.ResetLogging "C:/Project1.aedtresults/", 1

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	<code>RestoreProjectArchive (<Archivefilepath>, <ProjectFilePath>, <OverwriteExistingFiles>, <OpenProjectAfterRestore>)</code>
Python Example	<code>oDesktop.RestoreProjectArchive ("C:\Users\jdoe\Documents\OptimTee.aedt", "C:\Documents\OptimTee.aedt", False, True)</code>

VB Syntax	<code>RestoreProjectArchive <Archivefilepath>, <ProjectFilePath>, <OverwriteExistingFiles>, <OpenProjectAfterRestore></code>
VB Example	<code>oDesktop.RestoreProjectArchive "C:\Users\jdoe\Documents\OptimTee.aedt", _ "C:\Documents\OptimTee.aedt", false, true</code>

RestoreWindow

Restores a minimized Desktop window.

UI Access	N/A
Parameters	None.
Return Value	None.

Python Syntax	<code>RestoreWindow()</code>
Python Example	<code>oDesktop.RestoreWindow()</code>

VB Syntax	RestoreWindow
VB Example	<code>oDesktop.RestoreWindow</code>

ResumeRecording

Resume recording a script.

UI Access	N/A
Parameters	None.
Return Value	None

Python Syntax	ResumeRecording()
Python Example	<code>oDesktop.ResumeRecording()</code>

VB Syntax	ResumeRecording
VB Example	<code>oDesktop.ResumeRecording</code>

RunACTWizardScript

Note:

This command is for internal Ansys use only.

Python Syntax	RunACTWizardScript()
Python Example	<code>oDesktop.RunACTWizardScript()</code>

RunProgram

Runs an external program.

UI Access	NA		
Parameters	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 <code>None</code>
Return Value	None		

Python Syntax	RunProgram (<ProgName>, <ProgPath>, <WorkPath>, <ArgArray>)
Python Example	<pre>oDesktop.RunProgram("winword.exe", _ "C:\Program Files\Microsoft Office\Office10", _ "", None)</pre>

VB Syntax	RunProgram <ProgName>, <ProgPath>, <WorkPath>, <ArgArray>
VB Example	<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	Type	Description
	<SCriptPath>	String	Name or full path of the script to execute. If the full path to the script is not specified, Twin Builder searches for the specified script in the following locations, in this order: 1. Personal library directory. This is the PersonalLib subdirectory in the project directory. The project directory can be specified in the General Options dialog box (click >Tools > Options > General Options to open this dialog box) under the Project Options tab. 2. User library directory. This is the userlib subdirectory in the library directory. The library directory can be specified in the General Options dialog in the box (click Tools > Options > General Options to open this dialog box) under the Project Options tab. 3. System library directory. This is the syslib subdirectory in the library directory. The library dir-

	<table><tr><td></td><td></td><td>ectory can be specified in the General Options dialog box (click Tools > Options > General Options to open this dialog box) under the Project Options tab.</td></tr><tr><td></td><td></td><td>4. HFSS installation directory.</td></tr></table>			ectory can be specified in the General Options dialog box (click Tools > Options > General Options to open this dialog box) under the Project Options tab.			4. HFSS installation directory.
		ectory can be specified in the General Options dialog box (click Tools > Options > General Options to open this dialog box) under the Project Options tab.					
		4. HFSS installation directory.					
Return Value	Long the return code for the script method.						

Python Syntax	RunScript (<ScriptPath>)
Python Example	oDesktop.RunScript ("C:/Project/test1.vbs")

VB Syntax	RunScript <ScriptPath>
VB Example	oDesktop.RunScript ("C:/Project/test1.vbs")

RunScriptWithArguments

Similar to RunScript, launch another script from within the currently executing script, but with arguments.

UI Access	NA		
Parameters	Name	Type	Description
	<ScriptPath>	String	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:

			- Personal library directory: "PersonalLib" The PersonalLib directory can be specified in Tools > Options > General Options on the 'Project Options' tab. - User library directory: "userlib" The UserLib directory can be specified in Tools > Options > General Options on the 'Project Options' tab. - System library directory: "syslib" The SysLib directory can be specified in Tools > Options > General Options on the 'Project Options' tab. - Software installation directory
	<Arguments>	String	The arguments to supply to the specified script.
Return Value	Long the return code for the script method.		

Python Syntax	RunScriptWithArguments (<ScriptPath>, <Arguments>)
Python Example	<pre>oDesktop.RunScriptWithArguments ("C:/Project/test2.py", "foo")</pre>

VB Syntax	RunScriptWithArguments <ScriptPath>, <Arguments>
VB Example	<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** window and waits for the user to complete the settings.

UI Access	Tools > Job Management > Select Scheduler.		
Parameters	Name <option>	String	One of the following options (not case sensitive): • Empty string for remote RSM service • "RSM" for local RSM • "Windows HPC" for Windows HPC • "LSF" for Load-Sharing Facility • "SGE" for Grid Engine (GE, OGE, SGE, UGE, etc.) • "PBS" for Portable Batch Scheduler/System (PBSPRO, Torque, Maui, etc.) • "Ansys Cloud" for Ansys Cloud
	<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.
Return Value	The selected scheduler (if selection was successful, this string should match the input option string, although it		

	could differ in upper/lowercase).

Python Syntax	Select Scheduler(<option>, <address>, <username>, <forcePasswordEntry>)
Python Example	<pre>result = oDesktop.SelectScheduler("Windows HPC", "headnode.win.example.com")</pre>

VB Syntax	Select Scheduler <option>, <address>, <username>, <forcePasswordEntry>
VB Example	<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.

UI Access	N/A		
Parameters	Name	Type	Description
	<ProjectName>	String	The name of the project already in the Desktop that is to be activated.
Return Value	Object, the project that is activated.		

Python Syntax	SetActiveProject (<ProjectName>)
Python Example	<pre>oProject = oDesktop.SetActiveProject ("Project1")</pre>

VB Syntax	SetActiveProject <ProjectName>
VB Example	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.

UI Access	N/A		
Parameters	Name	Type	Description
	<ProjectName>	String	The full path name of the project already in the Desktop that is to be activated.
Return Value	Object, the project that is activated.		

Python Syntax	SetActiveProjectByPath(<ProjectName>)
Python Example	oProject = oDesktop.SetActiveProjectByPath("c:\Projects\MyProject.aedt")

VB Syntax	SetActiveProjectByPath <ProjectName>
VB Example	Set oProject = oDesktop.SetActiveProjectByPath "c:\Projects\MyProject.aedt"

SetCustomMenuSet

Sets the custom menu set for Electronics Desktop.

UI Access	Navigate to Tools > Options > General Options > General > Desktop Configuration . Select a configuration using the Custom Menu Set drop-down menu.		
Parameters	Name	Type	Description
	<customMenuSet>	String	Name of desired menu set. Can be one of: • 'Default' • 'EM' • 'RF' • 'RF.0' • 'SI' • 'SI1.0' • 'SI2.0' • 'Twin Builder'
Return Value	None		

Python Syntax	SetCustomMenuSet(<customMenuSet>)
Python Example	<pre>oDesktop.SetCustomMenuSet('Default')</pre>

VB Syntax	SetCustomMenuSet(<customMenuSet>)

VB Example	<code>Set oProject = oDesktop.SetCustomMenuSet "EM"</code>

SetDesktopConfiguration

Sets the desktop configuration.

UI Access	Navigate to Tools > Options > General Options > General > Desktop Configuration . Select a configuration using the Set targeted configuration drop-down menu.		
Parameters	Name	Type	Description
	<configName>	String	Name of desired Desktop configuration. Can be one of: 'All' 'EM' 'RF' 'SI' 'Twin Builder'
Return Value	None		

Python Syntax	<code>SetDesktopConfiguration (<configName>)</code>
Python Example	<code>oDesktop.SetDesktopConfiguration('RF')</code>

VB Syntax	<code>SetDesktopConfiguration(<configName>)</code>

VB Example	<code>Set oProject = oDesktop.SetDesktopConfiguration "SI"</code>

SetLibraryDirectory

Sets the library directory path. The specified directory must already exist and contain a syslib folder.

UI Access	NA		
Parameters	Name	Type	Description
	<DirectoryPath>	String	The path to the SysLib Directory
Return Value	None		

Python Syntax	<code>SetLibraryDirectory (<DirectoryPath>)</code>
Python Example	<code>oDesktop.SetLibraryDirectory("c:\libraries")</code>

VB Syntax	<code>SetLibraryDirectory <DirectoryPath></code>
VB Example	<code>oDesktop.SetLibraryDirectory"c:\libraries"</code>

SetProjectDirectory

Sets the project directory path.

UI Access	N/A		
Parameters	Name	Type	Description
	<DirectoryPath>	String	The path to the project directory. This should be writeable by the user.

Return Value	None.

Python Syntax	SetProjectDirectory (<DirectoryPath>)
Python Example	<code>oDesktop.SetProjectDirectory("c:\projects")</code>

VB Syntax	SetProjectDirectory <DirectoryPath>
VB Example	<code>oDesktop.SetProjectDirectory "c:\projects"</code>

SetRegistryFromFile

Configures registry by specifying an Analysis Configuration file which must have been exported from the HPC and Analysis panel.

UI Access	N/A		
Parameters	Name	Type	Description
	<filePath>	String	Full file path of registry file.
Return Value	Success if configuration is imported. Bad argument value if the file is not found or does not contain valid analysis configuration data.		

Python Syntax	SetRegistryFromFile(<filePath>)

Python Example	<code>oDesktop.SetRegistryFromFile('c:/temp/test.acf')</code>

VB Syntax	<code>SetRegistryFromFile <filePath></code>
VB Example	<code>oDesktop.SetRegistryFromFile "c:/temp/test.acf"</code>

SetRegistryInt

Sets registry key to an integer value.

UI Access	N/A		
Parameters	Name	Type	Description
	<KeyName>	String	Full name of registry key, including path.
	<int>	Integer	Integer value to be assigned to registry key.
Return Value	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.		

Python Syntax	<code>SetRegistryInt(<KeyName>, <int>)</code>
Python Example	<code>oDesktop.SetRegistryInt('Desktop/Settings/ProjectOptions/HFSS 3D Layout/UpdateReportsDynamicallyOnEdits', 0)</code>

VB Syn-	<code>SetRegistryInt <KeyName> <int></code>

tax	
VB Example	<code>oDesktop.SetRegistryInt "Desktop/Settings/ProjectOptions/HFSS 3D Layout/UpdateReportsDynamicallyOnEdits", 0</code>

SetRegistryString

Sets registry key to a string value.

UI Access	N/A		
Parameters	Name	Type	Description
	<KeyName>	String	Full name of registry key, including path.
	<value>	String	String value to be assigned to registry key.
Return Value	Success if the key is defined as a text string. Bad argument value if the key is not defined or requires an integer value.		

Python Syntax	<code>SetRegistryString(<KeyName>, <value>)</code>
Python Example	<code>oDesktop.SetRegistryString('Desktop/ActiveDSOConfigurations/HFSS 3D Layout', 'Local')</code>

VB Syntax	<code>SetRegistryString <KeyName>, <value></code>
VB Example	<code>oDesktop.SetRegistryString "Desktop/ActiveDSOConfigurations/HFSS 3D Layout", "Local"</code>

SetSchematicEnvironment

Sets the schematic environment for Electronics Desktop.

UI Access	Navigate to Tools > Options > General Options > General > Desktop Configuration . Select a schematic environment using the Custom Menu Set drop-down menu.		
Parameters	Name	Type	Description
	<schEnv>	Integer	Desired schematic environment. Can be one of: • 0 (Circuit) • 1 (Twin Builder) • 2 (Maxwell Circuit)
Return Value	None		

Python Syntax	SetSchematicEnvironment(<schEnv>)
Python Example	<code>oDesktop.SetSchematicEnvironment(1)</code>

VB Syntax	SetSchematicEnvironment(<schEnv>)
VB Example	<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.

UI Access	N/A		
Parameters	Name	Type	Description
	<DirectoryPath>	String	The path to the Temp directory. This should be writeable by the user.
Return Value	None.		

Python Syntax	SetTempDirectory (<DirectoryPath>)		
Python Example	<code>oDesktop.SetTempDirectory("c:\tmp")</code>		

VB Syntax	SetTempDirectory <DirectoryPath>		
VB Example	<code>oDesktop.SetTempDirectory "c:\tmp"</code>		

ShowDockingWindow

Shows or hides a docking window.

UI Access	Right click docking window > Show/Hide.		
Parameters	Name	Type	Description
	<windowName>	String	The window name (for example, "Message Manager", "Component Libraries", "Properties")
	<show>	Boolean	True to show; False to hide.
Return Value	None.		

Python Syntax	ShowDockingWindow (<windowName>, <show>)
Python Example	<code>oDesktop.ShowDockingWindow('Message Manager', False)</code>

VB Syntax	ShowDockingWindow <windowName> <show>
VB Example	<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).

UI Access	NA		
Parameters	Name	Type	Description
	<TimeInMil- liseconds>	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	oDesktop.Sleep 1000

StopSimulations

Either cleanly stops all running and pending simulations, or aborts them.

UI Access	NA		
Parameters	Name	Type	Description
	<bool> <Clean>	Integer	If true, clean stop for all running and pending simulations. If false, aborts them.
Return Value	A human readable status string specifying what happened.		

Python Syntax	StopSimulations (<bool>, clean)
Python Example	oDesktop.StopSimulations(bool clean)

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	Tools > Job Management > Submit Job.

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.
	<design (optional)>	String	Name of the design to use for batch solve.
	<setup (optional)>	String	Name of the specific setup to solve.
Return Value	Array of job ID strings (empty if no jobs submitted).		

Python Syntax	SubmitJob(<settingsPath>, <projectPath>, <design>, <setup>)
Python Example	<pre> jobIDs = oDesktop.SubmitJob("C:\\hpc-settings\\Submit_ Job_Settings.ared", "C:\\projects\\basic.aedt")) moreIDs = oDesktop.SubmitJob("C:\\hpc-settings\\Submit_ Job_Settings.ared", "C:\\projects\\basic.aedt", "Design1", "Setup1") </pre>

VB Syntax	SubmitJob <settingsPath>, <projectPath>, <design>, <setup>
VB Example	<pre> jobIDs = oDesktop.SubmitJob "C:\\hpc-settings\\Submit_ Job_Settings.ared", "C:\\projects\\basic.aedt" moreIDs = oDesktop.SubmitJob "C:\\hpc-settings\\Submit_ Job_Settings.ared", "C:\\projects\\basic.aedt", "Design1", "Setup1" </pre>

TileWindows

Arrange all open windows in a tiled format.

UI Access	From main menu, Window >Tile Horizontally or Window >Tile Vertically .		
Parameters	Name	Type	Description
	<TilingFlag>	Integer	• 0 – Tile vertically. • 1 – Tile horizontally.
Return Value	None.		

Python Syntax	TileWindows(<TilingFlag>)
Python Example	oDesktop.TileWindows(0)

VB Syntax	TileWindows <TilingFlag>
VB Example	oDesktop.TileWindows 0

Desktop Commands For Registry Values

The Ansys Registry is stored as XML format file. By default it is located at C:\User-s\<UserName>\Documents\Ansoft\<AnsysProductNameversion>\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 HFSS 3D Layout.
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/HFSS 3D Layout", "test")
```

See the following subtopics:

[DeleteRegistryEntry](#)

[DoesRegistryValueExist](#)

[GetRegistryInt](#)

[GetRegistryString](#)

[SetRegistryFromFile](#)

[SetRegistryInt](#)

[SetRegistryString](#)

DeleteRegistryEntry

Deletes a registry entry from a registry key. Returns true if deletion succeeded.

UI Access	N/A		
Parameters	Name	Type	Description

	<table><tr><td><pathToRegistry></td><td>String</td><td>Path to Registry entry.</td></tr></table>	<pathToRegistry>	String	Path to Registry entry.
<pathToRegistry>	String	Path to Registry entry.		
Return Value	Boolean: • True – Key has been deleted. • False – Key does not exist.			

Python Syntax	<code>DeleteRegistryEntry(<pathToRegistry>)</code>
Python Example	<pre>res = oDesktop.DeleteRegistryEntry("Desktop/ColorScheme")</pre>

VB Syntax	<code>DeleteRegistryEntry <pathToRegistry></code>
VB Example	<pre>res = oDesktop.DeleteRegistryEntry "Desktop/ColorScheme"</pre>

DoesRegistryValueExist

Determines whether a registry value exists.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><KeyName></code>	String	Full name of registry key, including path.
Return Value	Boolean: • True – Key exists. • False – Key does not exist.		

Python Syntax	DoesRegistryValueExist(<KeyName>)
Python Example	<pre>Exist = oDesktop.DoesRegistryValueExist('Desktop/ActiveDSOConfigurations/HFSS 3D Lay- out')</pre>

VB Syn- tax	DoesRegistryValueExist(<KeyName>)
VB Example	<pre>bExist = oDesktop.DoesRegistryValueExist("Desktop/ActiveDSOConfigurations/HFSS 3D Lay- out")</pre>

GetRegistryInt

Obtains registry key integer value.

UI Access	N/A		
Parameters	Name	Type	Description
	<KeyName>	String	Full name of registry key, including path.
Return Value	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.		

Python Syntax	GetRegistryInt(<KeyName>)
Python	<pre>num = oDesktop.GetRegistryInt('Desktop/Settings/ProjectOptions/HFSS 3D Layout/Up-</pre>

Examp- e	<code>dateReportsDynamicallyOnEdits')</code>

VB Syntax	<code>GetRegistryInt(<KeyName>)</code>
VB Examp- e	<code>num = oDesktop.GetRegistryInt("Desktop/Settings/ProjectOptions/HFSS 3D Layout/Up- dateReportsDynamicallyOnEdits")</code>

GetRegistryString

Obtains registry key string value.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><KeyName></code>	String	Full name of registry key, including path.
Return Value	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.		

Python Syntax	<code>GetRegistryString(<KeyName>)</code>
Python Example	<code>activeDSO = oDesktop.GetRegistryString('Desktop/ActiveDSOConfigurations/HFSS 3D Layout')</code>

VB Syntax	GetRegistryString(<KeyName>)
VB Example	activeDSO = oDesktop.GetRegistryString("Desktop/ActiveDSOConfigurations/HFSS 3D Layout")

SetRegistryFromFile

Configures registry by specifying an Analysis Configuration file which must have been exported from the HPC and Analysis panel.

UI Access	N/A		
Parameters	Name	Type	Description
	<filePath>	String	Full file path of registry file.
Return Value	Success if configuration is imported. Bad argument value if the file is not found or does not contain valid analysis configuration data.		

Python Syntax	SetRegistryFromFile(<filePath>)
Python Example	oDesktop.SetRegistryFromFile('c:/temp/test.acf')

VB Syntax	SetRegistryFromFile <filePath>
VB Example	oDesktop.SetRegistryFromFile "c:/temp/test.acf"

SetRegistryInt

Sets registry key to an integer value.

UI Access	N/A		
Parameters	Name	Type	Description
	<KeyName>	String	Full name of registry key, including path.
	<int>	Integer	Integer value to be assigned to registry key.
Return Value	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.		

Python Syntax	SetRegistryInt(<KeyName>, <int>)
Python Example	<pre>oDesktop.SetRegistryInt('Desktop/Settings/ProjectOptions/HFSS 3D Layout/UpdateReportsDynamicallyOnEdits', 0)</pre>

VB Syntax	SetRegistryInt <KeyName> <int>
VB Example	<pre>oDesktop.SetRegistryInt "Desktop/Settings/ProjectOptions/HFSS 3D Layout/UpdateReportsDynamicallyOnEdits", 0</pre>

SetRegistryString

Sets registry key to a string value.

UI Access	N/A

Parameters	Name	Type	Description
	<KeyName>	String	Full name of registry key, including path.
	<value>	String	String value to be assigned to registry key.
Return Value	Success if the key is defined as a text string. Bad argument value if the key is not defined or requires an integer value.		

Python Syntax	SetRegistryString(<KeyName>, <value>)
Python Example	<code>oDesktop.SetRegistryString('Desktop/ActiveDSOConfigurations/HFSS 3D Layout', 'Local')</code>

VB Syntax	SetRegistryString <KeyName>, <value>
VB Example	<code>oDesktop.SetRegistryString "Desktop/ActiveDSOConfigurations/HFSS 3D Layout", "Local"</code>

ImportExport Tool Commands

These oDesktop commands are run by using the GetTool script to call the ImportExport Tool.

```
oTool = oDesktop.GetTool("ImportExport")
```

Scripts run via the ImportExport Tool include:

[ImportANF](#)

[ImportANFV2](#)

[ImportAutoCAD](#)

[ImportAWRMicrowaveOffice](#)

[ImportEDB](#)

[ImportExtracta](#)

[ImportGDSII](#)

[ImportGerber](#)

[ImportIDF](#)

[ImportIDFandMerge](#)

[ImportIPC](#)

[ImportODB](#)

[ImportXFL](#)

ImportANF

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

UI Access	File > Import > ANF.		
Parameters	Name	Type	Description
	<ANFfilename>	String	Full path of ANF file.
Return Value	None.		

Python Syntax	ImportANF(<ANFfilename>)
Python Example	<pre>oDesktop.RestoreWindow() Set oTool = oDesktop.GetTool('ImportExport')</pre>

	<code>oTool.ImportANF('C:\\AnTranslator\\results\\package4.anf')</code>

VB Syntax	ImportANF <ANFfilename>
VB Example	<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") oTool.ImportANF("%UserProfile%\Documents\HFSS Examples\package.anf") </pre>

ImportANFv2

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

UI Access	File > Import > ANF.

Parameters	Name	Type	Description
	<ANFfilename>	String	Full path of ANF file.
	<outputPathName>	String	Full path of *.aedb output file.
	<controlFileName>	String	Full path of XML control file.
	<cmpFileName>	String	Full path of CMP file. Pass empty string if none.
Return Value	None.		

Python Syntax	ImportANFv2 (<ANFfilename>, <outputPathName>, <controlFileName>, <cmpFileName>)
Python Example	<pre>oDesktop.RestoreWindow() Set oTool = oDesktop.GetTool('ImportExport') oTool.ImportANFV2("C:/Users/jdoe/Documents/SAMPLEFILES/my_model.anf", "C:/Users/jdoe/Documents/Ansoft/my_model.aedb", "C:/Users/jdoe/Documents/Ansoft/my_model.xml", "")</pre>

VB Syntax	ImportANFv2 <ANFfilename>, <outputPathName>, <controlFileName>, <cmpFileName>
VB Example	<pre>Dim oAnsoftApp Dim oDesktop Dim oProject Dim oDesign</pre>

```

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")
oTool.ImportANFV2 "C:/Users/jdoe/Documents/SAMPLEFILES/my_model.anf", _
"C:/Users/jdoe/Documents/Ansoft/my_model.aedb", _
"C:/Users/jdoe/Documents/Ansoft/my_model.xml", ""

```

ImportAutoCAD

Imports an AutoCAD file into a new project.

UI Access	File > Import > AutoCAD.		
Parameters	Name	Type	Description
	<dxfFileName>	String	Full path of DXF file.
	<outputPathFileName>	String	Full path of EDB file to create during import.
	<controlFileName>	String	Full path of XML control file. Pass empty string if none.
Return Value	None.		

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

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

ImportAWRMicrowaveOffice

Imports an AWRMicrowaveOffice file into a new project.

UI Access	File > Import > AWRMicrowaveOffice.		
Parameters	Name	Type	Description
	<code><xmlFileName></code>	String	Full path of XML file.
	<code><outputPathName></code>	String	Full path of output file.
	<code><logFileName></code>	String	Full path of log file.
Return Value	None.		

Python Syntax	<code>ImportAWRMicrowaveOffice (<xmlFileName>, <outputPathName>, <logFileName>)</code>

Python Example	<pre>oDesktop.RestoreWindow() Set oTool = oDesktop.GetTool('ImportExport') oTool.ImportAWRMicrowaveOffice('C:/MyFiles/package4.xml', 'C:/MyFiles/package4.aedb', 'C:/MyFiles/package4.log')</pre>

VB Syntax	ImportAWRMicrowaveOffice <xmlFileName>, <outputPathName>, <logFileName>
VB Example	<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>


```
oTool.ImportAWRMicrowaveOffice "C:/MyFiles/package4.xml", "C:/MyFiles/package4.aedb",
-
"C:/MyFiles/package4.log"
```

ImportEDB

Imports an EDB file into a new project.

UI Access	File > Import > EDB.		
Parameters	Name	Type	Description
	<edbFileName>	String	Full path of EDB file.
Return Value	None.		

Python Syntax	ImportEDB (<edbFileName>)
Python Example	<pre>oDesktop.RestoreWindow() Set oTool = oDesktop.GetTool('ImportExport') oTool.ImportEDB('C:/MyFiles/projectforimport.aedb/edb.def')</pre>

VB Syntax	ImportEDB <edbFileName>
VB Example	<pre>Set oTool = oDesktop.GetTool("ImportExport") oTool.ImportEDB "C:/MyFiles/projectforimport.aedb/edb.def"</pre>

ImportExtracta

Imports a Cadence Extracta file into a new project.

Note:

In order for this script to work, you must have the Cadence-supplied executable Extracta.exe installed on your machine.

UI Access	File > Import > Cadence APD / Allegro / SiP.		
Parameters	Name	Type	Description
	<extractaFileName>	String	Full path of Extracta file.
	<outputPathName>	String	Full path of output file to be created.
	<controlFileName>	String	Optional. Full path to XML control file.
Return Value	None.		

Python Syntax	ImportExtracta (<extractaFileName>, <outputPathName>, <controlFileName>)
Python Example	<pre>oDesktop.RestoreWindow() Set oTool = oDesktop.GetTool('ImportExport') oTool.ImportExtracta('C:/MyFiles/projectforimport.brd', 'C:/MyFiles/project.edb', '')</pre>

VB Syntax	ImportExtracta <extractaFileName>, <outputPathName>, <controlFileName>
VB	Set oTool = oDesktop.GetTool("ImportExport")

Example	<code>oTool.ImportExtracta "C:/MyFiles/projectforimport.brd", "C:/MyFiles/project.edb", ""</code>

ImportGDSII

Imports a GDSII file into a new project.

UI Access	File > Import > GDSII.		
Parameters	Name	Type	Description
	<code><gdsiiFileName></code>	String	Full path of GDSII file.
	<code><outputPathName></code>	String	Full path of EDB file to create during import.
	<code><controlFileName></code>	String	Optional. Full path of XML control file. Full path of “technology” (corresponding to <code>-t=technologyfile</code> argument in anstranslator). Pass empty string if none.
	<code><mapFileName></code>	String	If technology file was used in place of the control file. Full path to either gdsii mapping file (corresponding to <code>-g=gdsMappingFile</code> argument in anstranslator) or XML control file. Otherwise, pass empty string.
Return Value	None.		

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

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

ImportGerber

Imports a GERBER file into a new project.

UI Access	File > Import > GERBER.		
Parameters	Name	Type	Description
	<gerberFileName>	String	Full path of GERBER 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.
Return Value	None.		

Python Syntax	<code>ImportGerber(<gerberFileName>, <outputPathName>, <controlFileName>)</code>
Python Example	<pre>oDesktop.RestoreWindow() Set oTool = oDesktop.GetTool('ImportExport') oTool.ImportGERBER('C:/Files/test.gbr', 'C:/Files/test.aedb.edb', 'C:/Files/test.xml')</pre>

VB Syntax	ImportGerber <gerberFileName>, <outputPathName>, <controlFileName>
VB Example	<pre>Set oTool = oDesktop.GetTool("ImportExport") oTool.ImportGERBER "C:/Files/test.gbr", "C:/Files/test.aedb.edb", _ "C:/Files/test.xml"</pre>

ImportIDF

Imports an IDF file into a new project. To merge with an existing design, use [ImportIDFandMerge](#).

UI Access	placeholder		
Parameters	Name	Type	Description
	<NAME>	string	Settings
	<Board>	bool	Full path of board file
	<Library>	int	Full path of library file
	<Control>	int	Full path of control file
	<Filters>	list	["Cap", "Height", "HeightExclude2D", "Ind", "Power", "Res"]
	<CreateFilteredAsNonModel>	bool	True or False
	<FootPrint>	string	Footprint size and unit
	<NAME>	string	definitionOverridesMap
	<NAME>	string	instanceOverridesMap
	<HighSurfThickness>	string	High side surface thickness and unit
	<LowSurfThickness>	string	Low side surface thickness and unit
	<InternalLayerThickness>	string	Internal layer thickness and unit
	<NumInternalLayer>	int	Number of internal layers
	<HighSurfaceCopper>	int	High side surface coverage percentage
	<LowSurfaceCopper>	int	Low side surface coverage percentage

	<InternalLayerCopper>	int	Internal layer coverage percentage
	<TraceMaterial>	string	Trace material name
	<SubstrateMaterial>	int	Substrate material name
	<CreateBoard>	bool	True or False
	<ModelBoardAsRect>	bool	True or False
	<ModelDeviceAsRect>	bool	True or False
	<Cutoff>	bool	True or False
	<IncludeDrilledHoles>	bool	True or False
	<HoleDiameterCutoff>	string	Hole diameter size and unit
	<ReplaceDevices>	bool	True or False
	<CreatePointsOnBoardForInstances>	bool	True or False
Return Value	None.		

Python Syntax	ImportIDF_1 (<NAME>, <Board>, <Library>, <Control>, <Filters>, <CreateFilteredAsNonModel>, <NAME>, <NAME>, <HighSurfThickness>, <LowSurfThickness>, <InternalLayerThickness>, <NumInternalLayer>, <HighSurfaceCopper>, <LowSurfaceCopper>, <InternalLayerCopper>, <TraceMaterial>, <SubstrateMaterial>, <CreateBoard>, <ModelBoardAsRect>, <ModelDeviceAsRect>, <Cutoff>, <IncludeDrilledHoles>, <ReplaceDevices>)
Python Example	<pre>oDesign.ImportIDF(["NAME:Settings", "Board:=", "Library:=", "Control:=", "Filters:=", "CreateFilteredAsNonModel:=", "FootPrint:="], "C:\\Users\\Model Files\\brd_board.emn", "C:\\Users\\Model Files\\brd_board.emp", "C:\\Users\\Model Files\\brd_board.xml", ["HeightExclude2D"], False, "0.1mm2",)</pre>

	<pre> ["NAME:definitionOverridesMap"], ["NAME:instanceOverridesMap"], "HighSurfThickness:=" , "0.07mm", "LowSurfThickness:=" , "0.07mm", "InternalLayerThickness:=" , "0.07mm", "NumInternalLayer:=" , 2, "HighSurfaceCopper:=" , 30, "LowSurfaceCopper:=" , 30, "InternalLayerCopper:=" , 30, "TraceMaterial:=" , "Cu-Pure", "SubstrateMaterial:=" , "FR-4", "CreateBoard:=" , True, "ModelBoardAsRect:=" , True, "ModelDeviceAsRect:=" , False, "Cutoff:=" , False, "IncludeDrilledHoles:=" , True, "HoleDiameterCutoff:=" , "0.1mm", "ReplaceDevices:=" , False, "CreatePointsOnBoardForInstances:=" , False]) </pre>

VB Syntax	ImportIDF <idfFileName>, <outputPathName>, <controlFileName>, <libFileName>
VB	oDesign.ImportIDF Array("NAME:Settings", "Board:=", _"C:\Users\Model Files" "\brd_

Example	<pre>board.emn", "Library:=", _"C:\Users\Model Files" "\brd_board.emp", "Control:=", _"C:\Users\Model Files" "\brd_board.xml", "Filters:=", Array("HeightExclude2D"), "CreateFilteredAsNonModel:=", _false, Array("NAME:definitionOverridesMap"), Array("NAME:instanceOverridesMap"), "HighSurfThickness:=", _ "0.07mm", "LowSurfThickness:=", "0.07mm", "InternalLayerThickness:=", "0.07mm", "NumInternalLayer:=", _2, "HighSurfaceCopper:=", 30, "LowSurfaceCopper:=", 30, "InternalLayerCopper:=", _30, "TraceMaterial:=", "Cu-Pure", "SubstrateMaterial:=", "FR-4", "CreateBoard:=", _true, "ModelBoardAsRect:=", true, "ModelDeviceAsRect:=", false, "Cutoff:=", _false, "IncludeDrilledHoles:=", true, "ReplaceDevices:=", false)</pre>

ImportIDFandMerge

Imports an IDF file into a new project. To import without merging, use [ImportIDF](#).

UI Access	File > Import > IDF. Enable Merge with existing and select a project/design.		
Parameters	Name	Type	Description
	<idfFileName>	String	Full path of GERBER file.
	<libFileName>	String	Optional. Full path of library file. Pass empty string if none.
	<controlFileName>	String	Optional. Full path of XML control file. Pass empty string if none.
	<project>	String	Name of project to merge with.
	<design>	String	Name of design to merge with.

Return Value	None.

Python Syntax	<code>ImportIDFandMerge(<idfFileName>, <libPathName>, <controlFileName>, <project>, <design>)</code>
Python Example	<pre>oDesktop.RestoreWindow() Set oTool = oDesktop.GetTool('ImportExport') oTool.ImportIDFandMerge('C:/Files/test.brd', 'C:/Files/test.aedb.lib', 'C:/Files/test.xml', 'Project1', 'Design12')</pre>

VB Syntax	<code>ImportIDFandMerge<idfFileName>, <libPathName>, <controlFileName>, <project>, <design></code>
VB Example	<pre>Set oTool = oDesktop.GetTool("ImportExport") oTool.ImportIDFandMerge "C:/Files/test.brd", "C:/Files/test.lib", _ "C:/Files/test.xml", "Project1", "Design12"</pre>

ImportIDX

Imports and IDX model into a new Icepak project.

UI Access	Icepak > Import IDX		
Parameters	Name	Type	Description
	<NAME>	string	Settings
	<Board>	bool	Full path of .idx file

	<Library>	string	NA
	<Control>	string	NA
	<Filters>	list	HeightExclude2D
	<CreateFilteredAsNonModel>	bool	True or False
	<NAME>	string	definitionOverridesMap
	<NAME>	string	instanceOverridesMap
	<HighSurfThickness>	string	High surface layer thickness value and unit
	<LowSurfThickness>	string	Low surface layer thickness value and unit
	<InternalLayerThickness>	string	Internal surface layer thickness value and unit
	<NumInternalLayer>	int	Number of internal layers value
	<HighSurfaceCopper>	int	High surface percent coverage value
	<LowSurfaceCopper>	int	Low surface percent coverage value
	<InternalLayerCopper>	int	Internal layer percent coverage value
	<TraceMaterial>	string	Trace material name
	<SubstrateMaterial>	int	Substrate material name
	<CreateBoard>	bool	True or False
	<ModelBoardAsRect>	bool	True or False
	<ModelDeviceAsRect>	bool	True or False
	<Cutoff>	bool	True or False
	<ReplaceDevices>	bool	True or False
Return Value	None		

Python Syntax	ImportIDX (<NAME>, <Board>, <Library>, <Control>, <Filters>, <CreateFilteredAsNonModel>, <NAME>, <NAME>, <HighSurfThickness>, <LowSurfThickness>, <InternalLayerThickness>, <NumInternalLayer>, <HighSurfaceCopper>, <LowSurfaceCopper>, <InternalLayerCopper>, <TraceMaterial>, <SubstrateMaterial>, <CreateBoard>, <ModelBoardAsRect>, <ModelDeviceAsRect>, <Cutoff>, <ReplaceDevices>)
Python Example	oDesign.ImportIDX(

	<pre> ["NAME:Settings", "Board:=" , "C:\\Users\\Model files\\PCB-00278_A.idx", "Library:=" , "", "Control:=" , "", "Filters:=" , ["HeightExclude2D"], "CreateFilteredAsNonModel:=", False, ["NAME:definitionOverridesMap"], ["NAME:instanceOverridesMap"], "HighSurfThickness:=" , "0.07mm", "LowSurfThickness:=" , "0.07mm", "InternalLayerThickness:=", "0.07mm", "NumInternalLayer:=" , 2, "HighSurfaceCopper:=" , 30, "LowSurfaceCopper:=" , 30, "InternalLayerCopper:=" , 30, "TraceMaterial:=" , "Cu-Pure", "SubstrateMaterial:=" , "FR-4", "CreateBoard:=" , True, "ModelBoardAsRect:=" , False, "ModelDeviceAsRect:=" , False, "Cutoff:=" , False, "ReplaceDevices:=" , False]) </pre>

VB Syn-	ImportIDX (<NAME>, <Board>, <Library>, <Control>, <Filters>, <CreateFilteredAsNonModel>, <NAME>, <NAME>,

tax	<code><HighSurfThickness>, <LowSurfThickness>, <InternalLayerThickness>, <NumInternalLayer>, <HighSurfaceCopper>, <LowSurfaceCopper>, <InternalLayerCopper>, <TraceMaterial>, <SubstrateMaterial>, <CreateBoard>, <ModelBoardAsRect>, <ModelDeviceAsRect>, <Cutoff>, <ReplaceDevices>)</code>
VB Example	<pre>oDesign.ImportIDX Array("NAME:Settings", "Board:=", _"C:\Users\Model files" _ "\PCB-00278_A.idx", "Library:=", "", "Control:=", "", "Filters:=", Array(_"HeightExclude2D"), "CreateFilteredAsNonModel:=", false, Array("NAME:definitionOverridesMap"), Array("NAME:instanceOverridesMap"), "HighSurfThickness:=", _0.07mm, "LowSurfThickness:=", "0.07mm", "InternalLayerThickness:=", "0.07mm", "NumInternalLayer:=", _2, "HighSur- faceCopper:=", 30, "LowSurfaceCopper:=", 30, "InternalLayerCopper:=", _30, "TraceMaterial:=", "Cu-Pure", "SubstrateMaterial:=", "FR-4", "CreateBoard:=", _true, "ModelBoardAsRect:=", false, "ModelDeviceAsRect:=", false, "Cutoff:=", _false, "ReplaceDevices:=", false)</pre>

ImportIPC

Imports an IPC2581 file into a new project.

UI Access	File > Import > IPC2581.		
Parameters	Name	Type	Description
	<code><ipcFileName></code>	String	Full path of IPC2581 file.
	<code><outputPathName></code>	String	Full path of EDB file to create during import.
	<code><controlFileName></code>	String	Optional. Full path of XML control file. Pass empty string if none.
Return Value	None.		

Python Syntax	<code>ImportIPC(<ipcFileName>, <outputPathName>, <controlFileName>)</code>
Python Example	<pre>oDesktop.RestoreWindow() Set oTool = oDesktop.GetTool('ImportExport') oTool.ImportIPC('C:/Files/test.cvg', 'C:/Files/test.aedb', 'C:/Files/test.xml', 'C:/Files/test.txt')</pre>

VB Syntax	<code>ImportIPC <ipcFileName>, <outputPathName>, <controlFileName></code>
VB Example	<pre>Set oTool = oDesktop.GetTool("ImportExport") oTool.ImportIPC "C:/Files/test.cvg", "C:/Files/test.aedb", _ "C:/Files/test.xml"</pre>

ImportODB

Imports an ODB++ file into a new project.

UI Access	File > Import > ODB++.		
Parameters	Name	Type	Description
	<code><odbFileName></code>	String	Full path of ODB++ file.
	<code><outputPathName></code>	String	Full path of EDB file to create during import.
	<code><controlFileName></code>	String	Optional. Full path of XML control file. Pass empty string if none.
Return Value	None.		

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

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

ImportXFL

Imports an XFL file into a new project.

UI Access	File > Import > XFL.		
Parameters	Name	Type	Description
	<code><xflFileName></code>	String	Full path of XFL file.
	<code><outputPathName></code>	String	Full path of EDB file to create during import.
	<code><controlFileName></code>	String	Optional. Full path of XML control file. Pass empty string if none.
Return Value	None.		

Python Syntax	<code>ImportXFL(<xflFileName>, <outputPathName>, <controlFileName>)</code>
Python Example	<pre>oDesktop.RestoreWindow() Set oTool = oDesktop.GetTool('ImportExport') oTool.ImportXFL('C:/Files/test.xfl', 'C:/Files/test.aedb', 'C:/Files/test.xml')</pre>

VB Syntax	<code>ImportXFL <xflFileName>, <outputPathName>, <controlFileName></code>
VB Example	<pre>Set oTool = oDesktop.GetTool("ImportExport") oTool.ImportXFL "C:/Files/test.xfl", "C:/Files/test.aedb", _ "C:/Files/test.xml"</pre>

5 - 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.

UI Access	N/A
Parameters	None.
Return Value	Array containing list of Ansys Electronics Desktop instances.

Python Syntax	GetAllRunningInstances()
Python Example	<code>obj = oRunningInstances.GetAllRunningInstances()</code>

VB Syntax	GetAllRunningInstances
VB Example	<code>set obj = oRunningInstances.GetAllRunningInstances()</code>

GetRunningInstanceByProcessID

Returns the instance of Ansys Electronics Desktop that is running a specified process.

UI Access	N/A		
Parameters	Name	Type	Description
	<processID>	Integer	Process ID
Return Value	String of the returned instance.		

Python Syntax	GetRunningInstanceByProcessID(<processID>)
Python Example	<code>obj = oRunningInstances.GetRunningInstanceByProcessID(12345)</code>

VB Syntax	GetRunningInstanceByProcessID <processID>
VB Example	<code>set obj = oRunningInstances.GetRunningInstanceByProcessID(12345)</code>

GetRunningInstancesMgr

Returns the object of the Running Instances Manager.

UI Access	N/A		
Parameters	Name	Type	Description
	None		

Return Value	Object Running instances manager object

Python Syntax	<code>GetRunningInstancesMgr()</code>
Python Example	<code>oRunningInstances = oDesktop.GetRunningInstancesMgr()</code>

VB Syntax	<code>GetRunningInstancesMgr()</code>
VB Example	<code>Set oRunningInstances = oDesktop.GetRunningInstancesMgr()</code>

This page intentionally
left blank.

6 - Project Object Script Commands

Project commands should be executed by the oProject object.

One example of accessing this object is:

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

Dataset Script Commands:

[AddDataset](#)

[DeleteDataset](#)

[EditDataset](#)

[ExportDataset](#)

[HasDataset](#)

[ImportDataset](#)

Other Project Object Script Commands:

[AddMaterial](#)

[AnalyzeAll](#)

[ChangeProperty](#)

[ClearMessages](#)

[CloneMaterial](#)

[Close](#)

[CopyDesign](#)

[CutDesign](#)

[DeleteDesign](#)

[DeleteToolObject](#)

[EditMaterial](#)

[ExportMaterial](#)

[GetActiveDesign](#)

[GetArrayVariables](#)

[GetChildNames \[Project\]](#)

[GetChildObject \[Project\]](#)

[GetChildTypes \[Project\]](#)

[GetConfigurableData](#)

[GetDefinitionManager](#)

[GetDependentFiles](#)

[GetDesign](#)

[GetDesigns](#)

[GetEDBHandle](#)

[GetLegacyName](#)

[GetName \[Project\]](#)

[GetObjPath \[Project\]](#)

[GetPath](#)

[GetPropEvaluatedValue](#)

[GetPropNames \[Project\]](#)

[GetPropSIValue](#)

[GetPropValue \[Project\]](#)

[GetProperties](#)

[GetPropertyValue](#)

[GetTopDesignList](#)

[GetVariableValue](#)

[GetVariables](#)

[InsertDesign](#)

[InsertDesign \(PlanarEM\)](#)

[InsertDesignWithWorkflow](#)

[InsertToolObject](#)

[Paste \[Project Object\]](#)

[Redo \[Project Level\]](#)

[RemoveAllUnusedDefinitions](#)

[RemoveMaterial](#)

[RemoveUnusedDefinitions](#)

[Rename](#)

[RunToolkit](#)

[Save](#)

[SaveAs](#)

[SaveAsStandAloneProject](#)[SaveProjectArchive](#)[SetActiveDefinitionEditor](#)[SetActiveDesign](#)[SetPropValue \[Project\]](#)[SetPropertyValue](#)[SetVariableValue](#)[SimulateAll](#)[Undo \[Project\]](#)[UpdateDefinitions](#)

AddDataset

Adds a dataset. This can be executed by the oProject, or oDesign variables.

UI Access	Project > Datasets > Add.		
Parameters	Name	Type	Description
	< <i>DatasetDataArray</i> >	Array	Array("NAME:<DatasetName>", Array("NAME:Coordinates", <CoordinateArray>, <CoordinateArray>, ...)
	< <i>DatasetName</i> >	String	Name of the dataset.
	< <i>CoordinateArray</i> >	Array	Array("NAME:Coordinate", "X:=", <double>, "Y:=", <double>)
Return Value	None.		

Python Syntax	AddDataset <DatasetDataArray>
Python Example	<pre>oProject.AddDataset(["NAME:\$ds1", ["NAME:Coordinates", ["NAME:Coordinate", "X:=", 2, "Y:=", 4], ["NAME:Coordinate", "X:=", 6, "Y:=", 8]]]) oDesign.AddDataset(</pre>


```
[  
  "NAME:$ds1",  
  [  
    "NAME:Coordinates",  
    [  
      "NAME:Coordinate",  
      "X:=", 2,  
      "Y:=", 4  
    ],  
    [  
      "NAME:Coordinate",  
      "X:=", 6,  
      "Y:=", 8  
    ]  
  ]  
]  
)
```

VB Syntax**AddDataset <DatasetDataArray>**

VB Example	<pre>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)))</pre>

AddMaterial

Adds a local material.

UI Access	Add Material in the material editor.		
Parameters	Name	Type	Description
	<MaterialParams>	Array	["NAME: <name of the material to be added>", <MatProperty>, <MatProperty>, ...]

	<MatProperty>	Array	For simple material: "<PropertyName>:=", <value> For anisotropic material: ["NAME:<PropertyName>", "property_type:=", "AnisoProperty", "unit:=", <Unit>", "component1:=", <value>, "component2:=", <value>, "component3:=", <value>))]
	<PropertyName>	String	Should be one of the following (depending on the material, design, and solution types): Electromagnetic (Maxwell-exclusive material properties omitted, see Maxwell Scripting help): "permittivity", "permeability", "conductivity", "dielectric_loss_tangent", "magnetic_loss_tangent", "electric_coercivity", "magnetic_coercivity", "saturation_mag", "lande_g_factor", "delta_H", "delta_h_freq", "mass_density"

		Thermal (including solids, Icepak fluid flow, and Mechanical rotating fluid modeling): "thermal_conductivity", "mass_density", "specific_heat", "thermal_expansion_coefficient", "thermal_material_type", "viscosity", "diffusivity", "molecular_mass", "clarity_type" Structural: "mass_density", "youngs_modulus", "poissons_ratio", "thermal_expansion_coefficient"
<Unit>	String	Possible values (Maxwell-exclusive properties omitted, see Maxwell Scripting Help; other missing entries are unitless): conductivity: "siemens/m" saturation_mag: "uTesla", "mTesla", "tesla", "kTesla", "uGauss", "mGauss", "gauss", "kGauss" delta_H: "A_per_meter", "kA_per_meter", "Oe", "kOe" delta_h_frequency: "Hz", "kHz", "MHz", "GHz", "THz", "rps", "per_sec" mass_density: "kg/m^3" thermal_conductivity: "W/m-C" specific_heat: "J/kg-C" youngs_modulus: "N/m^2" thermal_expansion_coefficient: "1/C"

Return Value	None

Python Syntax	AddMaterial ("NAME:<MaterialName>", <MatProperty>, <MatProperty>, ...)]
Python Example	<pre>oDefinitionManager.AddMaterial(["permittivity:=", "2.2", "0.002"]) oDefinitionManager.AddMaterial [("NAME:Material2",_ "dielectric_loss_tangent:=", "44", Array("NAME:saturation_mag",_ "property_type:=", "AnisoProperty",_ "unit:=", "Gauss",_ "component1:=", "11",_ "component2:=", "22",_ "component3:=", "33"),_ "delta_H:=", "440e")]</pre>

VB Syntax	AddMaterial Array("NAME:<MaterialName>", <MatProperty>, <MatProperty>, ...)

VB Example	<pre> oDefinitionManager.AddMaterial Array("permittivity:=", "2.2", "0.002") oDefinitionManager.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:=", "440e") </pre>

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.

UI Access	Project > Analyze All.
Parameters	None.
Return Value	None.

Python Syntax	AnalyzeAll()
Python Example	<code>oProject.AnalyzeAll()</code>

VB Syntax	AnalyzeAll
VB Example	<code>oProject.AnalyzeAll</code>

ChangeProperty

Changes the properties of an object in the history tree.

UI Access	Right-click an object in the History Tree and select Properties .		
Parameters	Name	Type	Description
	<code><propertyArgs></code>	Array	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 Automation tab.
Return Value	None.		

Python Syntax	ChangeProperty(<code><propertyArgs></code>)
Python Example	Example: Changing the Position of a Box <code>oEditor.ChangeProperty(</code>

```
[
  "NAME:AllTabs",
  [
    "NAME:Geometry3DCmdTab",
    [
      "NAME:PropServers" ,
      "Box1:CreateBox:1"
    ],
    [
      "NAME:ChangedProps",
      [
        "NAME:Position",
        "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\""\n    ],\n    [\n        "NAME:Display Wireframe",\n        "Value:="          , True\n    ]\n]\n]\n])\nExample: Change Default Handle (reference coordinate system)\noEditor.ChangeProperty(\n    [\n        "NAME:AllTabs",\n        [\n            "NAME:General",\n            [\n                "NAME:PropServers",\n                "HFSSDesign1_1"\n            ],\n            [\n                "NAME:ChangedProps",\n                [\n                    "NAME:Handle",\n                    "Value:="          , " NotchOutY"\n                ]\n            ]\n        ]\n    ]\n])
```

VB Syntax	ChangeProperty <propertyArgs>
VB Example	Example: Changing the Position of a Box <pre> oEditor.ChangeProperty Array("NAME:AllTabs", Array("NAME:Geometry3DCmdTab", Array("NAME:PropServers", "Box1: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 <pre> oEditor.ChangeProperty(Array("NAME:AllTabs", Array("NAME:Geometry3DAttributeTab", Array("NAME:PropServers", "Box1"), Array("NAME:ChangedProps", Array("NAME:Material", "Value:=", "\"vacuum\""), Array("NAME:Display Wireframe", </pre>

```
        "Value:=", True
    )
)
)
)
```

Example: Change Default Handle (reference coordinate system)

```
oEditor.ChangeProperty
Array("NAME:AllTabs",
    Array("NAME:General",
        Array("NAME:PropServers",
            "HFSSDesign1_1"
        ),
        Array("NAME:ChangedProps",
            Array("NAME:Handle",
                "Value:=", " NotchOutY"
            )
        )
    )
)
```

ClearMessages

Clears information in the **Messages** window.

Note:

Additional options are available when using the Desktop-level [ClearMessages](#) command.

UI Access	N/A
Parameters	None.
Return Value	None.

Python Syntax	ClearMessages()
Python Example	<code>oProject.ClearMessages()</code>

VB Syntax	ClearMessages
VB Example	<code>oProject.ClearMessages</code>

CloneMaterial

Clones a local material.

UI Access	N/A		
Parameters	Name	Type	Description
	<matName>	String	Name of existing material.
	<newName>	String	Name for newly cloned material.
Return Value	Boolean:		

	• 1 - Material is cloned. • 0 - Existing material not found or a conflict with the new material name.

Python Syntax	CloneMaterial (<matName>, <newName>)
Python Example	oDefinitionManager.CloneMaterial ("copper1", "copper3")

VB Syntax	CloneMaterial <matName>, <newName>
VB Example	oDefinitionManager.CloneMaterial "copper1", "copper3"

Close

Closes the active project.

Warning:

Unsaved changes will be lost.

UI Access	N/A
Parameters	None.
Return Value	None.

Python Syntax	Close()
Python Example	<code>oProject.Close()</code>

VB Syntax	Close
VB Example	<code>oProject.Close</code>

CopyDesign

Copies a specified design.

UI Access	Edit > Copy.		
Parameters	Name	Type	Description
	<DesignName>	String	Name of the design to copy from.
Return Value	None.		

Python Syntax	CopyDesign (<DesignName>)
Python Example	<code>oProject.CopyDesign ("HFSSDesign1")</code>

VB Syntax	CopyDesign <DesignName>
VB Example	<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.

UI Access	Edit > Cut.		
Parameters	Name	Type	Description
	<DesignName>	String	Name of the design.
Return Value	None.		

Python Syntax	CutDesign (<DesignName>)
Python Example	<code>oProject.CutDesign("SimplorerDesign1")</code>

VB Syntax	CutDesign <DesignName>
VB Example	<code>oProject.CutDesign "SimplorerDesign1"</code>

DeleteDataset

Deletes a specified dataset. This can be executed by the oProject, or oDesign variables.

UI Access	Project > Datasets > Remove.		
Parameters	Name	Type	Description
	<DatasetName>	String	Name of the dataset found in the project.
Return Value	None.		

Python Syntax	DeleteDataset (<DatasetName>)		
Python Example	oProject.DeleteDataset ('\$ds1')		
	oDesign.DeleteDataset ('\$ds1')		

VB Syntax	DeleteDataset <DatasetName>		
VB Example	oProject.DeleteDataset "\$ds1"		
	oDesign.DeleteDataset "\$ds1"		

DeleteDesign

Deletes a specified design in the project.

UI Access	Edit > Delete , or Delete in the ribbon.		
Parameters	Name	Type	Description
	<DesignName>	String	Name of the design.

Return Value	None.

Python Syntax	DeleteDesign (<DesignName>)
Python Example	<code>oProject.DeleteDesign("HFSS 3D LayoutDesign2")</code>

VB Syntax	DeleteDesign <DesignName>
VB Example	<code>oProject.DeleteDesign "HFSS 3D LayoutDesign2"</code>

DeleteToolObject

Note:

This command is for internal Ansys use only.

UI Access	N/A		
Parameters	Name	Type	Description
	<ObjectName>	String	Name of the tool object.
Return Value	None.		

Python Syntax	DeleteToolObject(<ObjectName>)

Python Example	<code>oProject.DeleteToolObject ("object1")</code>

VB Syntax	<code>DeleteToolObject <ObjectName></code>
VB Example	<code>oProject.DeleteToolObject "object1"</code>

EditDataset

Modifies a dataset. This can be executed by the oProject, or oDesign variables.

UI Access	Project > Datasets > Edit.		
Parameters	Name	Type	Description
	<OriginalName>	String	Name of the original dataset.
	<DatasetDataArray>	Array	Data for the modified dataset.
Return Value	None.		

Python Syntax	<code>EditDataset (<OriginalName> <DatasetDataArray>)</code>
Python Example	<pre>oProject.EditDataset ("ds1" ["NAME:ds2", ["NAME:Coordinates", ["NAME:Coordinate", "X:=", 1, "Y:=", 2</pre>

```
    ],  
    [  
        "NAME:Coordinate",  
        "X:=", 3, "Y:=", 4  
    ]  
]  
]  
)  
oDesign.EditDataset ("ds1"  
["NAME:ds2",  
    ["NAME:Coordinates",  
        [  
            "NAME:Coordinate",  
            "X:=", 1, "Y:=", 2  
        ],  
        [  
            "NAME:Coordinate",  
            "X:=", 3, "Y:=", 4  
        ]  
    ]  
]
```

	])

VB Syntax	EditDataset <OriginalName> <DatasetDataArray>
VB Example	<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>

EditMaterial

Modifies an existing material.

UI Access	View/Edit Materials command in the material editor		
Parameters	Name	Type	Description
	<OriginalName>	String	Name of the material before editing.
	<MatProperties>	Array	Structured array containing material properties:
			["NAME:<New material name>",

			<pre>"CoordinateSystemType:=", <string>, "BulkOrSurfaceType:=" , <integer>, ["NAME:PhysicsTypes", "set:=" , <array containing string physics types>], <Optional ModifierDataArray>, "permeability:=" , <string containing value>, "conductivity:=" , <string containing value>, "thermal_conductivity:=", <string containing value>, "mass_density:=" , <string containing value>, "specific_heat:=" , <string containing value>, "youngs_modulus:=" , <string containing value>, "poissons_ratio:=" , <string containing value>, "thermal_expansion_coefficient:=", <string containing value>]</pre>
	<i><ModifierDataArray></i>	Array	Optional structured array containing thermal or spatial modifiers: <pre>["NAME:ModifierData", [</pre>

			<pre>"NAME:<ThermalModifierData or SpatialModifierData>", "modifier_data:=" , <"thermal_modifier_data" or "spa- tial_modifier_data">, ["NAME:<all_thermal_modifiers or all_spatial_mod- ifiers>", ["NAME:<modifierName>", "Property::=" , <string property being mod- ified>, "Index::=" , <integer>, "prop_modifier:=" , <"thermal_modifier" or "spa- tial_modifier">, "use_free_form:=" , <Boolean>, "free_form_value:=" , <string modifier value>,]]]</pre>
Return Value	None.		

Python Syn-	EditMaterial (<OriginalName>, <MatProperties>)

tax	
Python Example	Without Modifiers: <pre>oDefinitionManager.EditMaterial("alumina_92pct", ["NAME:alumina_92pct", "CoordinateSystemType:=" , "Cartesian", "BulkOrSurfaceType:=" , 1, ["NAME:PhysicsTypes", "set:=" , ["Electromagnetic","Thermal","Structural"]], "permittivity:=" , "9.3", "dielectric_loss_tangent:=" , "0.008", "thermal_conductivity:=" , "26", "mass_density:=" , "3720", "specific_heat:=" , "790", "youngs_modulus:=" , "267000000000", "poissons_ratio:=" , "0.26", "thermal_expansion_coeffcient:=", "7.2e-006"])</pre>

```
)  
  
With Thermal Modifier:  
  
oDefinitionManager.EditMaterial("copper",  
    [  
        "NAME:copper",  
        "CoordinateSystemType:=", "Cartesian",  
        "BulkOrSurfaceType:=" , 1,  
        [  
            "NAME:PhysicsTypes",  
            "set:=" , ["Electromagnetic","Thermal","Structural"]  
        ],  
        [  
            "NAME:ModifierData",  
            [  
                "NAME:ThermalModifierData",  
                "modifier_data:=" , "thermal_modifier_data",  
                [  
                    "NAME:all_thermal_modifiers",  
                    [  
                        "NAME:one_thermal_modifier",  
                        "Property:=" , "permittivity",
```



```
        "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::=" , "58000000",
"thermal_conductivity::=" , "400",
"mass_density::=" , "8933",
"specific_heat::=" , "385",
"youngs_modulus::=" , "120000000000",
"poissons_ratio::=" , "0.38",
"thermal_expansion_coefficient::=" , "1.77e-05"
])
```

Transient Solve, Non-linear Drude Data Plasma

```
import ScriptEnv

ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
```

```
oDesktop.RestoreWindow()

oProject = oDesktop.SetActiveProject("Drude_plasma_parameters_r231")
oDefinitionManager = oProject.GetDefinitionManager()
oDefinitionManager.EditMaterial("Drude",
    [
        "NAME:Drude",
        "CoordinateSystemType:=", "Cartesian",
        "BulkOrSurfaceType:="    , 1,
        [
            "NAME:PhysicsTypes",
            "set:="                , ["Electromagnetic"]
        ],
        [
            "NAME:AttachedData",
            [
                "NAME:MatNonLinearDrudeFreqDepData",
                "property_data:="    , "nonlinear_drude_data",
                "EpsilonInfinity:="  , "1",
                "PlasmaFrequency:="  , "4.62348462366278GHz",
                "CollisionFrequency:=" , "0.00054491190162662GHz",
                "FieldBreakdown:="   , "10000V_per_meter",
```

	<pre>"PlasmaMaintainFrequency:=", "2.31174231183139GHz", "NeutralDensity:=" , 2.65164580488373E+20, "ElectronDensity:=" , 2.65164580488373E+17, "CollisionRateConstant:=", 2.05499505485618E-15]])</pre>

VB Syntax	EditMaterial <OriginalName>, <MatProperties>
VB Example	Without Modifiers: <pre>oDefinitionManager.EditMaterial "alumina_92pct", Array("NAME:alumina_92pct", "CoordinateSystemType:=", "Cartesian", "BulkOrSurfaceType:=", 1, Array("NAME:PhysicsTypes", "set:=", Array("Electromagnetic","Thermal","Structural")), "permittivity:=", "9.3", "dielectric_loss_tangent:=", "0.008",</pre>

```

"thermal_conductivity:=", "26",
"mass_density:=", "3720",
"specific_heat:=", "790",
"youngs_modulus:=", "267000000000",
"poissons_ratio:=", "0.26",
"thermal_expansion_coefficient:=", "7.2e-006"
)

```

With Thermal Modifier:

```

oDefinitionManager.EditMaterial "copper",
    Array("NAME:copper",
        "CoordinateSystemType:=", "Cartesian",
        "BulkOrSurfaceType:=" , 1,
        Array("NAME:PhysicsTypes",
            "set:=" , Array("Electromagnetic","Thermal","Structural")),
        Array("NAME:ModifierData",
            Array("NAME:ThermalModifierData",
                "modifier_data:=" , "thermal_modifier_data",
                Array("NAME:all_thermal_modifiers",
                    Array("NAME:one_thermal_modifier",
                        "Property:=" , "permittivity",
                        "Index:=" , 0,

```

	<pre> "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:=" , "58000000", "thermal_conductivity:=" , "400", "mass_density:=" , "8933", "specific_heat:=" , "385", "youngs_modulus:=" , "120000000000", "poissons_ratio:=" , "0.38", "thermal_expansion_coefficient:=" , "1.77e-05")</pre>

ExportDataset

Exports a dataset to a named file. This can be executed by the oProject, or oDesign variables.

UI Access	Project > Datasets > Export.

Parameters	Name	Type	Description
	<datasetFilePath>	String	The full path to the file.
Return Value	None.		

Python Syntax	ExportDataset (<datasetFilePath>)
Python Example	<pre>oProject.ExportDataset('e:/tmp/dsdata.txt') oDesign.ExportDataset('e:/tmp/dsdata.txt')</pre>

VB Syntax	ExportDataset <datasetFilePath>
VB Example	<pre>oProject.ExportDataset "e:/tmp/dsdata.txt" oDesign.ExportDataset "e:/tmp/dsdata.txt"</pre>

ExportMaterial

Exports a local material to a library.

UI Access	Export to Library command in the material editor.		
Parameters	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.
	<LibraryLocation>	String	Location to save the library. Only "PersonalLib" and "UserLib" are allowed.

Return Value	None.

Python Syntax	<code>ExportMaterial (<ExportData>, <LibraryLocation>)</code>
Python Example	<pre>oDefinitionManager.ExportMaterial (["NAME:mylib",_ "Material1", "Material2", "Material3"], "PersonalLib")</pre>

VB Syntax	<code>ExportMaterial <ExportData>, <LibraryLocation></code>
VB Example	<pre>oDefinitionManager.ExportMaterial Array("NAME:mylib",_ "Material1", "Material2", "Material3"), "PersonalLib"</pre>

GetActiveDesign

Returns the design in the active project

Note: GetActiveDesign will return normally if there are no active objects.

UI Access	N/A
Parameters	None.
Return Value	Object of the active design.
Python Syntax	<code>GetActiveDesign()</code>
Python Example	<pre>oDesign = oProject.GetActiveDesign()</pre>

VB Syntax	GetActiveDesign
VB Example	Set oDesign = oProject.GetActiveDesign

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.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing names of variables.

Python Syntax	GetArrayVariables()
Python Example	<pre>oProject.GetArrayVariables() oDesign.GetArrayVariables()</pre>

VB Syntax	GetArrayVariables
VB Example	<pre>oProject.GetArrayVariables oDesign.GetArrayVariables</pre>

GetChildNames [Project]

Returns the names of the project's child objects.

UI Access	N/A		
Parameters	Name	Type	Description
	<type>	String	(Optional) Default is "design"; Any value returned by GetChildTypes() can be used.
Return Value	Array of names of children for the queried object.		

Python Syntax	GetChildNames (<type>)
Python Example	<pre>arrDesignNames = oProject.GetChildNames() arrVarbleNames = oProject.GetChildNames("Variable")</pre>

VB Syntax	GetChildNames (<type>)
VB Example	<pre>arrDesignNames = oProject.GetChildNames() arrVarbleNames = oProject.GetChildNames ("Variable")</pre>

GetChildObject [Project]

Returns the project's child objects.

Note:

`GetChildObject` will return normally if there are no active objects.

UI Access	N/A		
Parameters	Name	Type	Description
	<path>	String	The path may include multiple generations (for example, "designObject/moduleObj/SetupObject"). See Object Path .
Return Value	Object of a found child.		

Python Syntax	<code>GetChildObject(<path>)</code>
Python Example	<pre>oProject = oDesktop.GetActiveProject() oDesign = oProject.GetChildObject("TeeModel") oVariable = oProject.GetChildObject("VariableName") oReport = oProject.GetChildObject("TeeModel/Results/S Parameter Plot 1")</pre>

VB Syntax	<code>GetChildObject(<path>)</code>
VB Example	<pre>Set oProject = oDesktop.GetActiveproject() Set oDesign = oProject.GetChildObject("TeeModel") Set oVariable = oProject.GetChildObject("VariableName") Set oReport = oProject.GetChildObject("TeeModel/Results/S Parameter Plot 1")</pre>

GetChildTypes [Project]

Returns the types of the project's child objects.

UI Access	N/A
Parameters	None.
Return Value	Array of string represents types of the child objects.

Python Syntax	GetChildTypes()
Python Example	<code>oProject.GetChildTypes()</code>

VB Syntax	GetChildTypes
VB Example	<code>oProject.GetChildTypes</code>

GetConfigurableData (Project)

Note:

This command is for internal Ansys use only.

Python Syntax	<code>GetConfigurableData()</code>
Python Example	<code>oProject.GetConfigurableData()</code>

GetDefinitionManager

Gets the `DefinitionManager` object.

UI Access	N/A
Parameters	None.
Return Value	<code>DefinitionManager</code> object.

Python Syntax	<code>GetDefinitionManager()</code>
Python Example	<code>oDefinitionManager = oProject.GetDefinitionManager()</code>

VB Syntax	<code>GetDefinitionManager</code>
VB Example	<code>Set oDefinitionManager = oProject.GetDefinitionManager</code>

Note: For more information on commands for the `DefinitionManager`, see [Definition Manager Script Commands](#).

GetDependentFiles

Provides a list of the external files referenced in the project, including characteristic (for example, MDX) and coupled project files.

UI Access	N/A
Parameters	None.
Return Value	List of referenced files.

Python Syntax	GetDependentFiles()
Python Example	<pre>files = oProject.GetDependentFiles()</pre>

VB Syntax	GetDependentFiles
VB Example	<pre>files = oProject.GetDependentFiles</pre>

GetDesign

Returns the interface to a specific design in a given project.

UI Access	N/A		
Parameters	Name	Type	Description
	<designName>	String	Name of the design.
Return Value	Object of the specified design.		

Python Syntax	GetDesign (< <i>designName</i> >)
Python Example	<code>oProject.GetDesign ("HFSSDesign1")</code>

VB Syntax	GetDesign < <i>designName</i> >
VB Example	<code>oProject.GetDesign "HFSSDesign1"</code>

GetDesigns

Obtains all designs in the current project.

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

Python Syntax	GetDesigns()
Python Example	<code>oProject.GetDesigns()</code>

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

GetEDBHandle

Returns the EDB handle for the project.

Important:

This script is for internal Ansys use only.

UI Access	N/A
Parameters	None.
Return Value	String indicating the EDB handle for the project.

Python Syntax	GetEDBHandle()
Python Example	<code>oProject.GetEDBHandle()</code>

VB Syntax	GetEDBHandle
VB Example	<code>oProject.GetEDBHandle</code>

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	<code>oProject.GetLegacyName()</code>

VB Syntax	GetLegacyName
VB Example	<code>oProject.GetLegacyName</code>

GetName [Project]

Obtains the project name

UI Access	N/A
Parameters	None.
Return Value	String containing the project name, not including the path or extension.

Python Syntax	GetName()
Python Example	<code>oProject.GetName()</code>

VB Syntax	GetName
VB Example	<code>oProject.GetName</code>

GetObjPath [Project]

Obtains the project name from the full path.

UI Access	N/A
Parameters	None.
Return Value	String containing only the project name.

Python Syntax	GetObjPath()
Python Example	<code>oProject.GetObjPath()</code>

VB Syntax	GetObjPath

VB Example	<code>oProject.GetObjPath</code>

GetPath

Returns the location of the project on disk.

UI Access	N/A
Parameters	None.
Return Value	String containing the path to the project, not including the project name.

Python Syntax	<code>GetPath()</code>
Python Example	<code>oProject.GetPath()</code>

VB Syntax	<code>GetPath</code>
VB Example	<code>oProject.GetPath</code>

GetPropEvaluatedValue

Returns the Evaluated-Value for Value-Property and Variable. Returns the Property-value as text string for other property types

Note:

This command is not supported by the EMIT and Circuit design types.

UI Access	N/A		
Parameters	Name	Type	Description
	<PropName>	String	Name of the property.
Return Value	String value of the evaluated value.		

Python Syntax	GetPropEvaluatedValue (<PropName>)		
Python Example	<pre>oVar = oDesign.GetChildObject(" Variables/var") oVar.GetPropEvaluatedValue()</pre>		

GetPropNames [Project]

Obtains the property name of the object. At the project level, GetPropNames always returns empty because the project is not associated with any property.

UI Access	N/A		
Parameters	Name	Type	Description
	<includeReadOnly>	Boolean	(Optional) • True – Include read only props. • False – Do not include read only props.
Return Value	Empty array.		

Python Syntax	GetPropNames ()
Python Example	<code>oProject.GetPropNames ()</code>

VB Syntax	GetPropNames
VB Example	<code>oProject.GetPropNames</code>

GetPropSIValue

Returns the SI-Value for Value-Property and Variable. Return NAN for other property type if its value is not able to convert to be a double-floating point value.

Note:

This command is not supported by the EMIT and Circuit design types.

UI Access	N/A		
Parameters	Name	Type	Description
	<PropName>	String	Name of the property.
Return Value	Property value as a double floating value, or NAN if the property value cannot be converted to double floating point.		

Python Syntax	GetPropSIValue (<PropName>)

Python Example	<pre>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</pre>

GetPropValue [Project]

Returns the property value for the active project object, or specified property values.

UI Access	N/A		
Parameters	Name	Type	Description
	<propPath>	String	A child object's property path. See: Property Function Summary .
Return Value	String of property value.		

Python Syntax	GetPropValue(<propPath>)
Python Example	<pre>oProject.GetPropValue("TeeModel/offset")</pre>

VB Syntax	GetPropValue <propPath>

VB Example	<code>oProject.GetPropValue "TeeModel/offset"</code>

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.

UI Access	N/A		
Parameters	Name	Type	Description
	<PropTab>	String	One of the following, where tab titles are shown in parentheses: - PassedParameterTab ("Parameter Values") - DefinitionParameterTab (Parameter Defaults) - LocalVariableTab ("Variables" or "Local Variables") - ProjectVariableTab ("Project variables") - ConstantsTab ("Constants") - BaseElementTab ("Symbol" or "Footprint") - ComponentTab ("General") - Component("Component") - CustomTab ("Intrinsic Variables") - Quantities ("Quantities") - Signals ("Signals")
	<PropServer>	String	An object identifier, generally returned from another script method, such as <code>CompInst@R;2;3</code>
Return Value	Array of strings containing the names of the appropriate properties.		

Python Syntax	<code>GetProperties(<PropTab>, <PropServer>)</code>

Python Example	<code>oEditor.GetProperties('PassedParameterTab', 'k')</code>
VB Syntax	<code>GetProperties <PropTab>, <PropServer></code>
VB Example	<code>oEditor.GetProperties "PassedParameterTab", "k"</code>

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.

UI Access	N/A		
Parameters	Name	Type	Description
	<i><PropTab></i>	String	One of the following, where tab titles are shown in parentheses: - PassedParameterTab ("Parameter Values") - DefinitionParameterTab (Parameter Defaults) - LocalVariableTab ("Variables" or "Local Variables") - ProjectVariableTab ("Project variables") - ConstantsTab ("Constants") - BaseElementTab ("Symbol" or "Footprint") - ComponentTab ("General") - Component("Component") - CustomTab ("Intrinsic Variables")

			- Quantities ("Quantities") - Signals ("Signals")
	<i><PropServer></i>	String	An object identifier, generally returned from another script method, such as <code>CompInst@R;2;3</code>
	<i><PropName></i>	String	Name of the property.
Return Value	String value of the property.		

Python Syntax	<code>GetPropertyValue (<PropTab>, <PropServer>, <PropName>)</code>		
Python Example	<pre>selectionArray = oEditor.GetSelections() for k in selectionArray: val = oEditor.GetPropertyValue("PassedParameterTab", k, "R") ...</pre>		

VB Syntax	<code>GetPropertyValue (<PropTab>, <PropServer>, <PropName>)</code>		
VB Example	<pre>selectionArray = oEditor.GetSelections for k in selectionArray: val = oEditor.GetPropertyValue("PassedParameterTab", k, "R") ...</pre>		

GetTopDesignList

Returns a list of top-level design names.

UI Access	N/A
Parameters	None.
Return Value	List of strings containing name of top-level designs.

Python Syntax	GetTopDesignList()
Python Example	<code>oProject.GetTopDesignList()</code>

VB Syntax	GetTopDesignList
VB Example	<code>oProject.GetTopDesignList</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`.

UI Access	N/A		
Parameters	Name	Type	Description
	<VarName>	String	Name of the variable to access.

Return Value	String represents the value of the variable.

Python Syntax	<code>GetVariableValue(<VarName>)</code>
Python Example	<code>oProject.GetVariableValue("var_name")</code>

VB Syntax	<code>GetVariableValue <VarName></code>
VB Example	<code>oProject.GetVariableValue "var_name"</code>

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`.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing the variables.

Python Syntax	<code>GetVariables()</code>
Python Example	<code>oProject.GetVariables() oDesign.GetVariables()</code>

VB Syntax	GetVariables
VB Example	<pre>oProject.GetVariables oDesign.GetVariables</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.		
Parameters	Name	Type	Description
	<datasetFileFullPath>	String	The full path to the file containing the dataset values. *.tab files recommended (see note below).
	<optionalDatasetName>	String	Optional. User-defined dataset name.
Return Value	None.		

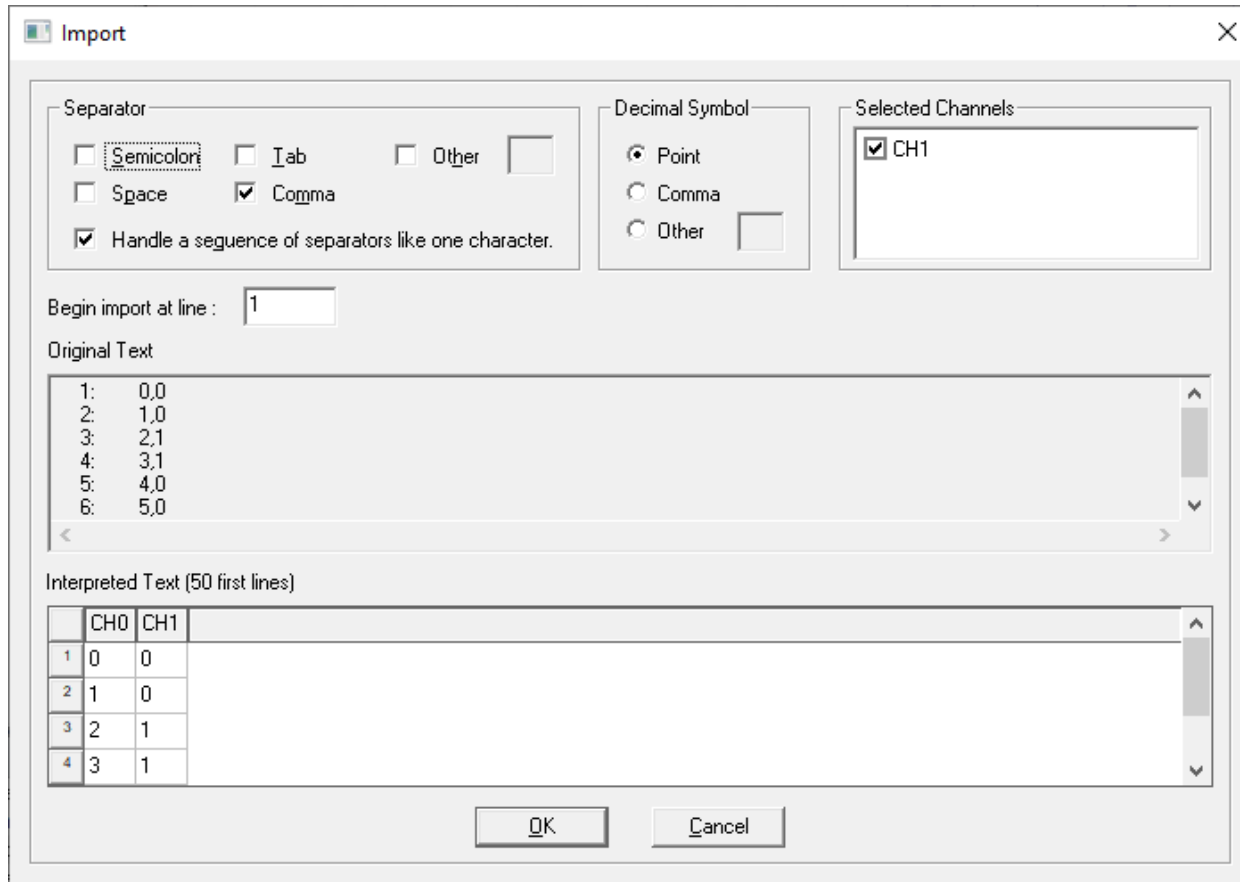
Python Syntax	ImportDataset (<datasetFileFullPath>, <optionalDatasetName>)
Python Example	<pre>oProject.ImportDataset('e:\tmp\dsdata.tab') oDesign.ImportDataset('e:\tmp\dsdata.tab') oProject.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName') oDesign.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName')</pre>

VB Syntax	ImportDataset <datasetFilePath>, <optionalDatasetName>
VB Example	<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**.



InsertDesign

Inserts a new design in the project. For Circuit and HFSS 3D Layout scripts, the last argument will always be empty.

UI Access	Project > [Insert Circuit Design / Insert Circuit Netlist / Insert HFSS 3D Layout Design].		
Parameters	Name	Type	Description
	<DesignType>	String	Design type. Valid types: "Circuit", "Nexxim Circuit", "System", "Nexxim Netlist", "HFSS3D".
	<DesignName>	String	Design name.
	<TechnologyFile>	String	The path to the Circuit technology file to be used in the design. Use a pair of empty double quotes ("") for none.
	<SubcircuitID>	String	(Optional) Must be preceded by a colon if included along with the Technology File name. No colon is necessary when the subcircuit ID is omitted.
	""	None	Empty argument.
Return Value	None.		

Python Syntax	InsertDesign (<DesignType>, <DesignName>, <TechnologyFile>:<SubCircuitID>,"")
Python Example	<pre>oProject.InsertDesign("Nexxim Circuit","MyDesign", "C:\\Program Files\\ANSYS Inc\\v251\\AnsysEM", "")</pre>

VB Syntax	InsertDesign <DesignType>, <DesignName>, <TechnologyFile>:<SubCircuitID>,""
VB Example	<pre>oProject.InsertDesign "Nexxim Circuit","MyDesign", "C:\\Program Files\\ANSYS Inc\\v251\\AnsysEM", ""</pre>

InsertDesign (Layout Editor)

Insert a design

Command: None

Syntax: InsertDesign <type>, <name>, <stationerypath>, <parentname>

Return Value: None

Parameters: <type>

Type: string

<name>

Type: string

<stationerypath>

Type: string

<parentname>

Type: string

VB Example: oDesign.InsertDesign <type>, <name>, <stationerypath>, <parentname>

where, <type> is NULL for top-level design

InsertDesignWithWorkflow

Inserts a design with a named workflow and returns an IDispatch string.

UI Access	N/A		
Parameters	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.

	<code><stationaryPath></code>	String	Path.
Return Value	IDispatch string, such as 'IDispatch(IAltraSimScript)'		

Python Syntax	<code>InsertDesignWithWorkflow(<type>, <workflowName>, <specName>, <fileName>, <libLoc>, <stationaryPath>)</code>
Python Example	<pre>oProject.InsertDesignWithWorkflow ("Circuit Design", "Serial Design", "PCIE3 Stressed", "LongChannel", "SysLib", "C:\\Program Files\\ANSYS Inc\\v251\\AnsysEM\\syslib\\MS" - RT_duroid 6010 (Er=10.2) 0.010 inch, 0.5 oz copper.asty")</pre>

VB Syntax	<code>InsertDesignWithWorkflow <type>, <workflowName>, <specName>, <fileName>, <libLoc>, <stationaryPath></code>
VB Example	<pre>oProject.InsertDesignWithWorkflow "Circuit Design", "Serial Design", _ "PCIE3 Stressed", "LongChannel", "SysLib", _ "C:\\Program Files\\ANSYS Inc\\v251\\AnsysEM\\syslib\\MS - RT_duroid 6010" & _ " (Er=10.2) 0.010 inch, 0.5 oz copper.asty"</pre>

InsertToolObject

Note:

This command is for internal Ansys use only.

Python Syntax	<code>InsertToolObject()</code>
Python Example	<code>oProject.InsertToolObject()</code>

Paste (Project Object)

Pastes a design in the active project.

UI Access	Edit > Paste.
Parameters	None.
Return Value	None

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

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

Redo [Project Level]

Reapplies the last project-level command.

UI Access	Edit > Redo.

Parameters	None.
Return Value	None.

Python Syntax	Redo()
Python Example	<code>oProject.Redo()</code>

VB Syntax	Redo
VB Example	<code>oProject.Redo</code>

RemoveAllUnusedDefinitions

Removes all unused project definitions.

UI Access	N/A
Parameters	None.
Return Value	None.

Python Syntax	RemoveAllUnusedDefinitions()
Python Example	<code>oProject.RemoveAllUnusedDefinitions()</code>

VB Syntax	RemoveAllUnusedDefinitions

VB Example	<code>oProject.RemoveAllUnusedDefinitions</code>

RemoveMaterial

Removes a material from a library.

UI Access	Remove Material(s) command in the material editor		
Parameters	Name	Type	Description
	<MaterialName>	String	Name of the material to be removed.
	<IsProjectMaterial>	Boolean	If True, 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".
Return Value	None.		

Python Syntax	<code>RemoveMaterial (<MaterialName>, <IsProjectMaterial>, <LibraryName>, <LibraryLocation>)</code>
Python Example	<code>oDefinitionManager.RemoveMaterial (["Material1", false, "mo0907","UserLib"])</code>

VB Syntax	<code>RemoveMaterial <MaterialName>, <IsProjectMaterial>, <LibraryName>, <LibraryLocation></code>
VB Example	<code>oDefinitionManager.RemoveMaterial "Material1", false, "mo0907", "UserLib"</code>

RemoveUnusedDefinitions

Removes any unused project definitions.

UI Access	Tools > Project Tools > Remove Unused Definitions.		
Parameters	Name	Type	Description
	<code><Definitions></code>	Array	Definitions to be removed, such as materials and surface materials.
Return Value	None.		

Python Syntax	<code>RemoveUnusedDefinitions(<Definitions>)</code>
Python Example	<pre>oProject.RemoveUnusedDefinitions([["NAME:Materials", "Al-Extruded"], ["NAME:SurfaceMaterials",</pre>

	<pre>"Steel-oxidised-surface"]])</pre>

VB Syntax	RemoveUnusedDefinitions <Definitions>
VB Example	<pre>oProject.RemoveUnusedDefinitions Array(Array("NAME:Materials", "Al-Extruded"), Array ("NAME:SurfaceMaterials", _ "Steel-oxidised-surface"))</pre>

Rename

Renames the project and saves it. Similar to [SaveAs\(\)](#).

UI Access	Edit > Rename.		
Parameters	Name	Type	Description
	<NewName>	String	Desired name of the project. The path is optional.
	<OverWriteOk>	Boolean	• True - overwrite the file on disk if it exists. • False - prevent overwrite.
Return Value	None.		

Python Syntax	Rename(<NewName>,<OverWriteOK>)

Python Example	<code>oProject.Rename("c:\projects\MyProject.aedt", True)</code>

VB Syntax	<code>Rename <NewName>,<OverWriteOK></code>
VB Example	<code>oProject.Rename "c:\projects\MyProject.aedt", 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.

UI Access	[HFSS 3D Layout] > Toolkit > [Script Name].		
Parameters	Name	Type	Description
	<code><LibraryType></code>	String	Name of the library in which the script is located.
	<code><ToolkitName></code>	String	Name of the Python script.
	<code><ToolkitArg></code>	Array	Structured array containing arguments required by the toolkit script.
Return Value	User-defined solution(s) and/or output(s).		

Python Syntax	<code>RunToolkit(<LibraryType>, <ToolkitName>, <ToolkitArg>)</code>
Python Example	<code>oDesign.RunToolkit("SysLib", "Hearing Aid Compliance", [])</code>

VB Syntax	RunToolkit <LibraryType>, <ToolkitName>, <ToolkitArg>
VB Example	<pre>oDesign.RunToolkit "SysLib", "Wavelength Calculator", Array() oDesign.RunToolkit "SysLib", "Hearing Aid Compliance", Array()</pre>

Save

Saves the active project.

UI Access	File > Save.
Parameters	None.
Return Value	None.

Python Syntax	Save()
Python Example	<pre>oProject.Save()</pre>

VB Syntax	Save
VB Example	<pre>oProject.Save</pre>

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.

UI Access	File > Save As.		
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:Over- rideActions', ['NAME:ef_copy_no_overwrite', ['NAME:Files',</pre>

	<pre>'\$PROJECTDIR/circuit_models.inc']], ['NAME:ef_make_path_absolute', ['NAME:Files', '\$PROJECTDIR/SL_6s.sp']]]])</pre>

VB Syntax	For non-Schematic/Layout project: SaveAs <NewName> <OverWriteOK> For Schematic/Layout project: SaveAs <NewName> <OverWriteOK> <DefaultAction> <OverrideActions>
VB Examples	<pre>oProject.SaveAs "D:/projects/project1.aedt", true ---- oProject.SaveAs "F:\Ansys 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.

UI Access	N/A		
Parameters	Name	Type	Description
	<projectName>	String	The desired name of the project, with directory and extension.
Return Value	None.		

Python Syntax	SaveAsStandAloneProject(<projectName>)
Python Example	<code>oProject.SaveAsStandAloneProject('D:/projects/project1.aedt')</code>

VB Syntax	SaveAsStandAloneProject <projectName>
VB Examples	<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.
	<ArchiveNotes>	String	String describe the archive.
Return Value	None.		

Python Syntax	SaveProjectArchive(<archivefilepath>, <IncludeExternalFiles>, <IncludeResultsFiles>, <AdditionalFiles>, <ArchiveNotes>)
Python	<code>oProject.SaveProjectArchive("C:\\Users\\Documents\\Ansoft\\Project27.aedt", True,</code>

Example	<code>False, [], "")</code>

VB Syntax	<code>SaveProjectArchive <archivefilepath>, <IncludeExternalFiles>, <IncludeResultsFiles>, <AdditionalFiles>, <ArchiveNotes></code>
VB Example	<code>oProject.SaveProjectArchive "C:\Documents\OptimTee.aedt", true, false, Array(), "My notes"</code>

SetActiveDefinitionEditor

Obtains a specified definition editor.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><EditorName></code>	String	Name of the definition editor to set active, one of "SymbolEditor", "FootprintEditor".
	<code><DefinitionName></code>	String	The combination name for the symbol or footprint, <code><libname>:<def-name></code>
Return Value	Object for the definition to be edited.		

Python Syntax	<code>SetActiveDefinitionEditor(<EditorName>, <DefinitionName>)</code>
Python Example	<code>oProject.SetActiveDefinitionEditor("SymbolEditor", "Simplorer Elements\Basic Elements\Circuit\Passive Elements:R")</code>

VB Syntax	SetActiveDefinitionEditor <EditorName>, <DefinitionName>
VB Example	<pre>oProject.SetActiveDefinitionEditor "SymbolEditor",_ "Simplorer Elements\Basic Elements\Circuit\Passive Elements:R"</pre>

SetActiveDesign

Sets a design to be the active design.

UI Access	N/A		
Parameters	Name	Type	Description
	<DesignName>	String	Name of the design to set as the active design.
Return Value	None.		

Python Syntax	SetActiveDesign (<DesignName>)
Python Example	<pre>oDesign = oProject.SetActiveDesign("SimplorerDesign2")</pre>

VB Syntax	SetActiveDesign <DesignName>
VB Example	<pre>Set oDesign = oProject.SetActiveDesign "SimplorerDesign2"</pre>

SetPropValue [Project]

Sets a property value for an active project's child object.

UI Access	Edit Properties on ProjectTree objects.		
Parameters	Name	Type	Description
	<propPath>	String	A child object's property path. See: Property Function .
	<newValue>	String	New property value.
Return Value	Boolean: • True – property found. • False – property not found.		

Python Syntax	SetPropValue(<propPath>, <newValue>)
Python Example	<pre>oProject.SetPropValue("TeeModel/offset", "2mm") oProject.SetPropValue("TeeModel/Results/S Parameter Plot 1/Display Type", "Data Table")</pre>

VB Syntax	SetPropValue <propPath>, <newValue>
VB Example	<pre>oProject.SetPropValue "TeeModel/offset", "2mm" oProject.SetPropValue "TeeModel/Results/S Parameter Plot 1/Display Type", "Data Table"</pre>

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.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><propTab></code>	String	One of the following, where tab titles are shown in parentheses: - PassedParameterTab ("Parameter Values") - DefinitionParameterTab (Parameter Defaults") - LocalVariableTab ("Variables" or "Local Variables") - ProjectVariableTab ("Project variables") - ConstantsTab ("Constants") - BaseElementTab ("Symbol" or "Footprint") - ComponentTab ("General") - Component("Component") - CustomTab ("Intrinsic Variables") - Quantities ("Quantities") - Signals ("Signals")
	<code><propServer></code>	String	An object identifier, generally returned from another script method, such as <code>CompInst@R;2;3</code>
	<code><propName></code>	String	Name of the property.
	<code><propValue></code>	String	The value for the property
Return Value	None.		

Python Syntax	SetPropertyValue(<propTab>, <propServer>, <propName>, <propValue>)
Python Example	oEditor.SetPropertyValue("PassedParameterTab", "k", "R", "2200")

VB Syntax	SetPropertyValue <propTab>, <propServer>, <propName>, <propValue>
VB Example	oEditor.SetPropertyValue "PassedParameterTab", "k", "R", "2200"

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`.

UI Access	N/A		
Parameters	Name	Type	Description
	<VarName>	String	Variable name.
	<VarValue>	Value	New value for the variable.
Return Value	None.		

Python Syntax	SetVariableValue (<VarName>, <VarValue>)
Python Example	oProject.SetVariableValue('\$Var1', '3mm')

VB Syntax	SetVariableValue <VarName>, <VarValue>

VB Example	<code>oProject.SetVariableValue "\$Var1", "3mm"</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.

UI Access	N/A
Parameters	None.
Return Value	None.

Python Syntax	<code>SimulateAll()</code>
Python Example	<code>oProject.SimulateAll()</code>

VB Syntax	<code>SimulateAll</code>
VB Example	<code>oProject.SimulateAll</code>

Undo [Project]

Cancels the last project-level command.

UI Access	Edit > Undo.

Parameters	None.
Return Value	None.

Python Syntax	Undo()
Python Example	<code>oProject.Undo()</code>

VB Syntax	Undo
VB Example	<code>oProject.Undo</code>

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 .
Parameters	None.
Return Value	None.

Python Syntax	UpdateDefinitions()
Python Example	<code>oProject.UpdateDefinitions()</code>

VB Syntax	UpdateDefinitions
VB Example	<code>oProject.UpdateDefinitions</code>

7 - Design Object Script Commands

Design object commands should be executed by the oDesign object.

```
oDesign.CommandName <args>
```

For example:

```
Set oDesign = oProject.SetActiveDesign("HFSSDesign1")
```

Conventions Used in this Chapter

<ModuleName> is a placeholder for any one of the following modules:

- [Cavity Commands](#) – "Cavities"
- [Design Verification Module](#) – "DV"
- [Excitations Commands](#) – "Excitations" or "Boundaries"
- [Field Overlays Module](#) – "FieldsReporter"
- [Optimetrics Module](#) – "Optimetrics"
- [Output Variable Module](#) – "OutputVariable"
- [Radiation Setup Manager Module](#) – "RadiationSetupMgr"
- [Reporter Module](#) – "ReportSetup"
- [Simulation Setup Commands](#) – "SolveSetups"
- [User Defined Documents Module](#) – "UserDefinedDocuments"
- [User Defined Solutions Module](#) – "UserDefinedSolutionModule"

[AddDataset](#)

[AddDesignVariablesForDynamicLink](#)

[AddDynamicLink](#)

[AddModelingProperties](#)

[Analyze](#)

[AnalyzeAll](#)

[AnalyzeAllNominal](#)

[AnalyzeDistributed](#)

[ApplyMeshOps](#)

ChangePortProperty [Nexxim]

[ChangeProperty](#)

[CopyItemCommand](#)

[CreateReport](#)

[DeleteDataset](#)

[DeleteDesignInstance](#)

[DeleteFieldVariation](#)

[DeleteFullVariation](#)

[DeleteOutputVariable](#)

[DeletePlotEntities](#)

DeletePort [Nexxim]

DeleteSource [Nexxim]

[DistributeAnalysis](#)

[EditCoSimulationOptions](#)

[EditDataset](#)

EditDesignSettings[EditImportData](#)[EditInfiniteArray](#)[EditLayoutForLayoutComponent](#)[EditNotes](#)[EditNotes \(Layout Editor\)](#)[EditOptions](#)[EditOutputVariable](#)[EMDesignOptions](#)[ExportCircuit \(Layout Editor\)](#)[ExportConvergence](#)[ExportDataset](#)[ExportForHSpice](#)[ExportForHSpice \(Layout Editor\)](#)[ExportForSpice \(Layout Editor\)](#)[ExportNMFData](#)[ExportLayoutViaCurrentDensity](#)[ExportMeshStats](#)[ExportNetworkData](#)[ExportNetworkData \(Layout Editor\)](#)[ExportNMFData](#)

[ExportProfile](#)

[ExportReport](#)

[GenerateMesh](#)

[GetActiveEditor](#)

[GetActiveEditor \(Layout Editor\)](#)

[GetAllPorts](#)

[GetBoundingBox](#)

[GetChildNames \[Design\]](#)

[GetChildObject \[Design\]](#)

[GetChildTypes \[Design\]](#)

[GetConfigurableData](#)

[GetData](#)

[GetDesignID](#)

[GetDesignType](#)

[GetDesignName](#)

[GetDesignValidationInfo](#)

[GetEditor](#)

[GetEditor \(Layout Editor\)](#)

[GetGeometryIdsForAllNetLayerCombinations](#)

[GetGeometryIdsForNetLayerCombinations](#)

[GetManagedFilePath](#)

[GetModule](#)

[GetName](#)

[GetNominalVariation](#)

[GetNoteText](#)

[GetObjPath](#)

[GetOutputVariableValue](#)

[GetOutputVariables](#)

[GetProject](#)

[GetPostProcessingVariables](#)

[GetPropHost](#)

[GetPropNames \[Design\]](#)

[GetPropValue \[Design\]](#)

[GetProperties](#)

[GetPropertyValue](#)

[GetSelections](#)

[GetSolutionType](#)

[GetSolveInsideThreshold](#)

[GetVariableValue](#)

[GetVariables](#)

[ImportDataFilePath \[Nexxim\]](#)

[ImportDataset](#)

[InsertDesign](#)

[OverlayCurrents](#)

[OverlayFarField](#)

[OverlayMesh](#)

[OverlayNearField](#)

[PasteArray](#)

[PasteDesign](#)

PasteItemCommand

[PushExcitations](#)

[Redo](#)

[RemoveImportData \(Layout Editor\)](#)

[RemoveModelingProperties](#)

[RenameDesignInstance](#)

RenameImportData [Nexxim]

[RenameSource](#)

[ReportTemplates](#)

[RunToolkit](#)

[SetActiveEditor](#)

[SetBendGraphicsMode](#)

[SetDesignMode](#)[SetPropValue \[Design\]](#)[SetPropertyValue](#)[SetShowLayoutForLayoutComponent](#)[SetSolveInsideThreshold](#)[SetSourceContexts](#)[SetVariableValue](#)[SimulateLink](#)[Solve](#)[StartAnalysis](#)[StopSimLink](#)[Undo](#)[ValidateCircuit](#)[ValidateLink](#)

AddDataset

Adds a dataset. This can be executed by the oProject, or oDesign variables.

UI Access	Project > Datasets > Add.		
Parameters	Name	Type	Description
	< <i>DatasetDataArray</i> >	Array	Array("NAME:<DatasetName>", Array("NAME:Coordinates", <CoordinateArray>, <CoordinateArray>, ...)

	<table><tr><td><DatasetName></td><td>String</td><td>Name of the dataset.</td></tr><tr><td><CoordinateArray></td><td>Array</td><td>Array("NAME:Coordinate", "X:=", <double>, "Y:=", <double>)</td></tr></table>			<DatasetName>	String	Name of the dataset.	<CoordinateArray>	Array	Array("NAME:Coordinate", "X:=", <double>, "Y:=", <double>)
	<DatasetName>	String	Name of the dataset.						
<CoordinateArray>	Array	Array("NAME:Coordinate", "X:=", <double>, "Y:=", <double>)							
Return Value	None.								

Python Syntax	AddDataset <DatasetDataArray>
Python Example	<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	<pre>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)))</pre>

AddDesignVariablesForDynamicLink

Creates design variables based on any in a Circuit, HFSS, 3D Layout, Q3D, or 2D extractor project that is connected to a Circuit or 3D Layout design through a dynamic link.

UI Access	In the Schematic Editor or Layout Editor, right click the new instance of the linked design and click Add

	Design Variables for Dynamic Link. In the Project Manager, right click on [Project Name] > [Design Name] > [Dynamic Link Design] and click Add Design Variables for Dynamic Link.		
Parameters	Name	Type	Description
	<InstanceID>	String	Instance ID for the dynamic link component
Return Value	None		

Python Syntax	AddDesignVariablesForDynamicLink (<InstanceID>)
Python Example	<pre>oDesign = oProject.SetActiveDesign("Circuit1") oDesign.AddDesignVariablesForDynamicLink ("6")</pre>

VB Syntax	AddDesignVariablesForDynamicLink <InstanceID>
VB Example	<pre>Set oDesign = oProject.SetActiveDesign("Circuit1") oDesign.AddDesignVariablesForDynamicLink "6"</pre>

AddDynamicLink

Adds a dynamic link component to a Circuit or HFSS 3D Layout design. Circuit designs can accept dynamic links from HFSS, Q3D, 2D Extractor, and SIwave.

Note:

HFSS 3D Layout designs can only accept dynamic links from HFSS.

UI Access	- Open the Circuit or HFSS 3D Layout design to which you want to add the link. On the Component Libraries menu, open the Symbols tab, and then click Import Models. Select the appropriate icon from the panel of icons. The Import Components dialog opens. - If the source for a dynamic link is an HFSS design, in the Project Manager drag and drop the design onto a Circuit or HFSS 3D Layout design to create the link.																													
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><designName></td><td>String</td><td>Design source for the dynamic link. Optional, defaults to the first design in the project pointed to by projectPath.</td></tr><tr><td><projectPath></td><td>String</td><td>Project source for the dynamic link. Optional, defaults to the current active project.</td></tr><tr><td><linkName></td><td>String</td><td>Name applied to the dynamic link. Optional, defaults to the designName.</td></tr><tr><td><solutionName></td><td>String</td><td>Solution in source design which will be referenced by the dynamic link. Optional, defaults to the first setup:sweep.</td></tr><tr><td><transLinePort ></td><td>String</td><td>For HFSS designs only, port name for transmission lines. It is created only if it is <i>not</i> empty. Optional, defaults to “”. This value is ignored for non-HFSS design types.</td></tr><tr><td><ReduceMatrix></td><td>String</td><td>For Q3D and 2D extractor designs, reduce matrix which will be referenced by the dynamic link. This is ignored for other design types. Optional, defaults to “Original”.</td></tr><tr><td><EnableHFSSCableModeling></td><td>Boolean</td><td>For 2D extractor designs only, flag to create cable models. This is ignored for other designs types. Optional, defaults to false.</td></tr><tr><td><Description></td><td>String</td><td>Description of the dynamic link's source. Optional.</td></tr></table>			Name	Type	Description	<designName>	String	Design source for the dynamic link. Optional, defaults to the first design in the project pointed to by projectPath.	<projectPath>	String	Project source for the dynamic link. Optional, defaults to the current active project.	<linkName>	String	Name applied to the dynamic link. Optional, defaults to the designName.	<solutionName>	String	Solution in source design which will be referenced by the dynamic link. Optional, defaults to the first setup:sweep.	<transLinePort >	String	For HFSS designs only, port name for transmission lines. It is created only if it is <i>not</i> empty. Optional, defaults to “”. This value is ignored for non-HFSS design types.	<ReduceMatrix>	String	For Q3D and 2D extractor designs, reduce matrix which will be referenced by the dynamic link. This is ignored for other design types. Optional, defaults to “Original”.	<EnableHFSSCableModeling>	Boolean	For 2D extractor designs only, flag to create cable models. This is ignored for other designs types. Optional, defaults to false.	<Description>	String	Description of the dynamic link's source. Optional.
Name	Type	Description																												
<designName>	String	Design source for the dynamic link. Optional, defaults to the first design in the project pointed to by projectPath.																												
<projectPath>	String	Project source for the dynamic link. Optional, defaults to the current active project.																												
<linkName>	String	Name applied to the dynamic link. Optional, defaults to the designName.																												
<solutionName>	String	Solution in source design which will be referenced by the dynamic link. Optional, defaults to the first setup:sweep.																												
<transLinePort >	String	For HFSS designs only, port name for transmission lines. It is created only if it is <i>not</i> empty. Optional, defaults to “”. This value is ignored for non-HFSS design types.																												
<ReduceMatrix>	String	For Q3D and 2D extractor designs, reduce matrix which will be referenced by the dynamic link. This is ignored for other design types. Optional, defaults to “Original”.																												
<EnableHFSSCableModeling>	Boolean	For 2D extractor designs only, flag to create cable models. This is ignored for other designs types. Optional, defaults to false.																												
<Description>	String	Description of the dynamic link's source. Optional.																												
Return Value	None.																													

Python Syn-	AddDynamicLink (<designName>, <projectPath>, <linkName>, <solutionName>, <transLinePort>, <reducedMatrix> ,

tax	<code><enableCableModeling>, <description></code>
Python Example	<pre>oDesign = oProject.SetActiveDesign("Circuit1") oDesign.AddDynamicLink("TeeModel", "\$PersonalLib\\TeeWrongFile.aedt", "TeeLink", "Setup1 : LastAdaptive", "Port1", "", False, "desc")</pre>

VB Syntax	<pre>AddDynamicLink <designName>, <projectPath>, <linkName>, <solutionName>, <transLinePort>, <reducedMatrix>, <enableCableModeling>, <description></pre>
VB Example	<pre>Set oDesign = oProject.SetActiveDesign("Circuit1") oDesign.AddDynamicLink "TeeModel", "\$PersonalLib\\TeeWrongFile.aedt", "TeeLink", "Setup1 : LastAdaptive", "Port1", "", False, "desc"</pre>

AddModelingProperties

Use: Add a modeling property to a design

Command: None

Syntax: AddModelingProperties <design>

Return Value: None

Parameters: <design>

Type: string

VB Example: oDesign.AddModelingProperties <design>

Analyze

Solves a single solution setup and all of its frequency sweeps.

UI Access	Right-click a solution setup in the project tree, and select Analyze .		
Parameters	Name	Type	Description
	<setup>	String	Setup name.
	isBlocking	Boolean	An optional arg that defaults to <code>true</code> . If it's <code>false</code> , the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.
Return Value	Integer value: • 0 – Success • Else – Error		

Python Syntax	Analyze (<setup>)
Python Example	<code>oDesign.Analyze("Setup1", isBlocking)</code>

VB Syntax	Analyze <setup>
VB Example	<code>oDesign.Analyze "Setup1"</code>

AnalyzeAll [design]

Runs all solution setups and Optimetrics setups for the current design instance.

UI Access	N/A

Parameters	Name	Type	Description
	isBlocking	Boolean	An optional arg that defaults to <code>true</code> . If it's <code>false</code> , the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.
Return Value	None		
Python Syntax	<code>AnalyzeAll()</code>		
Python Example	<code>oDesign.AnalyzeAll(isBlocking)</code>		

VB Syntax	<code>AnalyzeAll</code>
VB Example	<code>oDesign.AnalyzeAll</code>

AnalyzeAllNominal

Runs all defined setups.

UI Access	Right-click Analysis in the project tree, and select Analyze All .		
Parameters	Name	Type	Description
	isBlocking	Boolean	An optional arg that defaults to <code>true</code> . If it's <code>false</code> , the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.
Return Value	None		

Python Syntax	AnalyzeAllNominal()
Python Example	<code>oDesign.AnalyzeAllNominal()</code>

VB Syntax	AnalyzeAllNominal
VB Example	<code>oDesign.AnalyzeAllNominal</code>

AnalyzeDistributed

Performs a distributed analysis.

UI Access	N/A		
Parameters	Name	Type	Description
	<SetupName>	String	String name of the setup to be analyzed.
Return Value	Integer value: • 0 – Success • Else – Error		

Python Syntax	AnalyzeDistributed(<SetupName>)
Python Example	<code>oDesign.AnalyzeDistributed('Setup1')</code>

VB Syntax	AnalyzeDistributed <SetupName>

VB Example	<code>oDesign.AnalyzeDistributed "Setup1"</code>

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.

UI Access	HFSS 3D Layout > Analysis Setup > Generate Mesh.		
Parameters	Name	Type	Description
	<SetupNameArray>	Array	Array containing the string names of setups.
Return Value	Integer value: • 0 – Success • Else – Error		

Python Syntax	<code>ApplyMeshOps(<SetupNameArray>)</code>
Python Example	<code>oDesign.ApplyMeshOps(['Setup1','Setup2'])</code>

VB Syntax	<code>ApplyMeshOps(<SetupNameArray>)</code>
VB Example	<code>status = oDesign.ApplyMeshOps Array("Setup1","Setup2")</code>

ChangeProperty[ReportSetup]

Change the properties for a Report.

UI Access	Double-click on a report to change its properties.		
Parameters	Name	Type	Description
	<PropertyArray>	Array	Varies, depending on the properties associated with the select object command.
Return Value	None.		

Python Syntax	ChangeProperty(<PropertyArray>)
Python Example	<pre>oModule = oDesign.GetModule("ReportSetup") oModule.ChangeProperty(["NAME:AllTabs", ["NAME:General", ["NAME:PropServers", "XY Plot 1:General"],],</pre>

```
[
  "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"
    ]
]
])

oModule.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:General",
        [
            "NAME:PropServers",
            "S Parameter Plot 1:General"
        ],
    ]
]
[
    "NAME:ChangedProps",
    [

```

```
        "NAME:Auto Scale Fonts",  
        "Value:=" , False  
    ]  
]  
]  
])
```

Example For Rectangular Stacked Plot

```
oProject = oDesktop.SetActiveProject("antenna_test")  
oDesign = oProject.SetActiveDesign("Design1")  
oModule = oDesign.GetModule("ReportSetup")  
oModule.ChangeProperty(  
    [  
        "NAME:AllTabs",  
        [  
            "NAME:Scaling",  
            [  
                "NAME:PropServers",  
                "S Parameter Plot 1:AxisY2:Stack1",  
                "S Parameter Plot 1:AxisY2:Stack2"
```

```
    ],  
    [  
        "NAME:ChangedProps",  
        [  
            "NAME:Min",  
            "Value:="          , "-60"  
        ]  
    ]  
]  
])
```

Example Change eye rect and eye plot:

```
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")  
oDesktop.RestoreWindow()  
oProject = oDesktop.SetActiveProject("test_nport_synth2")  
oDesign = oProject.SetActiveDesign("Circuit1")  
oModule = oDesign.GetModule("ReportSetup")  
oModule.ChangeProperty(  
    [  
        "NAME:AllTabs",  
        [  

```

```
"NAME:Grid",  
[  
  "NAME:PropServers",  
  "QuickEye Voltage Plot 1:Grid:Eye"  
],  
[  
  "NAME:ChangedProps",  
  [  
    "NAME:Major grid line style",  
    "Value:=" , "Dot"  
  ],  
  [  
    "NAME:Minor grid line style",  
    "Value:=" , "Dot"  
  ]  
]  
],  
[  
  "NAME:Axis",  
  [  
    "NAME:PropServers",
```

```
        "QuickEye Voltage Plot 1:AxisX:Eye"
    ],
    [
        "NAME:ChangedProps",
        [
            "NAME:Axis Color",
            "R:=" , 255,
            "G:=" , 128,
            "B:=" , 128
        ]
    ]
],
[
    "NAME:Axis",
    [
        "NAME:PropServers",
        "QuickEye Voltage Plot 1:AxisY1:Rect"
    ],
    [
        "NAME:ChangedProps",
```

```
[
    "NAME:Axis Color",
    "R:=" , 255,
    "G:=" , 128,
    "B:=" , 128
]
]
]
])
oModule.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:Grid",
        [
            "NAME:PropServers",
            "QuickEye Voltage Plot 1:Grid:Rect"
        ],
        [
            "NAME:ChangedProps",
            [
```


	<pre> "NAME:Major grid line style", "Value:=", "LongDash"], ["NAME:Minor grid line style", "Value:=", "LongDash"]]]]) </pre>

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:Show X Scrollbar", "Value:=", _ true)))) oModule.ChangeProperty Array("NAME:AllTabs", Array("NAME:General", Array("NAME:PropServ- ers", _ </pre>

```
"XY Plot 1:General"), Array("NAME:ChangedProps", Array("NAME:Use Scientific Notation",
"Value:=", _
true))))
```

CopyItemCommand

Use: Copy tree items, such as Altrasim Solution Setups in Nexxim, or Solve Setups and Frequency Sweeps in Ensemble.

Command: None

Syntax: CopyItemCommand <ItemPathList>

Return Value: None

Parameters: <ItemPathList>

Type: Array of strings

VB Example: oDesign.CopyItemCommand Array("Project1|Nexxim1|Analysis|DCAnalysis2")

CreateReport

Creates a new report with a single trace and adds it to the **Results** branch in the project tree.

UI Access	Right-click on Results > Create [Type] Report		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<ReportType>	String	Type of report.
			Possible values are: "Modal S Parameters" - Only for Driven Modal solution-type problems with ports. "Terminal S Parameters" - Only for Driven Terminal solution-type problems with ports.

			"Eigenmode Parameters" - Only for Eigenmode solution-type problems. "Fields" "Far Fields" - Only for problems with radiation or PML boundaries. "Near Fields" - Only for problems with radiation or PML boundaries. "Emission Test"
	<DisplayType>	String	Type of display. If ReportType is "Modal S Parameters", "Terminal S Parameters", or "Eigenmode Parameters", then set to one of the following: "Rectangular Plot", "Polar Plot", "Radiation Pattern", "Smith Chart", "Data Table", "3D Rectangular Plot", "3D Rectangular Bar Plot", "Rectangular Stacked Plot" or "3D Polar Plot". If <ReportType> is "Fields", then set to one of the following: "Rectangular Plot", "Polar Plot", "Radiation Pattern", "Rectangular Contour Plot", "Data Table", "3D Rectangular Plot", "Rectangular Stacked Plot" or "3D Rectangular Bar Plot". If <ReportType> is "Far Fields" or "Near Fields", then set to one of the following: "Rectangular Plot", "Radiation Pattern", "Rectangular Contour Plot", "Data Table", "3D Rectangular Plot", "3D Rectangular Bar Plot" "Rectangular Stacked Plot" or "3D Polar Plot" If <ReportType> is "Emission Test", then set to one of the following: "Rectangular Plot" or "Data Table"

	<SolutionName>	String	Name of the solution as listed in the Modify Report dialog box.
	<ContextArray>	Array	Context for which the expression is being evaluated. This can be an empty string if there is no context. <pre>Array("Domain:=", <DomainType>)</pre> <DomainType> ex. "Sweep" or "Time" <pre>Array("Context:=", <GeometryType>)</pre> <GeometryType> ex. "Infinite Spheren", "Spheren", "Polylinen"
	<FamiliesArray>	Array	Contains sweep definitions for the report. <pre>Array("<VariableName>:= ", <ValueArray>)</pre> <ValueArray> <pre>Array("All") or Array("Value1", "Value2", ... "Valuen")</pre> examples of <VariableName> "Freq", "Theta", "Distance"
	<ReportDataArray>	Array	This array contains the report quantity and X, Y, and (Z) axis definitions. <pre>Array("X Component:=", <VariableName>, "Y Component:=", <VariableName> <ReportQuantityArray>)</pre> <ReportQuantityArray> ex. <pre>Array("dB(S(Port1, Port1))")</pre>
Return Value	None.		

Python Syntax	<pre>CreateReport(<ReportName>, <ReportType>, <DisplayType>, <SolutionName>, <ContextArray>, <FamiliesArray>,</pre>

	<ReportDataArray>, <ExtendedInfo>)
Python Example	Three examples are given below: (nominal, Optimetrics, spectral) Nominal Example: <pre>oModule.CreateReport("XY Plot 2", "Standard", "Rectangular Plot", "TR", ["NAME:Context", "SimValueContext:=", [], 1, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0]], ["Time:=" , ["All"], "aaa:=" , ["Nominal"]], ["X Component:=", "Time", "Y Component:=", ["E1.I"]], [])</pre> Optimetrics Example: <pre>oModule.CreateReport ("XY Plot 3", "Standard", "Rectangular Plot", "TR", ["NAME:Context", "OptiSetup:=", "ParametricSetup1", "SimValueContext:=", [1, 0, 2, 0, false, false, 0, 1, 0, 1, 1, "", 0, 0]])</pre>

```

],
[
    "Time:=", ["All"], "aaa:=", ["Nominal"]
],
[
    "X Component:=", "Time",
    "Y Component:=", ["E1.I"]
],
[])

```

Spectral Example:

```

oModule.CreateReport ("XY Plot 4", "Standard",
    "Rectangular Plot", "TR",
    [
        "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, "40ms", "TH", false,
            "40", "TS", false, "0ns", "UF", false, "0",
            "WT", false, "0", "WW", false, "100"
        ]
    ],
    [
        "Spectrum:=", ["All"], "aaa:=", ["Nominal"]
    ],
    [
        "X Component:=", "Spectrum",
        "Y Component:=", ["mag(E1.I)"]
    ],
    [])

```

Partial Discharge Example

```
oModule.CreateReport("Partial Discharge Plot 2",  
    "Partial Discharge", "Rectangular Plot",  
    "PartialDischarge1 : PartialDischarge", [],  
    [  
        "GasPressure:=", ["All"],  
        "Freq:="          , ["All"],  
        "bend_angle:="    , ["All"]  
    ],  
    [  
        "X Component:=", "GasPressure",  
        "Y Component:=", ["Power"]  
    ]  
))
```

Example 3D Rectangular Bar Plot with Change Bar properties

```
oModule = oDesign.GetModule("ReportSetup")  
oModule.CreateReport("S Parameter Plot 4", "Modal Solution Data",  
    "3D Rectangular Bar Plot", "Setup1 : Sweep", [],  
    [  
        "Freq:="          , ["All"],
```

```

        "UU:="          , ["All"],
        "ZZZ:="         , ["Nominal"]
    ],
    [
        "X Component:=", "Freq",
        "Y Component:=", "UU",
        "Z Component:=", ["dB(S(wDipole1_1_p1,wDipole1_1_p1))"]
    ])
oModule.ChangeProperty(
    [
        "NAME:AllTabs",
        [
            "NAME:Bar",
            [
                "NAME:PropServers"
                "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
            ],
            [
                "NAME:ChangedProps",
                [
                    "NAME:Transparency",
                    "Value:=", "0.5"
                ]
            ]
        ]
    ]
)

```



```
angedProps",
    [
        "NAME:Filled",
        "Value:=", False
    ]
]
])

oModule.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:Bar",
        [
            "NAME:PropServers",
            "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Filled",
                "Value:=", True
            ]
        ]
    ]
]
```

```

    ]
  ]
])

oModule.ChangeProperty(
[
  "NAME:AllTabs",
  [
    "NAME:Bar",
    [
      "NAME:PropServers",
      "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
    ],
    [
      "NAME:ChangedProps",
      [
        "NAME:Bar Width",
        "Value:=", "Narrow"
      ]
    ]
  ]
])

oModule.ChangeProperty(
[
  "NAME:AllTabs",
  [
    "NAME:Bar",
    [
      "NAME:PropServers",
      "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
    ],
    [

```


VB Syntax	CreateReport <ReportName>, <ReportType>, <DisplayType>, <SolutionName>, <ContextArray>, <FamiliesArray>, <ReportDataArray>, <ExtendedInfo
VB Example	<pre>oModule.CreateReport "3D Cartesian Plot1", "Far Fields", _ "3D Cartesian Plot", "Setup1 : LastAdaptive", _ Array("Context:=", "Infinite Spherel", "Domain:=", "Sweep"), _ Array("Theta:=", Array("All"), "Phi:=", Array("All"), _ "Freq:=", Array("10GHz")), _ Array("X Component:=", "Theta", _ "Y Component:=", "Phi", _ "Z Component:=", Array("rETotal")), _ Array() oModule.CreateReport "ReptSmithFreq", _ "Modal Solution Data", "Smith Plot", "Setup1 : Sweep1", _ Array("Domain:=", "Sweep"), _ Array("Freq:=", Array("All")), _ Array("Polar Component:=", _ Array("ln(Y(LumpPort1,LumpPort1))")), _ Array() oModule.CreateReport "Rectangular Contour Plot 2", "Far Fields", _ "Rectangular Contour Plot", "Setup1 : LastAdaptive", Array("Context:=", _ "Infinite Spherel"), Array("_u:=", Array("All"), "_v:=", Array("All"), "Freq:=", Array(_</pre>

```

"5GHz")), Array("X Component:=", "_u", "Y Component:=", "_v", "Z Component:=", Array
(
  _rETotal")), Array()

oModule.CreateReport "Rept2DRectFreq", _
  "Modal Solution Data", "XY Plot", _
  "Setup1 : Sweep1", _
  Array("Domain:=", "Sweep"), _
  Array("Freq:=", Array("All")), _
  Array("X Component:=", "Freq", _
  "Y Component:=", _ Array("dB(S(LumpPort1,LumpPort1))")), _
  Array()

```

DeleteDataset

Deletes a specified dataset. This can be executed by the oProject, or oDesign variables.

UI Access	Project > Datasets > Remove.		
Parameters	Name	Type	Description
	<DatasetName>	String	Name of the dataset found in the project.
Return Value	None.		

Python Syntax	DeleteDataset (<DatasetName>)
Python Example	oProject.DeleteDataset('\$ds1')
	oDesign.DeleteDataset('\$ds1')

VB Syntax	DeleteDataset <DatasetName>
VB Example	<pre>oProject.DeleteDataset "\$ds1" oDesign.DeleteDataset "\$ds1"</pre>

DeleteDesignInstance

Use: Delete the design instance

Command: None

Syntax: DeleteDesignInstance <instance_name>

Return Value: None

Parameters: <instance_name>

Type: string

Python Syntax	DeleteDesignInstance(<instanceName>)
Python Example	<pre>oProject.DeleteDesignInstance("Circuit1")</pre>

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

	<i><variationKeys></i>	Array	Array containing either "All" string to select all variations, or strings of variations to include.
	<i><deleteMesh></i>	Boolean	When true, deletes mesh data.
	<i><deleteLinkedData></i>	Boolean	When true, deletes linked data.
Return Value	None.		

Python Syntax	DeleteFieldVariation(<i><variationKeys></i> , <i><deleteMesh></i> , <i><deleteLinkedData></i>)		
Python Example	<pre>oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() Design.DeleteFieldVariation(['All'], True, False)</pre>		

VB Syntax	DeleteFieldVariation <i><variationKeys></i> , <i><deleteMesh></i> , <i><deleteLinkedData></i>		
VB Example	<pre>Set oProject = oDesktop.SetActiveProject "OptimTee" Set oDesign = oProject.SetActiveDesign "TeeModel" oDesign.DeleteFieldVariation Array("offset=" & Chr(39) & "0.0947046688081817in" & Chr(39) & ""), true, false</pre>		

DeleteFullVariation

Deletes either all solution data, or selected variation data.

UI Access	[HFSS 3D Layout] > Results > Clean Up Solutions.		
Parameters	Name	Type	Description
	<i><variationKeys></i>	Array	Array containing either "All" string to select all variations, or strings of vari-

			ations to include.
	<code><deleteLinkedData></code>	Boolean	When true, deletes linked data as well.
Return Value	None.		

Python Syntax	<code>DeleteFullVariation(<variationKeys>, <deleteLinkedData>)</code>
Python Example	<code>oDesign.DeleteFullVariation(['All'], false)</code>

VB Syntax	<code>DeleteFullVariation <variationKeys>, <deleteLinkedData></code>
VB Example	<code>oDesign.DeleteFullVariation Array('All'), false</code>

DeleteOutputVariable

Deletes an existing output variable. The variable can only be deleted if it is not in use by any traces.

UI Access	HFSS 3D Layout > Results > Output Variables. In the Output Variables window, click Delete .		
Parameters	Name	Type	Description
	<code><OutputVarName></code>	String	Name of the output variable.
Return Value	None.		

Python Syntax	<code>DeleteOutputVariable (<OutputVarName>)</code>

Python Example	<pre>oModule = oDesign.GetModule("OutputVariable") oModule.DeleteOutputVariable ("testNew")</pre>

VB Syntax	DeleteOutputVariable <OutputVarName>
VB Example	<pre>Set oModule = oDesign.GetModule("OutputVariable") oModule.DeleteOutputVariable "testNew"</pre>

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.

UI Access	Right-click on Results , select Browse Solutions... , click Delete button in the dialog.		
Parameters	Name	Type	Description
	<SoluParams>	Array	Structured array. Array(<DataSpecifierArray>, ...)
	<DataSpecifierArray>	Array	Structured array. Array(<DesignVariationKey>, <SetupName>, <Sol-nName>)
Return Value	None.		

Python Syntax	DeleteSolutionVariation(<SoluParams>)
Python Example	oModule.DeleteSolutionVariation([

	<pre>["width='2in'", "Setup1", "Adaptive_1"], ["width='2in'", "Setup1", "Sweep1"]]</pre>

VB Syntax	DeleteSolutionVariation <SoluParams>
VB Example	<pre>oModule.DeleteSolutionVariation Array(_ Array("width='2in'", "Setup1", "Adaptive_1"), _ Array("width='2in'", "Setup1", "Sweep1"))</pre>

DistributeAnalysis

Performs a distributed analysis.

UI Access	N/A		
Parameters	Name	Type	Description
	<SetupName>	String	String name of the setup to be analyzed.
Return Value	Integer value: • 0 – Success • Else – Error		

Python Syntax	DistributeAnalysis(<SetupName>)

Python Example	<code>oDesign.DistributeAnalysis('Setup1')</code>

VB Syntax	<code>DistributeAnalysis <SetupName></code>
VB Example	<code>oDesign.DistributeAnalysis "Setup1"</code>

EditCoSimulationOptions

Sets options for cosimulation.

Command: None

Syntax: EditCoSimulationOptions <array_name>

Return Value: None

Parameters: <array_name>

Type: string

```
oDesign.EditCoSimulationOptions Array("NAME:CoSimOptions",  
"Override:=", true, _  
"Setup:=", "Setup 1", _  
"OverrideSweep:=", true, _  
"Sweep:=", "Sweep 1", _  
"SweepType:=", 4, _  
"Interpolate:=", false, _  
"YMatrix:=", true, _  
"AutoAlignPorts:=", false, _
```

```
"InterpAlg:=", "auto")
```

EditDataset

Modifies a dataset. This can be executed by the oProject, or oDesign variables.

UI Access	Project > Datasets > Edit.		
Parameters	Name	Type	Description
	<OriginalName>	String	Name of the original dataset.
	<DatasetDataArray>	Array	Data for the modified dataset.
Return Value	None.		

Python Syntax	EditDataset (<OriginalName> <DatasetDataArray>)
Python Example	<pre>oProject.EditDataset ("ds1" ["NAME:ds2", ["NAME:Coordinates", ["NAME:Coordinate", "X:=", 1, "Y:=", 2], ["NAME:Coordinate",</pre>

```
        "X:=", 3, "Y:=", 4
    ]
]
)
oDesign.EditDataset ("ds1"
["NAME:ds2",
    ["NAME:Coordinates",
        [
            "NAME:Coordinate",
            "X:=", 1, "Y:=", 2
        ],
        [
            "NAME:Coordinate",
            "X:=", 3, "Y:=", 4
        ]
    ]
]
)
```

VB Syntax**EditDataset <OriginalName> <DatasetDataArray>**

VB Example	<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>

EditImportData

Use: Edit an imported solution

Command: None

Syntax: EditImportData <oldname> <newlink> <newpath> <newname> <data> <file>

Return Value: None

Parameters: <oldname>

Type: string

<newlink>

<newpath>

Type: string

<newname>

Type: string

<data>

Type: array

<file>

Type: string

VB Example: oDesign.EditImportData <oldname> <newlink> <newpath> <newname> <data> <file>

EditInfiniteArray

Edits the properties of an infinite array

Command: None

Syntax: EditInfiniteArray <array_name>

Return Value: None

Parameters: <array_name>

Type: string

VB Example: oDesign.EditInfiniteArray <array_name>

Edit (Layout Editor)

Use: Causes modification of one or more existing object(s).

Syntax: **Edit** <Array("NAME:items".....)>,

Parameters: <Array("NAME:items"
<edit_object_info>, // one object info

<edit_object_info>, // another one

...) // etc

<edit_object_info>:

```
Array("NAME:item",  
"name:=", <object_name>, // name of the object  
<object_description>) // 'new' object state, type should be consistent with <object_name>:
```

```
<object_description>:  
<circle_description> |  
<rectangle_description> |  
<line_description> |  
<polyon_description> |  
<text_description> |  
<circle_void_description> |  
<rectangle_void_description> |  
<line_void_description> |  
<polyon_void_description> |  
<component_description> |  
<pin_description> |  
<via_description>
```

VB Example:

```
oEditor.Edit Array("NAME:items", Array("NAME:item",  
"name:=", "circle_0", Array("NAME:contents",
```

```
"circleGeometry:=", Array("Layer:=", 6,
"Name:=", "circle_0", "LayerName:=", "Top",
"lw:=", "0mm", "x:=", "-0.008meter",
"y:=", "13.2924281984334mm", "r:=", 10.2924281984334mm)))
```

EditLayoutForLayoutComponent

Opens the definition component for editing in HFSS 3D Layout for a Layout Component in an HFSS 3D design.

UI Access	Edit Layout...		
Parameters	Name	Type	Description
	<ComponentDefinitionName>	String	Name of the Layout Component Definition.
Return Value	None.		

Python Syntax	EditLayoutForLayoutComponent ([Name:LayoutComponent EDB", Definitions:=", [<ComponentDefinitionName>])
Python Example	<pre>oDesign.EditLayoutForLayoutComponent (["NAME:Layout Component EDB", "Definitions:=", ["Diff_Via_diffViaNominal"]]) </pre>

EditNotes

Updates the notes for the design.

UI Access	HFSS 3D Layout > Edit Notes.		
Parameters	Name	Type	Description
	<DesignNotes>	String	New design notes that are applied, replacing any current notes.
Return Value	None.		

Python Syntax	EditNotes(<DesignNotes>)
Python Example	<code>oDesign.EditNotes("My design notes.")</code>

VB Syntax	EditNotes <DesignNotes>
VB Example	<code>oDesign.EditNotes "My design notes."</code>

EditNotes (Layout Editor)

Edits the notes of a design

Command: None

Syntax: EditNotes <design_name>

Return Value: None

Parameters: <design_name>

Type: string

VB Example: oDesign.EditNotes <design_name>

EditOptions (Layout Editor)

Edits the properties of an infinite array

Command: None

Syntax: EditOptions <array_name>

Return Value: None

Parameters: <array_name>

Type: string

VB Example: oDesign.EditOptions <array_name>

EditOutputVariable

Changes the name or expression of an existing output variable.

UI Access	N/A		
Parameters	Name	Type	Description
	<OrigVarName>	String	Name of the original output variable.
	<NewExpression>	String	New value to assign to the variable.
	<NewVarName>	String	New name of the variable if any, else pass empty string.
	<SolutionName>	String	Name of the solution as seen in the output variable UI. For example, "Setup1 : Last Adaptive".
	<ReportType> or <SolutionType>	String	The name of the report type as seen in the output variable UI.

			For Layout Editor, use the solution type.
	<ContextArray> or <DomainArray>	Array	Structured array containing context for which the output variable expression is being evaluated. Array("Context:=", <string>) For Layout Editor, use Domain array: Array("Domain:=", <string>)
Return Value	None		

Python Syntax	<code>EditOutputVariable (<OrigVarName>, <NewExpression>, <NewVarName>, <SolutionName>, <ReportType SolutionType>, <ContextArray DomainArray>)</code>
Python Example	<pre>oModule = oDesign.GetModule("OutputVariable") oModule.EditOutputVariable ("test", "normalize(R1_0.V)", "testNew", "TR", "Standard", [])</pre>

VB Syntax	<code>EditOutputVariable <OrigVarName>, <NewExpression>, <NewVarName>, <SolutionName>, <ReportType SolutionType>, <ContextArray DomainArray></code>
VB Example	<pre>Set oModule = oDesign.GetModule("OutputVariable") oModule.EditOutputVariable "test", "dB(S(WavePort1,WavePort1)) ", "testNew", "Setup1 : LastAdaptive", "Modal Solution Data", Array("Domain:=", "Sweep")</pre>

EMDesignOptions

Use: Set options for an EM Design.

Command: Right click on design and select **EM Design Options**

Syntax: DesignOptions <Options Array>

Return Value: None

Parameters:

<Options Array>

```
Array("NAME:options",  
"SaveSolFilesAsBinary:=", <boolean>,  
"LowPriorityForSimulations:=", <boolean>,  
"SaveNearFieldSolutions:=", <boolean>,  
"SchematicEnabled:=", <boolean>,  
"UseGlobalNumProc:=", <boolean>,  
"ComputeBothEvenAndOddCPWModes:=", <boolean>,  
"NumProcessors:=", <int>,  
"NumProcessorsDistrib:=", <int>,  
"CausalMaterials:=", <boolean>,  
"UseHPCForMP:=", <boolean>,  
"HPCLicenseType:=", <int>)
```

SaveSolFilesAsBinary - if true, solutions files are saved using a binary format.

LowPriorityForSimulations - if true, run simulations at a lower CPU priority.

SaveNearFieldSolutions - if true, save near field solutions

SchematicEnabled - if true, enable schematics

UseGlobalNumProc - if true, use global number of processors and ignore NumProcessors

ComputeBothEvenAndOddCPWModes - if true, compute both even and odd cpw modes

NumProcessors - number of processors

NumProcessorsDistrib- number of distributed processors

CausalMaterials - if true, use causal materials

UseHPCForMP - if true, use hpc for mp

HPCLicenseType - number indicating hpc license type

VB Example:

```
oDesign.DesignOptions Array("NAME:options",  
"SaveSolFilesAsBinary:=", true,  
"LowPriorityForSimulations:=", false,  
"SaveNearFieldSolutions:=", false,  
"SchematicEnabled:=", true,  
"UseGlobalNumProc:=", true,  
"ComputeBothEvenAndOddCPWModes:=", false,  
"NumProcessors:=", 1,
```

```
"NumProcessorsDistrib:=", 1,
"CausalMaterials:=", true,
"UseHPCForMP:=", true,
"HPCLicenseType:=", 1)
```

ExportCircuit (Layout Editor)

Exports lumped RLGC data.

UI Access	From the Project Manager window, right-click a setup and select Properties .		
Parameters	Name	Type	Description
	<Solution>	String	The precise name of the solution from which to export data.
	<SaveLocation>	String	User-selected directory path to save the exported data.
	<CircuitDataArray>	Array	An array that contains these parameters in order: • <NAME:CircuitData> • <MatrixName> • <NumberOfCells> • <UserHasChangedSettings> • <IncludeCap> • <IncludeCond> • <CouplingLimitsArray> • <IncludeDCR> • <IncludeDCL> • <IncludeACR> • <IncludeACL> • <AddResistance> • <IncludeCPP>

		• <PackageArray> This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.
<NAME:CircuitData>	String	Identifies the contents of the array (i.e., CircuitData).
<MatrixName>	String	Original, to export the solved matrix data, or enter the name of any reduced matrices, if available.
<NumberOfCells>	Integer	Number of cells in export.
<UserHasChangedSettings>	Boolean	True – export data with the user's chosen settings False – export data with the default settings
<IncludeCap>	Boolean	True – exports capacitance matrix False – does not export capacitance matrix
<IncludeCond>	Boolean	True – exports conductance matrix False – does not export conductance matrix
<CouplingLimitsArray>	Array	An array that contains these parameters in order: • <NAME:CouplingLimits> • <CouplingLimitType> This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.
<NAME:CouplingLimits>	String	Identifies the contents of the array (i.e., CouplingLimits).
<CouplingLimitType>	String	ByFraction or ByValue
<IncludeDCR>	Boolean	True – exports DC resistance matrix False – does not export DC resistance matrix
<IncludeDCL>	Boolean	True – exports DC inductance matrix False – does not export DC inductance matrix
<IncludeACR>	Boolean	True – exports AC resistance matrix False – does not export AC resistance matrix
<IncludeACL>	Boolean	True – exports AC inductance matrix False – does not export AC inductance matrix
<ADDResistance>	Boolean	True – adds DC and AC resistance

			False – does not add the DC and AC resistance Note: You cannot export both AC and DC matrices unless AddResistance is selected.
	<IncludeCPP>	Boolean	True – exports data with Chip Package Protocol active False – does not export data with Chip Package Protocol active
	<RLGCCPPInfoArray>	Array	An array that contains these parameters in order: • <NAME:RLGCCPPInfo> • <PackageType> • <LengthUnits> • <NAME:Pins> This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.
	<NAME:RLGCCPPInfo>	String	Identifies the contents of the array (i.e., RLGCCPPInfo).
	<PackageType>	String	wirebond dieup, wirebond diedown, flipchip dieup, or flip-chip diedown
	<LengthUnits>	String	A unit of measurement: um , mm , or mil .
	<NAME:Pins>	Array	Identifies the contents of the array (i.e., Pins).
	<Name>	String	The user-entered name of the model.
	<Freq>	Integer	The frequency, in Hz.
Return Value	None		

Python Syntax	oDesign.ExportCircuit(<Solution>, <SaveLocation>, [<CircuitData>, <MatrixName>, <NumberOfCells>, <User-HasChangedSettings>, <IncludeCap>, <IncludeCond>, [<NAME:CouplingLimits>, <CouplingLimitType>], <IncludeDCR>, <IncludeDCL>, <IncludeACR>, <IncludeACL>, <ADDResistance>, <IncludeCPP>,

	[<NAME:RLGCCPPInfo>, <PackageType>, <LengthUnits>, [<NAME:Pins>]], <Name>, <Freq>)
Python Example	<pre>oDesign.ExportCircuit("Q3D1 : Last Adaptive", "", "C:/Projects/Q3D/Example.cir", ["NAME:CircuitData", "MatrixName:=" , "Original", "NumberOfCells:=" , "1", "UserHasChangedSettings:=", True, "IncludeCap:=" , False, "IncludeCond:=" , False, ["NAME:CouplingLimits", "CouplingLimitType:=" , "None"], "IncludeDCR:=" , True, "IncludeDCL:=" , True, "IncludeACR:=" , False, "IncludeACL:=" , False, "ADDResistance:=" , False, "IncludeCPP:=" , False, [</pre>

	<pre> "NAME:RLGCCPPInfo", "PackageType:=" , "wirebond dieup", "LengthUnits:=" , "mm", ["NAME:Pins"]]], "Example", 100000000) </pre>

VB Syntax	<pre> oDesign.ExportCircuit <Solution>, <SaveLocation>, (<CircuitData>, <MatrixName>, <NumberOfCells>, <User-HasChangedSettings>, <IncludeCap>, <IncludeCond>, (<NAME:CouplingLimits>, <CouplingLimitType>), <IncludeDCR>, <IncludeDCL>, <IncludeACR>, <IncludeACL>, <ADDResistance>, <IncludeCPP>, (<NAME:RLGCCPPInfo>, <PackageType>, <LengthUnits>, (<NAME:Pins>))), <Name>, <Freq> </pre>
VB Example	<pre> oDesign.ExportCircuit "Q3D1 : Last Adaptive", "", "C:/Projects/Q3D/Example.cir", Array ("NAME:CircuitData", "MatrixName:=", "Original", "NumberOfCells:=", "1", "UserHasChangedSettings:=", True, "IncludeCap:=", False, "IncludeCond:=", False, Array("NAME:CouplingLimits", "CouplingLimitType:=", "None"), "IncludeDCR:=", True, "IncludeDCL:=", True, "IncludeACR:=", False, "IncludeACL:=", False, "ADDResistance:=", False, "IncludeCPP:=", False, Array("NAME:RLGCCPPInfo", "PackageType:=", "wirebond dieup", "LengthUnits:=", "mm", Array("NAME:Pins"))), "Example", 100000000 </pre>

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.
	<ProbType>	String	<i>Optional.</i> For Q3D Extractor designs. Valid options are "CG", "DC RL", and "AC RL".
Return Value	None.		

Python Syntax	ExportConvergence(<Setup>, <VariationKeys>, <FilePath> <ProbType>)
Python Example	<pre>oDesign.ExportConvergence('Setup1', 'x_size = 2mm', 'C:/Project12_Q3DDesign1.conv', 'CG')</pre>

VB Syntax	ExportConvergence <Setup>, <VariationKeys>, <FilePath> <ProbType>
VB Example	<pre>oDesign.ExportConvergence "Setup1", "x_size = 2mm", "CG", "C:/Project12_Q3DDesign1.conv", "CG"</pre>

ExportDataset

Exports a dataset to a named file. This can be executed by the oProject, or oDesign variables.

UI Access	Project > Datasets > Export.		
Parameters	Name	Type	Description
	<datasetFileFullPath>	String	The full path to the file.
Return Value	None.		

Python Syntax	ExportDataset (<datasetFileFullPath>)
Python Example	<pre>oProject.ExportDataset('e:/tmp/dsdata.txt') oDesign.ExportDataset('e:/tmp/dsdata.txt')</pre>

VB Syntax	ExportDataset <datasetFileFullPath>
VB Example	<pre>oProject.ExportDataset "e:/tmp/dsdata.txt" oDesign.ExportDataset "e:/tmp/dsdata.txt"</pre>

ExportForHSpice

Exports matrix solution data to a file in a format suitable for HSpice analysis. Available only for Driven Terminal solution types with ports. Output in an appropriate format will be generated for each of the non-empty file names provided.

UI Access	N/A

Parameters	Name	Type	Description
	<DesignVariation>	String	Design variation key.
	<SoluSelections>	Array	Array of selected solutions.
	<SpiceType>	Integer	1 - HSpice.
	<BandWidth>	Integer	0 - Low (narrow) band width.
	<FWSFile>	String	Full wave spice file name.
	<LumpedElementFile>	String	Lumped element file name.
	<PoleZeroSpiceFile>	String	Pole zero spice file name.
	<PoleZeroMatlabFile>	String	Pole zero Matlab file name.
	<PartialFractionFile>	String	Partial fraction file name.
	<FittingError>	Double	The accuracy to use in fitting the pole zero model.
	<MinimumOrder>	Integer	Minimum number of poles in rational function expansion.
	<MaximumOrder>	Integer	Maximum number of poles in rational function expansion.
	<UseCommonGround>	Integer	Optional. 1 to use common ground, 0 otherwise.
	<EnforcePassivity>	Integer	Optional. 1 to enforce passivity, 0 otherwise.
Return Value	None.		
Python Syntax	ExportForHSpice(<DesignVariation>, <SoluSelections>, <SpiceType>, <BandWidth>, <FWSFile>, <LumpedElementFile>, <PoleZeroSpiceFile>, <PoleZeroMatlabFile>, <PartialFractionFile>, <FittingError>, <MinimumOrder>, <MaximumOrder>, [Optional <UseCommonGround>], [Optional <EnforcePassivity>])		
Python Example	<pre>oModule.ExportForHSpice("width='2in'", ["Setup1:Sweep1"], 1, 0, "c:\mydir\Sweep1.fws", "", "", "", "", .005, 20, 200)</pre>		
VB Syntax	ExportForHSpice <DesignVariation>, <SoluSelections>, <SpiceType>, <BandWidth>, <FWSFile>, <LumpedElementFile>, <PoleZeroSpiceFile>, <PoleZeroMatlabFile>, <PartialFractionFile>, <FittingError>,		

	<code><MinimumOrder>, <MaximumOrder>, [Optional <UseCommonGround>], [Optional <EnforcePassivity>]</code>
VB Example	<pre>oModule.ExportForHSpice "width='2in'", _ Array("Setup1:Sweep1"), 1, 0, _ "c:\mydir\Sweep1.fws", "", "", "", "", _ .005, 20, 200</pre>

ExportForHSpice (Layout Editor)

Use: Export matrix solution data to an HSpice subcircuit.

Command: None

Syntax: ExportForHSpice <SolutionVariationKey> <SolnSelectionArray> <Spice Type> <Bandwidth> <FullWaveSpiceFilename> <LumpedElementFilename> <Unused> <Unused> <PartialFractionFilename> <Fitting Error> <Unusedint> <MaxOrder> <UseCommonGround> <DoPassivityCheck>

Return Value: None

Parameters: <SolutionVariationKey>

Type: string

<SolnSelectionArray>

Type: array

<Spice Type: 1=HSPICE>

Type: int

<Bandwidth: 0=Low Bandwidth, 1=Broad Bandwidth>

Type: int

<FullWaveSpiceFilename>

Type: string

<LumpedElementFilename>

Type: string

<Unused>

Type: string

<Unused>

Type: string

<PartialFractionFilename>

Type: string

<Fitting Error>

Type: double

<Unusedint>

Type: int

<MaxOrder>

Type: int

<UseCommonGround [Default: FALSE]>

Type: boolean

<DoPassivityCheck [Default: FALSE]>

Type: boolean

VB Example: oDesign.ExportForHSpice "", Array("Setup 1:Sweep 1"), 1, 0, "C:/Projects/MyTRL_
fws.sp", "", "", "", "", 0.5, 0, 200, 0, 0

ExportForSpice (Layout Editor)

Use: Export matrix solution data to a file in the given spice format.

Command: None

Syntax: ExportForSpice <SolutionVariationKey> <SolnSelectionArray> <Spice Type> <Bandwidth> <FullWaveSpiceFilename> <LumpedElementFilename> <Unused> <Unused> <PartialFractionFilename> <FittingError> <MaxOrder> <UseCommonGround> <DoPassivityCheck>

Return Value: None

Parameters: <SolutionVariationKey>

Type: string

<SolnSelectionArray>

Type: array

<Spice Type: 0=PSpice, 2=MaxwellSpice, 3=Spectre>

Type: int

<Bandwidth: 0=Low Bandwidth, 1=Broad Bandwidth>

Type: int

<FullWaveSpiceFilename>

Type: string

<LumpedElementFilename>

Type: string

<Unused>

Type: string

<Unused>

Type: string

<PartialFractionFilename>

Type: string

<FittingError [Default:0.5]>

Type: double

<MaxOrder [Default:200]>

Type: int

<UseCommonGround [Default:FALSE]>

Type: boolean

<DoPassivityCheck [Default:FALSE]>

Type: boolean

```
VB Example: oDesign.ExportForSpice "", Array("LNA:LNA"), 0, 0, _  
"C:/Documents and Settings/Marcia.BMT/My Documents/Code Modifications/Test" & _  
" Designs/s-params_fws_fws.lib", _  
"C:/Documents and Settings/Marcia.BMT/My Documents/Code Modifications/Test" & _  
" Designs/s-params_fws_lfws.lib", "", "", "", 0.52, 210, 1, 1
```

ExportMeshStats

Exports mesh statistics to a file.

UI Access	N/A

Parameters	Name	Type	Description
	<setupName>	String	Name of the solution setup.
	<variationString>	String	Name of the variation. An empty string is interpreted as the current nominal variation.
	<filePath>	String	Full file path, including extension *.ms.
	<overwriteIfExists>	Boolean	(Optional). Defaults to True. If True, script overwrites any existing file of the same name. If False, no overwrite.
Return Value	None.		

Python Syntax	ExportMeshStats(<setupName>, <variationString>, <filePath>, <overwriteIfExists>)
Python Example	oDesign.ExportMeshStats ("Setup1", "radius=3mm", "C:\mydir\meshstats.ms", True)

VB Syntax	ExportMeshStats <setupName>, <variationString>, <filePath> <overwriteIfExists>
VB Example	oDesign.ExportMeshStats "Setup1", "radius=3mm", "C:\mydir\meshstats.ms", true

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 syntax, but ignored if DoRenorm is false.
<DataType>	String	Optional. Type: "S", "Y", or "Z". The matrix to export.
<pass>	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.
<ComplexFormat>	Integer	Optional. Type: "0", "1", or "2" The format to use for the exported data. 0 = Magnitude/Phase. 1 = Real/Imaginary.

			2= db/Phase.
	<Precision>	Integer	Optional. Touchstone number of digits precision. Default if not specified is 15.
	<UseExportFreqs>	Boolean	Specifies whether to use export frequencies.
	<IncludeGammaComments>	Boolean	Specifies whether to include Gamma and Impedance comments.
	<SupportNonStdExport>	Boolean	Specifies whether to support non-standard Touchstone extensions for mixed reference impedance.
Return Value	None.		

Python Syntax	ExportNetworkData(<DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>, <FreqsArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <SupportNonStdExport>])
Python Example	oModule.ExportNetworkData("", ["Setup1:Sweep1"], 3, "C://Documents/package_ HFSSDesign1.s4p", ['all'], True, 50, "S", -1, 0, 15, True, True, True

VB Syntax	ExportNetworkData <DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>, <FreqsArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <SupportNonStdExport>]
VB Example	oModule.ExportNetworkData "width='2in'", _ Array("Setup1:Sweep1"),2, "c:\mydir\out.tab", _ Array("all"), false, 0 oModule.ExportNetworkData "width='2in'", _

	<pre>Array("Setup1:Sweep1", "Setup1:Sweep2"), 3, _ "c:\mydir\out.s2p", Array(1.0e9, 1.5e9, 2.0e9), _ true, 50.0</pre>

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. <pre>Array(<SolnSelector>, <SolnSelector>, ...)</pre> 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 syntax, but ignored if DoRenorm is false.
	<DataType>	String	Optional. Type: "S", "Y", or "Z". The matrix to export.
	<pass>	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.
	<ComplexFormat>	Integer	Optional. Type: "0", "1", or "2" The format to use for the exported data. 0 = Magnitude/Phase. 1= Real/Imaginary. 2= db/Phase.
	<Precision>	Integer	Optional. Touchstone number of digits precision. Default if not specified is 15.
	<UseExportFreqs>	Boolean	Specifies whether to use export frequencies.
	<IncludeGammaComments>	Boolean	Specifies whether to include Gamma and Impedance comments.
	<SupportNonStdExport>	Boolean	Specifies whether to support non-standard Touchstone extensions for mixed reference impedance.
Return Value	None.		

Python Syntax	ExportNetworkData(<DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>, <FreqsArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <SupportNonStdExport>])

Python Example	<pre>oModule.ExportNetworkData("", ["Setup1:Sweep1"], 3, "C://Documents/package_ HFSSDesign1.s4p", ['all'], True, 50, "S", -1, 0, 15, True, True, True</pre>

VB Syntax	ExportNetworkData <DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>, <FreqsArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <SupportNonStdExport>]
VB Example	<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

Exports s-parameters in neutral file format (*.nmf).

UI Access	Results > Solution Data. On the Matrix tab, click Export tab. Then click Export > S-parameter . Select Neutral Model format and export.		
Parameters	Name	Type	Description
	<sourceName>	Array	String names of solutions. Use more than one array entry for a com-

			binned Interpolating sweep.
	<code><outputFileName></code>	String	Full path of output file, including *.nmf extension.
	<code><reduceMat></code>	String	Either "Original" or name of reduced matrix.
	<code><refImp></code>	Double	Reference impedance.
	<code><freqs></code>	Array	Either "All" or array of doubles representing frequencies.
	<code><variationKey></code>	String	Design variation key. Pass empty string for the current nominal variation.
	<code><displayFormat></code>	String	"MagPhase", "Reallmag", "DbPhase".
	<code><pass></code>	Integer	Optional. The pass to export. This is ignored if the <i>sourceName</i> is a frequency sweep. Leaving out this value or specifying -1 gets all passes.
	<code><ComplexFormat></code>	Integer	Optional. The format to use for the exported data. Options are 0 = Magnitude/Phase 1= Real/Imaginary. 2= db/Phase.
	<code><Precision></code>	Integer	Optional. Number of digits precision. If not specified, the default is 6.
Return Value	None.		

Python Syntax	<code>ExportNMFData (<sourceName>, <outputFileName>, <reduceMat>, <refImp>, <freqs>, <variationKey>, <displayFormat>, <pass>, <ComplexFormat>, <Precision>)</code>
Python Example	<pre>oDesign.ExportNMFData("Setup2 : LastAdaptive", "C:/outputs/data.nmf", "Original", 50, [10000000, 20000000, 30000000, 40000000, 50000000, 60000000, 70000000, 80000000, 90000000, 100000000], "", "MagPhase", 0, 1, 11)</pre>

VB Syntax	<code>ExportNMFData(<sourceName>, <outputFileName>, <reduceMat>, <refImp>, <freqs>, <variationKey>, <displayFormat>, <pass>, <ComplexFormat>, <Precision>)</code>
VB Example	<pre>dim solns[1]</pre>

	<pre>solns[0] = "Setup1:Sweep1" solns[1] = "Setup1:Sweep2" dim freqs[5] freqs[0] = 10000000 freqs[1] = 20000000 freqs[2] = 30000000 freqs[3] = 40000000 freqs[4] = 50000000 freqs[5] = 60000000 oDesign.ExportNMFData(solns, "C:/outputs/data.nmf", "Original", 3.2133, freqs, "", "MagPhase", 2, 1, 11)</pre>

ExportProfile

Exports a solution profile to file.

UI Access	N/A		
Parameters	Name	Type	Description
	<SetupName>	String	Text of the design's notes.
	<VariationString>	String	Variation name. Pass empty string for the current nominal variation.
	<filePath>	String	Full file path, including extension *.prof.
	<probType>	String	Optional. For Q3D Extractor designs. Valid options are "CG", "DC RL", and "AC RL".
Return Value	None.		

Python Syntax	<code>ExportProfile(<SetupName>, <VariationString>, <filePath>, <probType>)</code>
Python Example	<code>oDesign.ExportProfile('Setup1', 'radius=3mm', 'C:\Files\Profile.prof', 'CG')</code>

VB Syntax	<code>ExportProfile <SetupName>, <VariationString>, <filePath>, <probType></code>
VB Example	<code>oDesign.ExportProfile "Setup1", "radius=3mm", "C:\Files\Profile.prof", "CG"</code>

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"
```

Python Syntax	ExportReport(<ReportName>, <FileName>)

Python Example	<pre>oDesign.ExportReport ("Plot1", "c:\report1.dat")</pre>

ExportRLGCData

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. <pre>Array(<SolnSelector>, <SolnSelector>, ...)</pre> 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 syntax, but ignored if DoRenorm is false.
	<DataType>	String	Optional. Type: "S", "Y", or "Z". The matrix to export.
	<pass>	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.
	<ComplexFormat>	Integer	Optional. Type: "0", "1", or "2" The format to use for the exported data. 0 = Magnitude/Phase. 1 = Real/Imaginary. 2 = db/Phase.
	<Precision>	Integer	Optional. Touchstone number of digits precision. Default if not specified is 15.
	<UseExportFreqs>	Boolean	Specifies whether to use export frequencies.
	<IncludeGammaComments>	Boolean	Specifies whether to include Gamma and Impedance comments.
	<SupportNonStdExport>	Boolean	Specifies whether to support non-standard Touchstone extensions for mixed reference impedance.
Return Value	None.		

Python Syntax	ExportRLGCData(<DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>, <FreqsArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <SupportNonStdExport>])
Python	oModule.ExportRLGCData("", ["Setup1:Sweep1"], 3, "C://Documents/package_

Example	<pre>HFSSDesign1.s4p", ['all'], True, 50, "S", -1, 0, 15, True, True, True</pre>

VB Syntax	<pre>ExportRLGCData <DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>, <FreqsArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <SupportNonStdExport>]</pre>
VB Example	<pre>oModule.ExportRLGCData "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>

ExportRLGCGen

Exports RLGC data for GenEquiv.

UI Access	From the Project Manager window, right-click a Q3D setup and select RLGC Matrix Data , then navigate to the RLGC Matrix Data tab.		
Parameters	Name	Type	Description
	<Contents>	Array	Structured array containing these parameters in order: • <NAME:ExportRLGCGen> • <FileName>

			• <SolutionName> • <Variation> • <VersionID> • <Pass> • <Frequencies>
	<NAME:ExportRLGCGen>	String	Identifies the contents of the array (i.e., "ExportRLGCGen").
	<FileName>	String	Full path to export directory for the new *.rlgdata file.
	<SolutionName>	String	Name of the solution to export data from.
	<Variation>	String	Empty string for current nominal variation.
	<VersionID>	String	The solution's ID number.
	<Pass>	String	The pass to export the data file. If the value is empty or -1, all passes are given.
	<Frequencies>	Array	Structured array containing a list of frequencies. If array is empty, all frequencies are used.
Return Value	None.		

Python Syntax	ExpOrRLGCGen([<Contents>])
Python Example	<pre>oDesign.ExportRLGCGen (["NAME:ExportRLGCGen", "FileName:=" , "C:\\name.rlgdata", "SolutionName:=" , "Q3D1:Adaptive Pass", "Variation:=" , "",</pre>

	<pre>"VersionID:=" , 138, "Pass:=" , -1, "Frequencies:=" , []])</pre>

VB Syntax	ExpOrRLGCGen (<Contents>)
VB Example	<pre>oDesign.ExportRLGCGen Array("NAME:ExportRLGCGen", "FileName:=" , "C:\\- name.rlgcdata", "SolutionName:=" , "Q3D1:Adaptive Pass", "Variation:=" , "", "Ver- sionID:=" , 138, "Pass:=" , -1, "Frequencies:=" , ())</pre>

Generate Mesh

If any mesh operations have been defined but 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.

UI Access	[HFSS 3D Layout] > Analysis Setup > Generate Mesh.		
Parameters	Name	Type	Description
	<SimulationNames>	Array	List of simulation names.
Return Value	Integer: • 0 – Success • Else – Failure		
Python Syntax	GenerateMesh(<SimulationNames>)		
Python Example	<pre>oDesign.GenerateMesh(['Setup1','Setup2','Setup3'])</pre>		

VB Syntax	GenerateMesh <SimulationNames>
VB Example	<code>oDesign.GenerateMesh Array('Setup1','Setup2','Setup3')</code>

GetActiveEditor (Layout Editor)

Use: Get the name of the active editor.

Command: None

Syntax: GetActiveEditor <object_variable>

Return Value: String

Parameters: <object_variable>

Type: string

VB Example: `oDesign.GetActiveEditor <object_variable>`

Note that GetActiveEditor returns NULL if the editor is open in a non-active window.

GetChildNames [Design]

Returns the names of the design's child objects.

UI Access	N/A		
Parameters	Name	Type	Description
	<type>	String	Optional. Valid options are "Module", "Editor", "Variable", and "Design Data". Default returns Module names.
Return Value	Names of children for the queried object (i.e., if "Design Data" is entered, Return Value will be ['Variables', 'Data-sets', 'Design Settings']).		

Python Syntax	GetChildNames (<type>)
Python Example	<code>oDesign.GetChildNames ("Design Data")</code>
VB Syntax	GetChildNames <type>
VB Example	<code>oDesign.GetChildNames "Design Data"</code>

GetChildObject [Design]

Returns the design's child objects.

UI Access	N/A		
Parameters	Name	Type	Description
	<path>	String	The path may include multiple generations (for example, "designObject/moduleObj/SetupObject"). See: Object Path .
Return Value	A child object if one is found. Otherwise script error.		

Python Syntax	GetChildObject(<path>)
Python Example	<pre> oDesign = oProject.GetActiveDesign() oOptimModule = oDesign.GetChildObject("Optimetrics") oEditor = oDesign.GetChildObject("3D Model") oBox = oDesign.GetChildObject("3D Model/Box1") oRpt = oDesign.GetChildObject("Results/S Parameter Plot 1") oVariable= oDesign.GetChildObject("Offset") </pre>

VB Syntax	<code>GetChildObject <path></code>
VB Example	<pre>oDesign = oProject.GetActiveDesign oOptimModule = oDesign.GetChildObject "Optimetrics" oEditor = oDesign.GetChildObject "3D Model" oBox = oDesign.GetChildObject "3D Model/Box1" oRpt = oDesign.GetChildObject "Results/S Parameter Plot 1" oVariable= oDesign.GetChildObject "Offset"</pre>

GetChildTypes [Design]

Returns the design's child object types.

UI Access	N/A
Parameters	None.
Return Value	String: "Module", "Editor", or "Variable"

Python Syntax	<code>GetChildTypes()</code>
Python Example	<code>oDesign.GetChildTypes()</code>

VB Syntax	GetChildTypes
VB Example	<code>oDesign.GetChildTypes</code>

GetConfigurableData

Obtains configurable data of a specified type

Note:

This command is for internal Ansys use only.

UI Access	N/A		
Parameters	Name	Type	Description
	<type>	String	Specified type.
Return Value	None.		

Python Syntax	GetConfigurableData(<type>)
Python Example	<code>oDesign.GetConfigurableData("model")</code>

VB Syntax	GetConfigurableData <type>
VB Example	<code>oDesign.GetConfigurableData "model"</code>

GetData

Note:

This command is for internal Ansys use only.

Python Syntax	GetData()
Python Example	<code>oDesign.GetData()</code>

GetDesignID

Returns the unique identification number of the active design.

UI Access	N/A
Parameters	None.
Return Value	String indicating the unique identification number of the active design.

Python Syntax	GetDesignID()
Python Example	<pre>oDesign = oProject.GetActiveDesign() oDesign.GetDesignID()</pre>

VB Syntax	GetDesignType
VB Example	<pre>Set oDesign = oProject.GetActiveDesign oDesign.GetDesignID</pre>

GetDesignName

Returns the name of the active design.

UI Access	N/A
Parameters	None.
Return Value	String indicating the name of the active design.

Python Syntax	GetDesignName()
Python Example	<pre>oDesign = oProject.GetActiveDesign() oDesign.GetDesignName()</pre>

VB Syntax	GetDesignName
VB Example	<pre>Set oDesign = oProject.GetActiveDesign oDesign.GetDesignName</pre>

GetDesignType

Returns the design type of the active design.

UI Access	N/A

Parameters	None.
Return Value	String indicating the design type of the active design ("Circuit Design", "Circuit Netlist", "EMIT", "HFSS 3D Layout Design", "HFSS", "HFSS-IE", "Icepak", "Maxwell 2D", "Maxwell 3D", "Q2D Extractor", "Q3D Extractor", "RMxpvt", or "Twin Builder").

Python Syntax	GetDesignType()
Python Example	<pre>oDesign = oProject.GetActiveDesign() oDesign.GetDesignType()</pre>

VB Syntax	GetDesignType
VB Example	<pre>Set oDesign = oProject.GetActiveDesign oDesign.GetDesignType</pre>

GetDesignValidationInfo

Returns the design's validation information.

UI Access	N/A		
Parameters	Name	Type	Description
	<validationResultFilePath>	String	Full path for validation file, including *.xml extension.
Return Value	Boolean: • 0 – Failure (validation not done)		

- 1 – Success (results written to specified location; see file format below)

Python Syntax	<code>GetDesignValidationInfo(<validationResultFilePath>)</code>
Python Example	<code>oDesign.GetDesignValidationInfo('E:/MyProjects/Validation.xml')</code>

VB Syntax	<code>GetDesignValidationInfo <validationResultFilePath></code>
VB Example	<code>oDesign.GetDesignValidationInfo "E:/MyProjects/Validation.xml"</code>

GetEditor

Use: Returns an interface to the editor requested IF the PropServer behind the PropHost is contained within that type of editor.

Command: None

Syntax: `GetEditor(<editorName>)`

Return Value: String

VB Example: `Set oLayout2 = PropHost.GetEditor("Layout");` returns the interface to the layout containing a selected component. This interface can be used to call Layout Scripting functions.

Python Syntax	<code>GetEditor(<editorName>)</code>
Python Example	<code>oEditor = GetEditor("SchematicEditor")</code>

GetEditor (Layout Editor)

Use: Get the named editor without activating it.

Command: None

Syntax: GetEditor <editorName object_variable>

Return Value: String

Parameters: <editorName>

Type: string

<object_variable>

Type: string

VB Example: oDesign.GetEditor <editorName> <object_variable>

GetGeometryIdsForNetLayerCombinations

Returns ID numbers of all faces and edges of the active design related to the current target combination of nets and layers.

Note: Intended to support [CreateFieldPlot](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<netName>	String	Net's name.
	<layerName>	String	Layer's name.
	<setupName>	String	Name of the setup.
Return Value	Array of strings indicating face and edge ID numbers and net/layer combination they are assigned to.		

Python Syntax	GetGeometryIdsForNetLayerCombinations ()
Python Example	<pre>oDesign = oProject.GetActiveDesign() oDesign.GetGeometryIdsForAllNetLayerCombinations ("<no-net>","Trace","HFSS Setup : Last Adaptive")</pre>
Example with Variables	<pre>oProject = oDesktop.SetActiveProject("Spiral_Inductor_Microstrip") oDesign = oProject.SetActiveDesign("Spiral") res = oDesign.GetGeometryIdsForNetLayerCombination("<no-net>","Trace","HFSS Setup : Last Adaptive") res = ['Surface', 'FacesList', '21', '22']</pre>

VB Syntax	GetGeometryIdsForNetLayerCombinations
VB Example	<pre>oDesign = oProject.GetActiveDesign() oDesign.GetGeometryIdsForAllNetLayerCombinations "<no-net>","Trace","HFSS Setup : Last Adaptive"</pre>

GetGeometryIdsForAllNetLayerCombinations

Returns ID numbers of all faces and edges of the active design for all possible combinations of nets and layers the user can choose.

Note: Intended to support [CreateFieldPlot](#).

UI Access	N/A

Parameters	Name	Type	Description
	<setupName>	String	Name of the setup.
Return Value	Array of strings indicating face and edge ID numbers.		

Python Syntax	GetGeometryIdsForAllNetLayerCombinations ()
Python Example	<pre>oDesign = oProject.SetActiveDesign() oDesign.GetGeometryIdsForAllNetLayerCombinations("HFSS Setup : Last Adaptive")</pre>
Example with Variables	<pre>oProject = oDesktop.SetActiveProject("Spiral_Inductor_Microstrip") oDesign = oProject.SetActiveDesign("Spiral") res = oDesign.GetGeometryIdsForAllNetLayerCombinations("HFSS Setup : Last Adaptive") res = ['PlotGeomInfo for <no-net>/<no-layer> (net/layer combination):', 'Surface', 'FacesList', '30', '31', '32', '33', '34', '35', '36', '37', '38', '39', '40', '41', '42', '43', '44', '45', '46', '47', '48', '49', '50', '51', '52', '53', '54', '55', '56', '57', 'PlotGeomInfo for <no-net>/AirbridgeMetal (net/layer combination):', 'Surface', 'FacesList', '14', 'PlotGeomInfo for <no-net>/BottomGround (net/layer combination):', 'Surface', 'FacesList', '29', 'PlotGeomInfo for <no-net>/Trace (net/layer combination):', 'Surface', 'FacesList', '21', '22']</pre>

VB Syntax	GetGeometryIdsForAllNetLayerCombinations
VB Example	<pre>oDesign = oProject.SetActiveDesign()</pre>

	<code>oDesign.GetGeometryIdsForAllNetLayerCombinations "HFSS Setup : Last Adaptive"</code>

GetManagedFilePath

Get the path to the project's results folder.

UI Access	N/A
Parameters	None.
Return Value	String containing path where project results are located.

Python Syntax	<code>oDesign.GetManagedFilePath()</code>
Python Example	<code>oDesign.GetManagedFilePath()</code>

VB Syntax	<code>oDesign.GetManagedFilePath</code>
VB Example	<code>oDesign.GetManagedFilePath</code>

GetModule

Returns the IDispatch for the specified module.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><modulename></code>	String	One of the following: Cavity Commands – "Cavities"

			• Design Verification Module – "DV" • Excitations Commands – "Excitations" or "Boundaries" • Field Overlays Module – "FieldsReporter" • Optimetrics Module – "Optimetrics" • Output Variable Module – "OutputVariable" • Radiation Setup Manager Module – "RadiationSetupMgr" • Reporter Module – "ReportSetup" • Simulation Setup Commands – "SolveSetups" • User Defined Documents Module – "UserDefinedDocuments" • User Defined Solutions Module – "UserDefinedSolutionModule"
Return Value	Module IDispatch		

Python Syntax	<code>GetModule (<modulename>)</code>
Python Example	<code>oModule = oDesign.GetModule("SimSetup")</code>

VB Syntax	<code>GetModule <modulename></code>
VB Example	<code>set oModule = oDesign.GetModule "SimSetup"</code>

GetModule (Layout Editor)

Returns the IDispatch for the specified module.

UI Access	N/A		
Parameters	Name	Type	Description
	<modulename>	String	One of the following: • Cavity Commands – "Cavities" • Design Verification Module – "DV" • Excitations Commands – "Excitations" or "Boundaries" • Field Overlays Module – "FieldsReporter" • Optimetrics Module – "Optimetrics" • Output Variable Module – "OutputVariable" • Radiation Setup Manager Module – "RadiationSetupMgr" • Reporter Module – "ReportSetup" • Simulation Setup Commands – "SolveSetups" • User Defined Documents Module – "UserDefinedDocuments" • User Defined Solutions Module – "UserDefinedSolutionModule"
Return Value	Module IDispatch		

Python Syntax	GetModule (<modulename>)
Python Example	<code>oModule = oDesign.GetModule("Cavities")</code>

VB Syntax	GetModule <modulename>
VB Example	<code>set oModule = oDesign.GetModule "Cavities"</code>

GetModule (RadiationSetupMgr)

Returns the IDispatch for the specified module.

UI Access	N/A		
Parameters	Name	Type	Description
	<modulename>	String	One of the following: • Cavity Commands – "Cavities" • Design Verification Module – "DV" • Excitations Commands – "Excitations" or "Boundaries" • Field Overlays Module – "FieldsReporter" • Optimetrics Module – "Optimetrics" • Output Variable Module – "OutputVariable" • Radiation Setup Manager Module – "RadiationSetupMgr" • Reporter Module – "ReportSetup" • Simulation Setup Commands – "SolveSetups" • User Defined Documents Module – "UserDefinedDocuments" • User Defined Solutions Module – "UserDefinedSolutionModule"
Return Value	Module IDispatch		

Python Syntax	GetModule (<modulename>)
Python Example	<code>oModule = oDesign.GetModule("SimSetup")</code>

VB Syntax	GetModule <modulename>
VB Example	set oModule = oDesign.GetModule("SimSetup")

GetName

Returns the design ID and design name of the active design, in that order separated by a semicolon. If the active design is a sub-circuit, it also returns the parent ID and parent design name, listed before the active design information.

To get the *only* the design's name without the parent design name or design IDs, see oDesign.GetDesignName().

UI Access	N/A
Parameters	None.
Return Value	String indicating the name and ID of the active design and parent design (if the active design is a sub-circuit).

Python Syntax	GetName()
Python Example	<pre>oDesign.GetName() for a subcircuit <i>EMDesign2</i> in a parent design <i>EMDesign1</i> design_name = oDesign.GetName(1;EMDesign1/2;EMDesign2 In the returned string: 1 is the parent design ID; <i>EMDesign1</i> is the parent design name. 2 is the active design ID; <i>EMDesign2</i> is the active design name.</pre>

VB Syntax	GetName
VB Example	<code>design_name = oDesign.GetName</code>

GetName (Layout Editor)

Get the name of the active design.

Command: None

Syntax: GetName <namestring_variable>

Return Value: String

Parameters: <namestring_variable>

Type: string

VB Example: `oDesign.GetName namestring_variable`

GetNoteText

Returns the text of the note attached to a design.

UI Access	N/A
Parameters	None.
Return Value	String: text of the design note.

Python Syntax	GetNoteText()

Python Example	<code>oDesign.GetNoteText()</code>

VB Syntax	<code>GetNoteText</code>
VB Example	<code>oDesign.GetNoteText</code>

GetObjPath [Design]

Obtains the path to the design.

UI Access	N/A
Parameters	None.
Return Value	String containing the path to the design.

Python Syntax	<code>GetObjPath()</code>
Python Example	<code>oDesign.GetObjPath()</code>

VB Syntax	<code>GetObjPath</code>
VB Example	<code>oDesign.GetObjPath</code>

GetOutputVariableValue

Returns the double value of an output variable. Only expressions that return a double value are supported. The expression is evaluated only for a single point.

UI Access	N/A		
Parameters	Name	Type	Description
	<OutputVarName>	String	Name of the output variable.
	<IntrinsicVariation>	String	A set of intrinsic variable value pairs to use when evaluating the output expression. HFSS example: <code>"Freq='20GHz' Theta='20deg' Phi='30deg'"</code>
	<SolutionName>	String	Name of the solution as listed in the output variable UI. For example, <code>"Setup1 : Last Adaptive"</code> .
	<ReportType>	String	The name of the report type as seen in the output variable UI. Possible values in Layout/Schematic Editor are: • "Standard" • "Load Pull" • "Constellation" • "Data Table" • "Eye Diagram" • "Statistical"
	<ContextArray>	Array	Structured array containing context for which the output variable expression is being evaluated. Can be empty. <code>Array("Context:=", <string>)</code>
Return Value	Double value of the output variable.		

Python Syntax	<code>GetOutputVariableValue(<OutputVarName>, <IntrinsicVariation>, <SolutionName>, <ReportTypeName>,</code>

	<code><ContextArray></code>)
Python Example	<pre>Val = oDesign.GetOutputVariableValue("test","Freq = '20Ghz' Theta='20deg' Phi='30deg'", "TR", "Standard", [])</pre>

VB Syntax	<code>GetOutputVariableValue <OutputVarName>, <IntrinsicVariation>, <SolutionName>, <ReportTypeName>, <ContextArray></code>
VB Example	<pre>Val = oDesign.GetOutputVariableValue "test","Freq = '20Ghz' Theta='20deg' Phi='30deg'", "TR", "Standard", Array()</pre>

GetOutputVariables

Returns the list of output variables.

UI Access	N/A
Parameters	None.
Return Value	Array containing all output variables.

Python Syntax	<code>GetOutputVariables()</code>
Python Example	<pre>oDesign.GetOutputVariables()</pre>

VB Syntax	<code>GetOutputVariables</code>

VB Example	<code>oDesign.GetOutputVariables</code>

GetProject

Returns to the active design's project or, if a project name is specified in the parentheses, changes focus to the specified project.

UI Access	N/A.		
Parameters	Name	Type	Description
	<ProjectName>	String	Name of an open project. Can be empty.
Return Value	None.		

Python Syntax	<code>GetProject(<ProjectName>)</code>
Python Example	<code>oDesign.GetProject('Project2')</code>

VB Syntax	<code>GetProject <ProjectName></code>
VB Example	<code>oDesign.GetProject "Project3"</code>

GetPropHost

Use: Returns an interface to the PropHost of the CompInstance, which gives access to its properties.

Command: None

Syntax: `GetPropHost()`

VB Example: `Set oPropHost2 = CompInstance.GetPropHost();`

Returns the interface to the properties of the compInstance.

This interface can be used to call PropHost functions; for more information see [Callback Scripting Using PropHost Object](#).

Python Syntax	GetPropHost()
Python Example	<code>oPropHost2 = CompInstance.GetPropHost()</code>

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.

UI Access	N/A		
Parameters	Name	Type	Description
	<PropTab>	String	One of the following, where tab titles are shown in parentheses: - PassedParameterTab ("Parameter Values") - DefinitionParameterTab (Parameter Defaults") - LocalVariableTab ("Variables" or "Local Variables") - ProjectVariableTab ("Project variables") - ConstantsTab ("Constants") - BaseElementTab ("Symbol" or "Footprint") - ComponentTab ("General") - Component("Component") - CustomTab ("Intrinsic Variables") - Quantities ("Quantities") - Signals ("Signals")
	<PropServer>	String	An object identifier, generally returned from another script method, such as

	CompInst@R;2;3
Return Value	Array of strings containing the names of the appropriate properties.

Python Syntax	<code>GetProperties(<PropTab>, <PropServer>)</code>
Python Example	<code>oEditor.GetProperties('PassedParameterTab', 'k')</code>

VB Syntax	<code>GetProperties <PropTab>, <PropServer></code>
VB Example	<code>oEditor.GetProperties "PassedParameterTab", "k"</code>

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.

UI Access	N/A		
Parameters	Name	Type	Description
	<i><PropTab></i>	String	One of the following, where tab titles are shown in parentheses: - PassedParameterTab ("Parameter Values") - DefinitionParameterTab (Parameter Defaults) - LocalVariableTab ("Variables" or "Local Variables")

			• ProjectVariableTab ("Project variables") • ConstantsTab ("Constants") • BaseElementTab ("Symbol" or "Footprint") • ComponentTab ("General") • Component("Component") • CustomTab ("Intrinsic Variables") • Quantities ("Quantities") • Signals ("Signals")
	<i><PropServer></i>	String	An object identifier, generally returned from another script method, such as <code>CompInst@R;2;3</code>
	<i><PropName></i>	String	Name of the property.
Return Value	String value of the property.		

Python Syntax	<code>GetPropertyValue (<PropTab>, <PropServer>, <PropName>)</code>
Python Example	<pre>selectionArray = oEditor.GetSelections() for k in selectionArray: val = oEditor.GetPropertyValue("PassedParameterTab", k, "R") ...</pre>

VB Syntax	<code>GetPropertyValue (<PropTab>, <PropServer>, <PropName>)</code>
VB Example	<pre>selectionArray = oEditor.GetSelections for k in selectionArray:</pre>


```
val = oEditor.GetPropertyValue("PassedParameterTab", k, "R")  
...
```

GetPropNames [Design]

Returns array containing names of property. For designs, always returns an empty array because the design has no property.

UI Access	N/A		
Parameters	Name	Type	Description
	<includeReadOnly>	Boolean	(Optional). • True - include read only property. • False - do not include read only property.
Return Value	Empty array.		
Python Syntax	GetPropNames(<includeReadOnly>)		
Python Example	oDesign.GetPropNames()		
VB Syntax	GetPropNames <includeReadOnly>		
VB Example	oDesign.GetPropNames		

GetPropValue [Design]

Returns the property value for the active design object, or specified property values.

UI Access	N/A		
Parameters	Name	Type	Description
	<propPath>	String	A child object's property path. See: Object Property Script Function Summary .
Return Value	The property value (integer, string, or array of strings) of the specified child object.		

Python Syntax	GetPropValue(<propPath>)		
Python Example	oDesign.GetPropValue('offset/SIValue')		
	oDesign.GetPropValue('Results/S Parameter Plot 1/Display Type')		
	oDesign.GetPropValue('Results/S Parameter Plot 1/Display Type/Choices')		

VB Syntax	GetPropValue(<propPath>)		
VB Example	val = oDesign.GetPropValue("offset/SIValue")		
	disp = oDesign.GetPropValue("Results/S Parameter Plot 1/Display Type")		
	arr = oDesign.GetPropValue("Results/S Parameter Plot 1/Display Type/Choices")		

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.

UI Access	N/A
Parameters	None.

Return Value	Array of strings containing the variables.

Python Syntax	GetVariables ()
Python Example	<pre>oProject.GetVariables() oDesign.GetVariables()</pre>

VB Syntax	GetVariables
VB Example	<pre>oProject.GetVariables oDesign.GetVariables</pre>

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`.

UI Access	N/A		
Parameters	Name	Type	Description
	<VarName>	String	Name of the variable to access.
Return Value	String represents the value of the variable.		

Python Syntax	<code>GetVariableValue(<VarName>)</code>
Python Example	<code>oProject.GetVariableValue("var_name")</code>

VB Syntax	<code>GetVariableValue <VarName></code>
VB Example	<code>oProject.GetVariableValue "var_name"</code>

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.		
Parameters	Name	Type	Description
	<datasetFileFullPath>	String	The full path to the file containing the dataset values. *.tab files recommended (see note below).
	<optionalDatasetName>	String	<i>Optional.</i> User-defined dataset name.
Return Value	None.		

Python Syntax	<code>ImportDataset (<datasetFileFullPath>,<optionalDatasetName>)</code>
Python Example	<pre>oProject.ImportDataset('e:\tmp\dsdata.tab') oDesign.ImportDataset('e:\tmp\dsdata.tab')</pre>

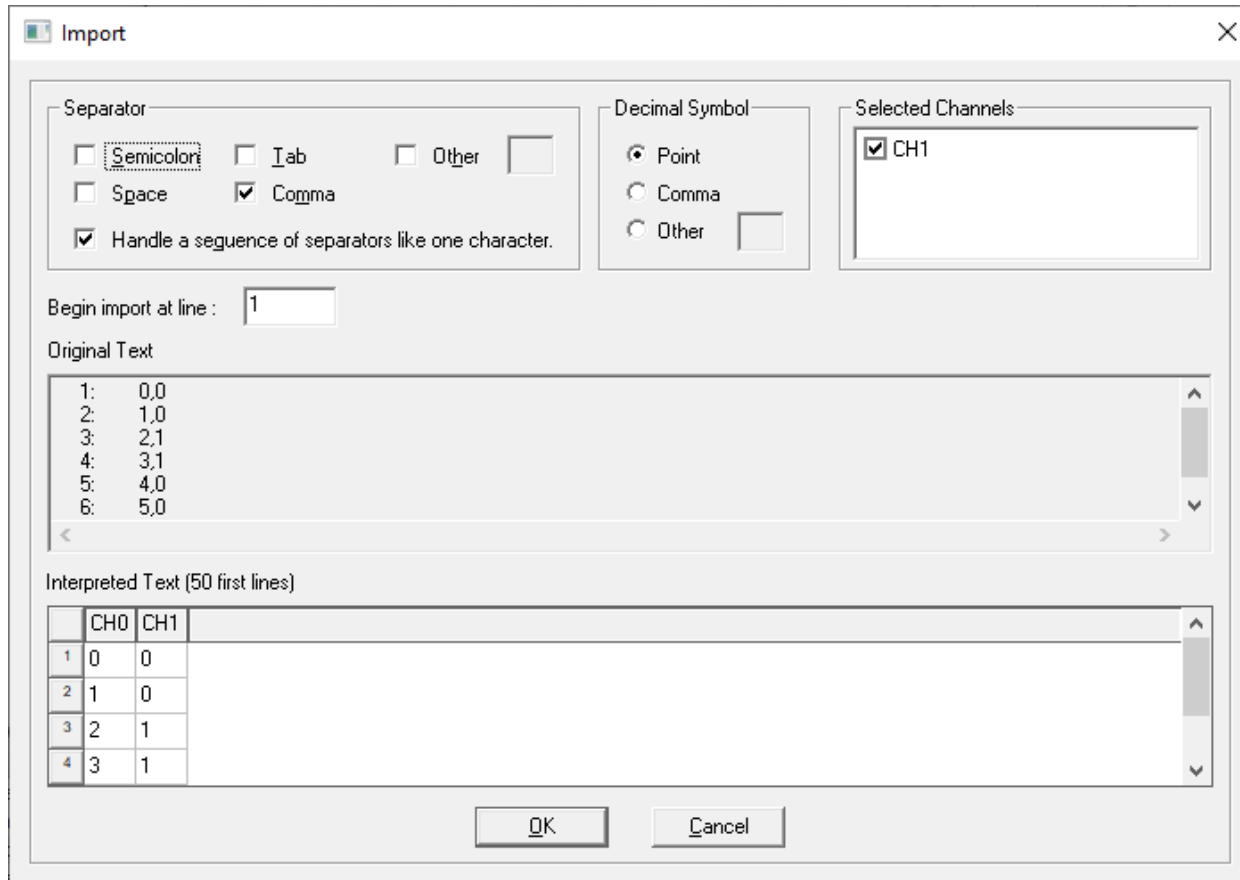
	<pre>oProject.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName') oDesign.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName')</pre>

VB Syntax	ImportDataset <datasetFilePath>, <optionalDatasetName>
VB Example	<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**.



InsertDesign

Inserts a new design in the project. For Circuit and HFSS 3D Layout scripts, the last argument will always be empty.

UI Access	Project > [Insert Circuit Design / Insert Circuit Netlist / Insert HFSS 3D Layout Design].		
Parameters	Name	Type	Description
	<DesignType>	String	Design type. Valid types: "Circuit", "Nexxim Circuit", "System", "Nexxim Netlist", "HFSS3D".
	<DesignName>	String	Design name.
	<TechnologyFile>	String	The path to the Circuit technology file to be used in the design. Use a pair of empty double quotes ("") for none.
	<SubcircuitID>	String	(Optional) Must be preceded by a colon if included along with the Technology File name. No colon is necessary when the subcircuit ID is omitted.
	""	None	Empty argument.
Return Value	None.		

Python Syntax	InsertDesign (<DesignType>, <DesignName>, <TechnologyFile>:<SubCircuitID>,"")
Python Example	oProject.InsertDesign("Nexxim Circuit","MyDesign", "C:\\Program Files\\ANSYS Inc\\v251\\AnsysEM", "")

VB Syntax	InsertDesign <DesignType>, <DesignName>, <TechnologyFile>:<SubCircuitID>,""
VB Example	oProject.InsertDesign "Nexxim Circuit","MyDesign", "C:\\Program Files\\ANSYS Inc\\v251\\AnsysEM", ""

OverlayCurrents (Layout Editor)

Creates a 3D view with a currents overlay

Command: None

Syntax: OverlayCurrents <solution name>

Return Value: <success>

Type: boolean

Parameters: <solution name>

Type: string

VB Example: Dim success

```
success = oDesign.OverlayCurrents ("HFSS Setup : Sweep 1")
```

OverlayFarField (Layout Editor)

Creates a 3D view with a far field overlay

Command: None

Syntax: OverlayFarField <solution name>

Return Value: <success>

Type: boolean

Parameters: <solution name>

Type: string

VB Example: Dim success

```
success = oDesign.OverlayFarField ("HFSS Setup : Sweep 1")
```

OverlayMesh (Layout Editor)

Creates a 3D view with a mesh overlay

Command: None

Syntax: OverlayMesh <solution name>

Return Value: <success>

Type: boolean

Parameters: <solution name>

Type: string

VB Example: Dim success

```
success = oDesign.OverlayMesh ("HFSS Setup : Sweep 1")
```

OverlayNearField (Layout Editor)

Creates a 3D view with a near field overlay

Command: None

Syntax: OverlayNearField <solution name>

Return Value: <success>

Type: boolean

Parameters: <solution name>

Type: string

VB Example: Dim success

```
success = oDesign.OverlayNearField ("HFSS Setup : Sweep 1")
```

PasteDesign (Design Object)

Pastes a design that has already been copied to the clipboard into another design.

UI Access	Edit > Paste.		
Parameters	Name	Type	Description
	<pasteOption>	Integer	One of the following: • 0 – Link to the existing design that was copied • 1 – Create a new copy of the design and keep the original layers of the design being copied • 2 – Create a new copy of the design and merge the layers of the design being copied.
Return Value	None.		

Python Syntax	PasteDesign(<pasteOption>)
Python Example	<pre>oProject.CopyDesign('DesignB') oDesign = oProject.SetActiveDesign('DesignA') oDesign.PasteDesign(1)</pre>

VB Syntax	PasteDesign <pasteOption>
VB Example	<pre>oProject.CopyDesign "B" Set oDesign = oProject.SetActiveDesign("A") oDesign.PasteDesign 1</pre>

PasteItemCommand (Layout Editor)

Paste tree items, such as Altrasm Solution and Solve Setups pasted to the Analysis Tree, or Frequency Sweeps pasted to the Solve Setup Tree.

Command: Right-click on the Analysis item in the Project tree and select Paste.

Syntax: PasteItemCommand <ItemPath>

Return Value: None

Parameters: <ItemPath>

Type: string

VB Example: oDesign.PasteItemCommand "Project1|Nexxim1|Analysis"

PushExcitations

Allows access to computed excitations for transient and linear frequency solutions. The script command can be accessed from three locations.

UI Access	- Layout Editor - Schematic Editor - Select a Nexxim solution in a 3D Layout design, right click, and choose Push Excitations.		
Parameters	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 =" , <true false>, "Sol:=", "<solution name>") </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", "transient:=", Array("start:=", <start time>, "stop:=", <stop time>, "maxHarmonics:=", <max harmonics>, "winType:=", <window>, ["widthPct:=", <width percentage>], ["kaiser:=", <Kaiser value>], ["correctCoherentGain:=", true]), "Sol:=", "<solution name>") </pre> winType can have the following values: • Rectangular • Bartlett • Blackman • Hamming • Hanning

			• Kaiser • Welch • Weber • Lanczos
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

VB Syntax	<code>PushExcitations "<refdes>", Array("NAME:options", "transient:=", ["CalcThevenin =", <true>], Array("start:=", <start time>, "stop:=", <stop time>, "maxHarmonics:=", <max harmonics>, "winType:=", <window>, ["widthPct:=", <width percentage>], ["kaiser:=", <Kaier value>], ["correctCoherentGain:=", true]), "Sol:=", "<solution name>")</code>
VB Example	For a transient solution: <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>

```

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:=", _
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")

```

Redo [Design]

Reapplies the last design-level command.

UI Access	Edit > Redo.
Parameters	None.
Return Value	None.

Python Syntax	Redo()
Python Example	oDesign.Redo()

VB Syntax	Redo
VB Example	<code>oDesign.Redo</code>

Redo (Layout Editor)

Redo the last operation

Command: Edit>Redo

Syntax: Redo

Return Value: None

Parameters: None

VB Example: `oDesign.Redo`

RemoveImportData (Layout Editor)

Remove an imported solution

Command: None

Syntax: RemoveImportData <solution_name>

Return Value: None

Parameters: <solution_name>

Type: string

VB Example: `oDesign.RemoveImportData <solution_name>`

RemoveModelingProperties (Layout Editor)

Remove a modeling property from a design

Command: None

Syntax: RemoveModelingProperties <design>

Return Value: None

Parameters: <design>

Type: string

VB Example: oDesign.RemoveModelingProperties <design>

RenameDesignInstance

Renames a design instance.

UI Access	Right-click a design instance in the project tree, and then click Rename on the shortcut menu.		
Parameters	Name	Type	Description
	<OldName>	String	The current name of the design, which must be the design on which this command is invoked.
	<NewName>	String	The new name for the design.
Return Value	None.		

Python Syntax	RenameDesignInstance (<OldName>, <NewName>)
Python Example	oDesign.RenameDesignInstance ("Design1", "Design2")

VB Syntax	RenameDesignInstance <OldName>, <NewName>
VB Example	oDesign.RenameDesignInstance "Design1", "Design2"

RenameDesignInstance (Layout Editor)

Rename the design instance

Command: None

Syntax: RenameDesignInstance <oldname> <newname>

Return Value: None

Parameters: <oldname>

Type: string

<newname>

Type: string

VB Example: oDesign.RenameDesignInstance <oldname> <newname>

RenameImportData (Layout Editor)

Rename an imported solution

Command: None

Syntax: RenameImportData <oldname> <newname>

Return Value: None

Parameters: <oldname>

Type: string

<newname>

Type: string

VB Example: oDesign.RenameImportData <oldname> <newname>

RenameSource [Interface Source]

Use: Renames an interface source

Syntax: RenameSource (STRING: Interface Source name)

VB Example: oModule.RenameSource "OldName" "NewName"

ReportTemplates (Layout Editor)

Use: Creates a report using one of the HFSS 3D Layout report templates

Command: None

Syntax: ReportTemplates <template>

Return Value: None

Parameters: <template>

Type: string

Example: oDesign.ReportTemplates <template>

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.

UI Access	[HFSS 3D Layout] > Toolkit > [Script Name].		
Parameters	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.
Return Value	User-defined solution(s) and/or output(s).		

Python Syntax	RunToolkit(<LibraryType>, <ToolkitName>, <ToolkitArg>)
Python Example	oDesign.RunToolkit("SysLib", "Hearing Aid Compliance", [])

VB Syntax	RunToolkit <LibraryType>, <ToolkitName>, <ToolkitArg>
VB Example	oDesign.RunToolkit "SysLib", "Wavelength Calculator", Array()
	oDesign.RunToolkit "SysLib", "Hearing Aid Compliance", Array()

SetActiveEditor

Sets the active editor.

UI Access	N/A.		
Parameters	Name	Type	Description
	<EditorName>	String	Text of the design's notes.
Return Value	Editor object.		

Python Syntax	SetActiveEditor(<EditorName>)
Python Example	oDesign.SetActiveEditor('3D Modeler')

VB Syntax	SetActiveEditor <EditorName>
VB Example	oDesign.SetActiveEditor "3D Modeler"

SetActiveEditor (Layout Editor)

Set the active editor to the named one. (Activates window or opens window if necessary).

Command: None

Syntax: SetActiveEditor <editorName> <object_variable>

Return Value: None

Parameters: <editorName>

Type: string

<object_variable>

Type: string

VB Example: oDesign.SetActiveEditor <editorName> <object_variable>

SetBendGraphicsMode

Activates/deactivates a simplified version of the model rendering for designs with rigid-flex bends.

UI Access	From the Layout tab, select Rigid-Flex > Simplified Bend Graphics .		
Parameters	Name	Type	Description
	<SetBendGraphics>	Integer	0 – Deactivates Simplified Bend Graphics 1 – Activates Simplified Bend Graphics
Return Value	None.		

Python Syntax	SetBendGraphics (<SetBendGraphics>)
Python Example	oEditor.SetBendGraphicsMode ("1")

VB Syntax	SetBendGraphics <SetBendGraphics>
VB Example	oEditor.SetBendGraphicsMode ('1')

SetDesignMode

Switches between HFSS 3D Layout operating modes.

UI Access	Edit Layout...

Parameters	Name	Type	Description
	<modeName>	String	Enter "IC" or "General" to change operating modes.
Return Value	None.		

Python Syntax	SetDesignMode(<modeName>)
Python Example	oDesign.SetDesignMode ("IC")

VB Syntax	SetDesignMode <modeName>
VB Example	oDesign.SetDesignMode "IC"

SetDoMeshAssembly

Set the active Mesh fusion properties for each object.

Command: Set Components for Mesh Fusion

Syntax: SetDoMeshAssembly <MeshAssemblyParameters>

Return Value: None

Parameters: <MeshAssemblyParameters>

Python Example

Example with Priority Level Enabled, with a section for Component Priority Level is added.

```
oDesign.SetDoMeshAssembly(
[
    "NAME:AllSettings",
```

```
[
  "NAME:MeshAssembly",
  [
    "NAME:LC1_2",
    "MeshAssemblyBoundingVolumePadding:=", ["0","0","0","0","0","0"]
  ],
  [
    "NAME:LC1_3",
    "MeshAssemblyBoundingVolumePadding:=", ["0","0","0","0","0","0"]
  ],
  [
    "NAME:Modal1",
    "MeshAssemblyBoundingVolumePadding:=", ["0","0","0","0","0","0"]
  ]
],
"Component Priority Level:=", ["LC1_1","Modal1","LC1_2","Modal_with_mesh_region1","LC1_3","Modal3"]
])
```

If a design which had Priority Level set but then disabled, a section of the script records the disabled priority level.

```
oDesign.SetDoMeshAssembly(
```

```
[
  "NAME:AllSettings",
  [
    "NAME:MeshAssembly",
    [
      "NAME:LC1_3",
      "MeshAssemblyBoundingVolumePadding:=", ["0","0","0","0","0","0"]
    ],
    [
      "NAME:Modal1",
      "MeshAssemblyBoundingVolumePadding:=", ["0","0","0","0","0","0"]
    ]
  ],
  "Priority Components:=" , ["LC1_3","Modal1","Modal_with_mesh_region1"]
])
```

If Priority Level is not enabled, the behavior is as before.

```
oDesign.SetDoMeshAssembly(
  [
    "NAME:AllSettings",
    [
```



```
"NAME:MeshAssembly",  
[  
  "NAME:Cylinder3",  
  [  
    "NAME:MeshSetting",  
    [  
      "NAME:GlobalSurfApproximation",  
      "CurvedSurfaceApproxChoice:=", "UseSlider",  
      "SliderMeshSettings:=" , 7  
    ],  
    [  
      "NAME:GlobalCurvilinear",  
      "Apply:=" , False  
    ],  
    [  
      "NAME:GlobalModelRes",  
      "UseAutoLength:=" , True  
    ],  
    "MeshMethod:=" , "AnsoftClassic",  
    "UseLegacyFaceterForTauVolumeMesh:=", False,
```

```
"DynamicSurfaceResolution:=", False,  
"UseFlexMeshingForTAUvolumeMesh:=", False,  
"UseAlternativeMeshMethodsAsFallBack:=", True,  
"AllowPhiForLayeredGeometry:=", True  
],  
"MeshAssemblyBoundingVolumePadding:=", ["0","0","0","0","0","0"]  
],  
[  
  "NAME:Pentagon_tilted2_1",  
  "MeshAssemblyBoundingVolumePadding:=", ["0","0","0","0","0","0"]  
],  
[  
  "NAME:Cone_with_variable1",  
  "MeshAssemblyBoundingVolumePadding:=", ["0","0","0","0","0","0"]  
]  
],  
[  
  "NAME:Priority Components",  
  "Cylinder3",  
  "Cone_with_variable1",  
  "BoxWithMeshRegion1"
```

]

])

SetFastTransformationForLayoutComponent

Toggles whether layout components in show outlines of objects across layers during view changes. This improves the visualization performance.

UI Access	Fast Transformation		
Parameters	Name	Type	Description
	<Layout ComponentName>	String	Name of the Layout Component.
	Boolean	String	True or False
Return Value	None.		

Python Syntax	SetFastTransformationForLayoutComponent ("<LayoutComponentName", <boolean>)
Python Example	oDesign.SetFastTransformationForLayoutComponent ("LC1_1", False)

SetObjectAttributesForLayoutComponent

Sets visualization attributes for layout components in for a named component for layers, nets, and objects.

UI Access	Object Attributes		
Parameters	Name	Type	Description
	<Layout ComponentName>	String	Name of the Layout Component.

Return Value	None.

Python Syntax	SetObjectAttributesForLayoutComponent ("<LayoutComponentName>",[<ObjectAttributesinNetMode>][<ObjectAttributesinLayerMode>],[<ObjectAttributesinObjectMode>])
Python Example	<pre> oDesign.SetObjectAttributesForLayoutComponent (["Name:=" , "LC1_1", "ShowDielectric:=" , False, "DisplayMode:=" , 2, ["NAME:ObjectAttributesInLayerMode", "GND_1_L2:=" , [True,True,0], "GND_2_L4:=" , [True,True,0], "GND_3_L6:=" , [True,True,0], "GND_4_L9:=" , [True,True,0], "GND_5_L11:=" , [True,True,0], "Inner_Layer_3_L10:=" , [True,True,0], "Top_L1:=" , [True,True,100], "VCC1_L7:=" , [True,True,0], "VCC2_L8:=" , [True,True,0]]], </pre>

```
[  
  "NAME:ObjectAttributesInNetMode",  
  "GND:="          , [True,True,29],  
  "VCC:="          , [True,True,0],  
  "neg:="          , [True,True,67],  
  "pos:="          , [True,True,67]  
],  
[  
  "NAME:ObjectAttributesInObjectMode",  
  "line_1:="        , [True,True,0],  
  "line_2:="        , [True,True,0],  
  "line_3:="        , [True,True,100],  
  "line_4:="        , [True,True,100],  
  "rect_6:="        , [True,True,0],  
  "rect_17:="       , [True,True,0],  
  "rect_18:="       , [True,True,0],  
  "rect_19:="       , [True,True,0],  
  "rect_20:="       , [True,True,0],  
  "rect_21:="       , [True,True,0],  
  "rect_22:="       , [True,True,0],
```

	<pre> "line_24:=" , [True,True,100], "via_0:=" , [True,True,100], "via_1:=" , [True,True,100], "via_2:=" , [True,True,100], "via_3:=" , [True,True,100]]]) </pre>

SetPropValue [Design]

Sets the property value for the active property object.

UI Access	Edit properties on Project Tree objects.		
Parameters	Name	Type	Description
	<propPath>	String	A child object's property path. See: Object Property Script Function Summary .
	<Value>	String	New property value.
Return Value	Boolean: • True - property found and the new value is valid. • False - property not found.		

Python Syntax	SetPropValue(<propPath>, <Value>)
Python	oDesign.SetPropValue("offset", "12mm")

Example	<code>oDesign.SetPropValue("Results/S Parameter Plot 1/Display Type", "Rectangular Plot")</code>

VB Syntax	<code>SetPropValue <propPath>, <Value></code>
VB Example	<code>oDesign.SetPropValue "offset", "12mm"</code> <code>oDesign.SetPropValue "Results/S Parameter Plot 1/Display Type", "Rectangular Plot"</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.

UI Access	N/A		
Parameters	Name	Type	Description
	<propTab>	String	One of the following, where tab titles are shown in parentheses: - PassedParameterTab ("Parameter Values") - DefinitionParameterTab (Parameter Defaults") - LocalVariableTab ("Variables" or "Local Variables") - ProjectVariableTab ("Project variables") - ConstantsTab ("Constants") - BaseElementTab ("Symbol" or "Footprint") - ComponentTab ("General")

			• Component("Component") • CustomTab ("Intrinsic Variables") • Quantities ("Quantities") • Signals ("Signals")
	<i><propServer></i>	String	An object identifier, generally returned from another script method, such as <code>CompInst@R;2;3</code>
	<i><propName></i>	String	Name of the property.
	<i><propValue></i>	String	The value for the property
Return Value	None.		

Python Syntax	<code>SetPropertyValue(<propTab>, <propServer>, <propName>, <propValue>)</code>		
Python Example	<code>oEditor.SetPropertyValue("PassedParameterTab", "k", "R", "2200")</code>		

VB Syntax	<code>SetPropertyValue <propTab>, <propServer>, <propName>, <propValue></code>		
VB Example	<code>oEditor.SetPropertyValue "PassedParameterTab", "k", "R", "2200"</code>		

SetShowLayoutForLayoutComponent

Layout visualization, rather than bounding box, for a Layout Component in design.

UI Access	Edit Layout...		
Parameters	Name	Type	Description
	<i><Layout ComponentName></i>	String	Name of the Layout Component.

	<table><tr><td>Boolean</td><td>String</td><td>True or False</td></tr></table>	Boolean	String	True or False
Boolean	String	True or False		
Return Value	None.			

Python Syntax	SetShowLayoutForLayoutComponent ("<LayoutComponentName", <boolean>)
Python Example	<code>oDesign.SetShowLayoutForLayoutComponent ("LC1_1", True)</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`.

UI Access	N/A		
Parameters	Name	Type	Description
	<VarName>	String	Variable name.
	<VarValue>	Value	New value for the variable.
Return Value	None.		

Python Syntax	SetVariableValue (<VarName>, <VarValue>)
Python Example	<code>oProject.SetVariableValue ('\$Var1', '3mm')</code>

VB Syntax	SetVariableValue <VarName>, <VarValue>
VB Example	oProject.SetVariableValue "\$Var1", "3mm"

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 .		
Parameters	Name	Type	Description
	<SimulationNames>	Array	Array containing string simulation names.
Return Value	Integer: • 0 – Simulation(s) completed. • 1 – Simulation error. • -1 – Command execution error.		

Python Syntax	Solve <SimulationNames>
Python Example	oDesign.Solve(['Setup1', 'Setup2', 'Setup3'])

VB Syntax	Solve <SimulationNames>
VB Example	oDesign.Solve Array("Setup1", "Setup2", "Setup3")

StartAnalysis (Layout Editor)

Simulates all setups

Command: None

Syntax: StartAnalysis

Return Value: None

Parameters: None

VB Example: oDesign.StartAnalysis

StopSimLink

Stops linked simulation.

UI Access	N/A		
Parameters	Name	Type	Description
	<simId>	Integer	ID of specified simulation.
	<abort>	Boolean	Whether to stop the running simulation: • True - abort the running simulation. • False - do not abort running simulation.
Return Value	None.		

Python Syntax	StopSimLink(<simId>, <abort>)
Python Example	oDesign.StopSimLink(18, True)

VB Syntax	StopSimLink <simId>, <abort>
VB Example	oDesign.StopSimLink 18, true

Undo [Design]

Cancels the last design-level command.

UI Access	Edit > Undo
Parameters	None.
Return Value	None.

Python Syntax	Undo()
Python Example	<code>oDesign.Undo()</code>

VB Syntax	Undo
VB Example	<code>oDesign.Undo</code>

Undo (Layout Editor)

Undo the last operation

Command: Edit>Undo

Syntax: Undo

Return Value: None

Parameters: None

VB Example: `oDesign.Undo`

ValidateCircuit (Layout Editor)

Use: Validates a HFSS 3D Layout design for consistency

Command: None

Syntax: ValidateCircuit

Return Value: Boolean:

- 1 - design is valid
- 0 - design is not valid

Parameters: None

Example: oDesign.ValidateCircuit

ValidateLink

Note:
This command is for internal Ansys use only.

Python Syntax	ValidateLink()
Python Example	oDesign.ValidateLink()

8 - 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](#)

[CreateSubregion](#)

[CreateTorus](#)

[CreateUserDefinedPart](#)

[Edit3DComponent](#)

[EditPolyline](#)

[Get3DComponentDefinitionNames](#)

[Get3DComponentInstanceNames](#)

[Get3DComponentMaterialNames](#)

[Get3DComponentMaterialProperties](#)

[Get3DComponentParameters](#)

[Insert3DComponent](#)

[InsertPolylineSegment](#)

[SweepAlongPath](#)

[SweepAlongVector](#)

[SweepAroundAxis](#)

[SweepFacesAlongNormal](#)

[SweepFacesAlongNormalWithAttributes](#)

[UpdateComponentDefinition](#)

Create3DComponent

Creates a 3D component.

UI Access	Draw > 3D Component Library > Create 3D Component.		
Parameters	Name	Type	Description
	<CreateData>	Array	Structured array containing geometry data: (Array["NAME:CreateData", "ComponentName:=", "<string>", "Company:=", "<string>", "Company URL:=", "<string>", "Model Number:=", "<string>", "Help URL:=" , "<string>", "Version:=", "<string>", "Notes:=", "<string>", "IconType:=", "<string>", "Owner:=", "<string>", "Email:=", "<string>", "Date:=", "<string>", #Component creation date "HasLabel:=", <boolean>, "IsEncrypted:=", <boolean>,

			<pre> "AllowEdit:=", <boolean>, "SecurityMessage:=" , "<string>", "Password:=" , "<string>", "EditPassword:=" , "<string>", "PasswordType:=" , "<string>", "HideContents:=" , <boolean>, "ReplaceNames:=" , <boolean>, #for objects and materials "ComponentOutline:=" , "<string>","#None" or "Bounding Box" "IncludedParts:=" , [Array of partnames], "HiddenParts:=" , [Array of hidden parts], "IncludedCS:=" , [Array of included CS], "DefaultHandle:=" , "<reference CS>" "IncludedParameters:=" , [Array of included parameters], "IncludedDependentParameters:=", [Array of dependent parameters], "ParameterDescription:=", <array> "IsLicensed:=" , <boolean> "LicensingDllName:=" , "<string>" "VendorComponentIdentifier:=", "<string>" "PublicKeyFile:=" , "<string>"] </pre>

	<i><DesignData></i>	Array	Structured array containing design data: Array["NAME:DesignData", "Boundaries:=", <array>, "Excitations:=", <array>, "MeshOperations:=", <array>]
	<i><FileName></i>	String	Full file path to 3D component.
	<i><ImageFile></i>	Array	Optional. Structured array containing file path of 3D component image: Array["NAME:ImageFile", "ImageFile:=", <string>])
Return Value	None.		

Python Syntax	Create3DComponent(<i><CreateData></i> , <i><DesignData></i> , <i><FileName></i> , <i><ImageFile></i>)
Python Example	<pre>oEditor.Create3DComponent (["NAME:CreateData", "ComponentName:=", "Connector", "Company:=", "", "Version:=", "1.0", "Notes:=", "", "Owner:=", "",</pre>

```

"Email:=", "",
"Date:=", "11:41:01 AM Aug 28, 2014",
"HasLabel:=", false,
"IncludedParts:=", ["Box1", "Cylinder1", "Cone1"],
"IncludedCS:=", ["RelativeCS1"],
"DefaultHandle:=", "Global",
"IncludedParameters:=", ["htcone", "lr", "htcyl", "zs", "radcyl", "xs", "$rp",
"$con"],
"ParameterDescription:=", [],
"IsLicensed:=",      , False,
"LicensingDllName:=",      , "",
"VendorComponentIdentifier:=", "",
"PublicKeyFile:=",      , ""
],
["NAME:DesignData",
  "Boundaries:=", ["PerfE1", "FiniteCond1"],
  "Excitations:=", ["1"],
  "MeshOperations:=", [],
"C:/tmp/Connector.a3dcomp",
["NAME:ImageFile", "ImageFile:=", ""]]

```

VB Syntax	Create3DComponent <CreateData>, <DesignData>, <FileName>, <ImageFile>
VB Example	<pre> oEditor.Create3DComponent Array("NAME:CreateData", "ComponentName:=", "Connector", "Company:=", "", "Version:=", "1.0", "Notes:=", "", "Owner:=", "", "Email:=", "", "Date:=", "11:41:01 AM Aug 28, 2014", "HasLabel:=", false, "IncludedParts:=", Array("Box1", "Cylinder1", "Cone1"), "IncludedCS:=", Array("RelativeCS1"), "DefaultHandle:=", "Global", "IncludedParameters:=", Array("htcone", "lr", "htcyl", "zs", "radcyl", "xs", "\$rp", "\$con"), "ParameterDescription:=", Array() "IsLicensed:=", , False, "LicensingDllName:=", , "", "VendorComponentIdentifier:=", "", </pre>

```

"PublicKeyFile:="      , ""
),
Array("NAME:DesignData",
    "Boundaries:=", Array( "PerfE1", "FiniteCond1"),
    "Excitations:=", Array("1"),
    "MeshOperations:=", Array()),
"C:/tmp/Connector.a3dcomp",
Array("NAME:ImageFile", "ImageFile:=", "")

```

CreateBondwire

Creates a bondwire.

UI Access	Draw > Bondwire.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre> Array("NAME:BondwireParameters", "WireType:=", <string("JEDEC_4Points", "JEDEC_5Points", or "LOW")>, "WireDiameter:=", <string>, "NumSides:=", <value>, "XPadPos:=", <value>, "YPadPos:=", <value>, "ZPadPos:=", <value>, </pre>

			<pre> "XDir:=", <value>, "YDir:=", <value>, "ZDir:=", <value>, "Distance:=", <value>, "h1:=", <value>, "h2:=", <value>, "alpha:=", <value>, "beta:=", <value>, "WhichAxis:=", <string("X","Y", or "Z")>, "ReverseDirection:=", <boolean> </pre>
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateBondwire(<Parameters>, <Attributes>)		
Python Example	<pre> oEditor.CreateBondwire(["NAME:BondwireParameters", "WireType:=" , "JEDEC_4Points", "WireDiameter:=" , "0.025mm", "NumSides:=" , "6", "XPadPos:=" , "1.6mm", </pre>		

```

"YPadPos:="          , "-0.2mm",
"ZPadPos:="          , "0mm",
"XDir:="             , "-2.2mm",
"YDir:="             , "-1.4mm",
"ZDir:="             , "0mm",
"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\"",

```

	<pre> "SurfaceMaterialValue:=", "\"\"", "SolveInside:=", True, "ShellElement:=", False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", True, "UseMaterialAppearance:=", False, "IsLightweight:=", False] </pre>

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:=", "" & Chr(34) & "copper" & Chr(34) & "", "Sur- faceMaterialValue:=", _ </pre>

```
"" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _
false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa-
terialAppearance:=", _
false, "IsLightweight:=", false)
```

CreateBox

Creates a box.

UI Access	Draw > Box.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:BoxParameters", "XPosition:=", <string>, "YPosition:=", <string>, "ZPosition:=", <string>, "XSize:=" , <string>, "YSize:=" , <string>, "ZSize:=" , <string>)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateBox(<Parameters>, <Attributes>)
Python Example	oEditor.CreateBox(

```
[ "NAME:BoxParameters",  
  "XPosition:=" , "0.5mm",  
  "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",
```

	<pre> "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False 1) </pre>

VB Syntax	CreateBox <Parameters>, <Attributes>
VB Example	<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:=", "" & Chr(34) & "copper" & Chr(34) & "", "Sur- faceMaterialValue:=", _ "" & Chr(34) & "" & Chr(34) & "", "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	Structured array. <pre>Array("NAME:CircleParameters", "IsCovered:=", <boolean>, "XCenter:=", <value>, "YCenter:=", <value>, "ZCenter:=", <value>, "Radius:=", <value>, "WhichAxis:=", <string> "NumSegments:=", <string containing integer>)</pre>
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateCircle(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateCircle(["NAME:CircleParameters", "IsCovered:=" , True, "XCenter:=" , "5.5mm", "YCenter:=" , "-3mm", "ZCenter:=" , "0mm", "Radius:=" , "0.707106781186548mm",</pre>

```

"WhichAxis:="          , "Z",
"NumSegments:="        , "0"
],
["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
])

```

**VB
Syntax**
CreateCircle <Parameters>, <Attributes>

VB Example	<pre>oEditor.CreateCircle Array("NAME:CircleParameters", "IsCovered:=", true, "XCenter:=", _ "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) & "", "SurfaceMaterialValue:=", _ "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa- terialAppearance:=", _ false, "IsLightweight:=", false)</pre>

CreateCone

Creates a cone.

UI Access	Draw > Cone.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre>Array("NAME:ConeParameters", "XCenter:=", <string>, "YCenter:=", <string>, "ZCenter:=", <string>, "WhichAxis:=", <string>,</pre>

			"Height:=", <string>, "BottomRadius:=", <string>, "TopRadius:=", <string>)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateCone(<Parameters>, <Attributes>)		
Python Example	<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)", </pre>		

	<pre> "Transparency:=" , 0, "PartCoordinateSystem:=", "Global", "UDMId:=" , "", "MaterialValue:=" , "\"copper\"", "SurfaceMaterialValue:=", "\"\"", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False] </pre>

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:=", _ </pre>

```
"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)
```

CreateCutplane

Creates a cutplane.

UI Access	Draw > Plane.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:PlaneParameters", "PlaneBaseX:=", <string>, "PlaneBaseY:=", <string>, "PlaneBaseZ:=", <string>, "PlaneNormalX:=", <string>, "PlaneNormalY:=", <string>), "PlaneNormalZ:=", <string>)
	<AttributesArray>	Array	See: AttributesArray . CreateCutplane only takes the Name and Color attributes.
Return Value	None.		

Python Syntax	CreateCutplane(<Parameters>, <Attributes>)
Python Example	<pre> oEditor.CreateCutplane(["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)"]) </pre>

VB Syntax	CreateCutplane <Parameters>, <Attributes>
VB Example	<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- </pre>

```
or:=", _
"(143 175 143) ")
```

CreateCylinder

Creates a cylinder.

UI Access	Draw > Cylinder.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:CylinderParameters", "XCenter:=", <string>, "YCenter:=", <string>, "ZCenter:=", <string>, "Radius:=", <string>, "Height:=", <string>, "WhichAxis:=", <string>, "NumSides:=", <string containing integer>)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateCylinder(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateCylinder(["NAME:CylinderParameters",</pre>

```
"XCenter:="          , "6mm",
"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",
```

	<pre> "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False l) </pre>

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:=", "" & Chr(34) & "copper" & Chr(34) & "", "SurfaceMaterialValue:=", _ "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMaterialAppearance:=", _ 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:=", <string>, "XCenter:=", <string>, "YCenter:=", <string>, "ZCenter:=", <string>, "MajRadius:=", <string>, "Ratio:=", <string>, "WhichAxis:=", <string>, "NumSegments:=", <string>)</pre>
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateEllipse(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateEllipse(["NAME:EllipseParameters", "IsCovered:=" , True, "XCenter:=" , "0.6mm", "YCenter:=" , "-0.6mm", "ZCenter:=" , "0mm",</pre>

```

    "MajRadius:="          , "0.2mm",
    "Ratio:="              , "7",
    "WhichAxis:="         , "Z",
    "NumSegments:="       , "0"
],
["NAME:Attributes",
    "Name:="              , "Ellip1",
    "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	CreateEllipse <Parameters>, <Attributes>
VB Example	<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, "PartCoordin- 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)</pre>

CreateEquationCurve

Creates an equation-based curve.

UI Access	Draw > Equation-Based Curve.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:EquationBasedCurveParameters",

			"XtFunction:=", <string>, "YtFunction:=", <string>, "ZtFunction:=", <string>, "tStart:=", <string>, "tEnd:=", <string>, "NumOfPointsOnCurve:=", <string>, "Version:=", <integer>), <polylineArray(optional)>
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateEquationCurve(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateEquationCurve(["NAME:EquationBasedCurveParameters", "XtFunction:=" , "1", "YtFunction:=" , "3", "ZtFunction:=" , "32", "tStart:=" , "1", "tEnd:=" , "3", "NumOfPointsOnCurve:=" , "0", "Version:=" , 1,</pre>

```
[ "NAME:PolylineXSection",  
  "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,
```

	<pre> "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False] </pre>

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", "XSec- tionOrient:=", _ "Auto", "XSectionWidth:=", "0", "XSectionTopWidth:=", "0", "XSectionHeight:=", _ "0", "XSectionNumSegments:=", "0", "XSectionBendType:=", "Corner")), Array("NAME:At- tributes", "Name:=", _ "EquationCurve2", "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", "IsMa- terialEditable:=", _ </pre>

```
true, "UseMaterialAppearance:=", false, "IsLightweight:=", false)
```

CreateEquationSurface

Creates an equation-based surface.

UI Access	Draw > Equation-Based Surface.		
Parameters	Name	Type	Description
	<Parameters>	Array	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>)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateEquationSurface(<Parameters>, <Attributes>)
Python Example	<pre> 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:=" , "\"\"", </pre>

	<pre> "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False] </pre>

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:=", _ "" & Chr(34) & "copper" & Chr(34) & "", "SurfaceMaterialValue:=", "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", _ false, "ShellElement:=", false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", _ </pre>

```
true, "UseMaterialAppearance:=", false, "IsLightweight:=", false)
```

CreateHelix

Creates a helix based on a sweep of specified objects.

UI Access	Draw > Helix.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. <pre>Array("NAME:HelixParameters", "XCenter:=" , <string>, "YCenter:=" , <string>, "ZCenter:=" , <string>, "XStartDir:=" , <string>, "YStartDir:=" , <string>, "ZStartDir:=" , <string>, "NumThread:=" , <string>, "RightHand:=" , <boolean>, "RadiusIncrement:=" , <string>, "Thread:=" , <string>)</pre>
Return Value	None.		

Python Syntax	CreateHelix(<SelectionsArray>, <Parameters>)

Python Example	<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>
VB Syntax	CreateHelix <SelectionsArray>, <Parameters>

VB Example	<pre>oEditor.CreateHelix Array("NAME:Selections", "Selections:=", "EquationSurface1", "NewPartsModelFlag:=", _ "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	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:PointParameters", "PointX:=", <value>, "PointY:=", <value>, "PointZ:=", <value>)
	<AttributesArray>	Array	Structured array. See: AttributesArray . CreatePoint takes only the Name and Color attributes.
Return Value	None.		

Python Syntax	CreatePoint(<Parameters>, <Attributes>)

Python Example	<pre>oEditor.CreatePoint(["NAME:PointParameters", "PointX:=" , "0.2mm", "PointY:=" , "-0.2mm", "PointZ:=" , "0mm"], ["NAME:Attributes", "Name:=" , "Point1", "Color:=" , "(143 175 143)"])</pre>

VB Syntax	CreatePoint <Parameters>, <Attributes>
VB Example	<pre>oEditor.CreatePoint Array("NAME:PointParameters", "PointX:=" , "0.2mm", "PointY:=" , "-0.2mm",</pre>

	<pre> "PointZ:=" , "0mm") , Array("NAME:Attributes", "Name:=" , "Point1", "Color:=" , "(143 175 143)") </pre>

CreatePolyline

Creates a polyline.

UI Access	Draw > Line.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre> Array("NAME:PolylineParameters", "IsPolylineCovered:=", <bool>, "IsPolylineClosed:=", <bool>, <PolylinePointsArray>, <PolylineSegmentsArray>) </pre>
	<PolylinePointsArray>	Array	Array("NAME:PolylinePoints", <OnePointArray>, <OnePointArray>, ...)
	<OnePointArray>	Array	Array("NAME:PLPoint", <pre> "X:=", <value>, "Y:=", <value>, "Z:=", <value>)) </pre>
	<PolylineSegmentsArray>	Array	<PolylineSegmentsArray> <pre> Array("NAME:PolylineSegments", </pre>

			<OneSegmentArray>, <OneSegmentArray>, ...)
	<OneSegmentArray>	Array	Array("NAME:PLSegment", "SegmentType:=", <"Line", "Arc", "Spline", or "AngularArc">, "StartIndex:=", <value>, "NoOfPoints:=", <value>)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreatePolyline(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreatePolyline(["NAME:PolylineParameters", "IsPolylineCovered:=" , True, "IsPolylineClosed:=" , False, ["NAME:PolylinePoints", ["NAME:PLPoint", "X:=" , "20000mm", "Y:=" , "-20000mm", "Z:=" , "0mm"], ["NAME:PLPoint",</pre>

```
"X:="                , "-90000mm",
"Y:="                , "20000mm",
"Z:="                , "0mm"
],
["NAME:PLPoint",
"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"  
]],  
["NAME:Attributes",  
  "Name:="              , "Polyline1",  
  "Flags:="             , "",  
  "Color:="             , "(143 175 143)",  
  "Transparency:="      , 0,  
  "PartCoordinateSystem:=", "Global",  
  "UDMId:="            , "",  
  "MaterialValue:="     , "\"copper\"",  
  "SurfaceMaterialValue:=", "\"\"",  
  "SolveInside:="       , False,
```

	<pre> "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False]) </pre>

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", "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, "PartCoordinateSystem:=", _ "Global", "UDMId:=", "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", </pre>

	<pre>"SurfaceMaterialValue:=", _ "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa- terialAppearance:=", _ false, "IsLightweight:=", false)</pre>

CreateRectangle

Creates a rectangle.

UI Access	Draw > Rectangle.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre>Array("NAME:RectangleParameters", "IsCovered:=" , <boolean>, "XStart:=" , <string>, "YStart:=" , <string>, "ZStart:=" , <string>, "Width:=" , <string>, "Height:=" , <string>, "WhichAxis:=" , <string "X", "Y", or "Z">)</pre>
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateRectangle(<Parameters>, <Attributes>)
Python Example	<pre> oEditor.CreateRectangle(["NAME:RectangleParameters", "IsCovered:=" , True, "XStart:=" , "-80000mm", "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:=" , "\"\"", </pre>

	<pre> "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False] </pre>

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:=", "" & Chr(34) & "copper" & Chr(34) & "", "SurfaceMaterialValue:=", _ "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMaterialAppearance:=", _ </pre>

```
false, "IsLightweight:=", false)
```

CreateRegion

Creates a region containing the design.

UI Access	Draw > Region.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre>Array("NAME:RegionParameters", "+XPaddingType:=", <string "Percentage Offset", "Absolute Offset", or "Absolute Position">, "+XPadding:=", <string X value>, "-XPaddingType:=", <string "Percentage Offset", "Absolute Offset", or "Absolute Position">, "-XPadding:=", <string -X value>, "+YPaddingType:=", <string "Percentage Offset", "Absolute Offset", or "Absolute Position">, "+YPadding:=", <string Y value>, "-YPaddingType:=", <string "Percentage Offset", "Absolute Offset", or "Absolute Position">, "-YPadding:=", <string -Y value>, "+ZPaddingType:=", <string "Percentage Offset", "Absolute Offset", or "Absolute Position">, "+ZPadding:=", <string Z value>, "-ZPaddingType:=", <string "Percentage Offset",</pre>

			"Absolute Offset", or "Absolute Position">, "-ZPadding:=", <string -Z value>)
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateRegion(<Parameters>, <Attributes>)
Python Example	<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"])</pre>

```

],
["NAME:Attributes",
    "Name:="                , "Region",
    "Flags:="                , "Wireframe#",
    "Color:="                , "(143 175 143)",
    "Transparency:="         , 0,
    "PartCoordinateSystem:=" , "Global",
    "UDMId:="                , "",
    "MaterialValue:="        , "\"vacuum\"",
    "SurfaceMaterialValue:=" , "\"\"",
    "SolveInside:="          , False,
    "ShellElement:="         , False,
    "ShellElementThickness:=" , "nan ",
    "IsMaterialEditable:="    , True,
    "UseMaterialAppearance:=" , False,
    "IsLightweight:="         , False
])

```

VB Syntax	CreateRegion <Parameters>, <Attributes>
VB Examp-	oEditor.CreateRegion Array("NAME:RegionParameters", "+XPaddingType:=", _ "Percentage Offset", "+XPadding:=", "0", "-XPaddingType:=", "Percentage Offset", "-XPad-

le	<pre> 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:=", _ "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) </pre>

CreateRegularPolygon

Creates a regular polygon.

UI Access	Draw > Regular Polygon.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:RegularPolygonParameters", "IsCovered:=" , <boolean>, "XCenter:=" , <string>,

			"YCenter:=" , <string>, "ZCenter:=" , <string>, "XStart:=" , <string>, "YStart:=" , <string>, "ZStart:=" , <string>, "NumSides:=" , <string containing number greater than 2>, "WhichAxis:=" , <string "X", "Y", or "Z")
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateRegularPolygon(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateRegularPolygon(["NAME:RegularPolygonParameters", "IsCovered:=" , True, "XCenter:=" , "-70000mm", "YCenter:=" , "-100000mm", "ZCenter:=" , "0mm", "XStart:=" , "-50000mm", "YStart:=" , "-80000mm", "ZStart:=" , "0mm", "NumSides:=" , "12",</pre>

```
"WhichAxis:="          , "Z"
],
["NAME:Attributes",
  "Name:="              , "Polygon1",
  "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 Syn-	CreateRegularPolygon <Parameters>, <Attributes>

tax	
VB Example	<pre>oEditor.CreateRegularPolygon Array("NAME:RegularPolygonParameters", "IsCovered:=", _ true, "XCenter:=", "-60000mm", "YCenter:=", "40000mm", "ZCenter:=", "0mm", "XStart:=", _ "-50000mm", "YStart:=", "50000mm", "ZStart:=", "0mm", "NumSides:=", "8", "WhichAxis:=", _ "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)</pre>

CreateRegularPolyhedron

Creates a regular polyhedron.

UI Access	Draw > Regular Polyhedron.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre>Array("NAME:PolyhedronParameters", "XCenter:=" , <string>, "YCenter:=" , <string>,</pre>

			"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: AttributesArray .
Return Value	None.		

Python Syntax	CreateRegularPolyhedron(<Parameters>, <Attributes>)
Python Example	<pre>oEditor.CreateRegularPolyhedron(["NAME:PolyhedronParameters", "XCenter:=" , "40000mm", "YCenter:=" , "-80000mm", "ZCenter:=" , "0mm", "XStart:=" , "50000mm", "YStart:=" , "-70000mm", "ZStart:=" , "0mm",</pre>

```

    "Height:="                , "50000mm",
    "NumSides:="              , "8",
    "WhichAxis:="            , "Z"
],
["NAME:Attributes",
    "Name:="                  , "RegularPolyhedron1",
    "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	CreateRegularPolyhedron <Parameters>, <Attributes>
VB Example	<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:=", "", "Col- or:=", _ "(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)</pre>

CreateSphere

Creates a sphere.

UI Access	Draw > Sphere.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:SphereParameters",

			"XCenter:=", <string>, "YCenter:=", <string>, "ZCenter:=", <string>, "Radius:=", <string>
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateSphere(<Parameters>, <Attributes>)
Python Example	<pre> 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", </pre>

	<pre> "UDMId:=" , "", "MaterialValue:=" , "\"copper\"", "SurfaceMaterialValue:=", "\"\"", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False] </pre>

VB Syntax	CreateSphere <Parameters>, <Attributes>
VB Example	<pre> oEditor.CreateSphere Array("NAME:SphereParameters", "XCenter:=", "-120000mm", "YCenter:=", _ "90000mm", "ZCenter:=", "0mm", "Radius:=", "10000mm"), Array("NAME:Attributes", "Name:=", _ "Sphere2", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", _ "Global", "UDMId:=", "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "SurfaceMaterialValue:=", _ "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _ </pre>

```
false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMaterialAppearance:=", _
false, "IsLightweight:=", false)
```

CreateSpiral

Creates a spiral by sweeping the specified object(s).

UI Access	Draw > Spiral.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. <pre>Array("NAME:SpiralParameters", "XCenter:=" , <string>, "YCenter:=" , <string>, "ZCenter:=" , <string>, "YStartDir:=" , <string>, "ZStartDir:=" , <string>, "NumThread:=" , <string>, "RightHand:=" , <boolean>, "RadiusIncrement:=" , <string>)</pre>
Return Value	None.		

Python Syntax	CreateSpiral(<Parameters>, <Attributes>)

Python Example	<pre>oEditor.CreateSpiral(["NAME:Selections", "Selections:=", "Polygon2", "NewPartsModelFlag:=", "Model"], ["NAME:SpiralParameters", "XCenter:=", "-70000mm", "YCenter:=", "50000mm", "ZCenter:=", "0mm", "XStartDir:=", "-60000mm", "YStartDir:=", "-10000mm", "ZStartDir:=", "0mm", "NumThread:=", "1", "RightHand:=", True, "RadiusIncrement:=", "1mm"])</pre>

VB Syntax	CreateSpiral <Parameters>, <Attributes>
VB Example	<pre>oEditor.CreateSpiral Array("NAME:Selections", "Selections:=", "Rectangle2", "NewPartsModelFlag:=", _</pre>

```
"Model"), Array("NAME:SpiralParameters", "XCenter:=", "30000mm", "YCenter:=", _
"-30000mm", "ZCenter:=", "0mm", "XStartDir:=", "-90000mm", "YStartDir:=", _
"-30000mm", "ZStartDir:=", "0mm", "NumThread:=", "1", "RightHand:=", false, "Radi-
usIncrement:=", _
"1mm")
```

CreateTorus

Creates a torus.

UI Access	Draw > Torus.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre>Array("NAME:TorusParameters", "XCenter:=" , <string>, "YCenter:=" , <string>, "ZCenter:=" , <string>, "MajorRadius:=" , <string>, "MinorRadius:=" , <string>, "WhichAxis:=" , <string "X", "Y", or "Z">)</pre>
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateTorus(<Parameters>, <Attributes>)

Python Example

```
oEditor.CreateTorus(  
  ["NAME:TorusParameters",  
    "XCenter:="          , "0.6mm",  
    "YCenter:="          , "-0.6mm",  
    "ZCenter:="          , "0mm",  
    "MajorRadius:="      , "0.365028153987289mm",  
    "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,
```

	<pre>"ShellElementThickness:=", "0mm", "IsMaterialEditable:=", True, "UseMaterialAppearance:=", False, "IsLightweight:=", False])</pre>

VB Syntax	CreateTorus <Parameters>, <Attributes>
VB Example	<pre>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, "PartCoordin- ateSystem:=", _ "Global", "UDMId:=", "", "MaterialValue:=", "" & Chr(34) & "copper" & Chr(34) & "", "SurfaceMaterialValue:=", _ "" & Chr(34) & "" & Chr(34) & "", "SolveInside:=", false, "ShellElement:=", _ false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=", true, "UseMa- terialAppearance:=", _ false, "IsLightweight:=", false)</pre>

CreateUserDefinedPart

Creates a user-defined part.

UI Access	Draw > User-Defined Primitive > [Part]

Parameters	Name	Type	Description
	<code><Parameters></code>	Array	Structured array. <pre>Array("NAME:UserDefinedPrimitiveParameters", "DllName:=" , <string>, "Version:=" , <string>, "NoOfParameters:=" , <integer>, "Library:=" , <string>, <paramVectorArray>)</pre>
	<code><paramVectorArray></code>	Array	Structured array containing arrays for each pair: <pre>Array("NAME:ParamVector", <pair>, <pair>, <pair>,...)</pre>
	<code><pair></code>	Array	Structured array: <pre>Array("NAME:Pair", "Name:=" , <string>, "Value:=" , <string>)</pre>
	<code><Attributes></code>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateUserDefinedPart(<Parameters>, <Attributes>)
Python Example	oEditor.CreateUserDefinedPart (

```
[ "NAME:UserDefinedPrimitiveParameters",  
  "DllName:="          , "RMxpert/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:=",  
  "Value:=",  
],  
["NAME:Pair",  
  "Name:=",  
  "Value:=",  
],  
["NAME:Pair",  
  "Name:=",  
  "Value:=",  
]  
]  
],  
["NAME:Attributes",  
  "Name:=",  
  "Flags:=",  
  "Color:=",  
  "Transparency:=",  
  "PartCoordinateSystem:=",  
  "UDMId:="
```

	<pre> "MaterialValue:=" , "\"copper\"", "SurfaceMaterialValue:=", "\"\"", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=", False, "IsLightweight:=" , False]) </pre>

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 by saving it to a new name so that its properties are accessible. To change the properties on an exist-
ing component without changing its name, see [ChangeProperty](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<compName>	String	Component name.
	<Parameters>	Array	Structured array.
			<pre> Array("NAME:EditComponentParametersData", "NewComponentName:=", <string>, "GeometryParameters:=", <string>, </pre>

			"MaterialParameters:=", <string>, "DesignParameters:=", <string>, <ComponentMeshing>, <Excitations>)
	<ComponentMeshing>	Array	Structured array. Array("NAME:Component Meshing", "MeshAssembly:=", <boolean>)
	<Excitations>	Array	Structured array containing array of suppressed excitations. Array("NAME:Excitations", "Suppressed:=", <array>)
Return Value	None.		

Python Syntax	Edit3DComponent (<compName>, <Parameters>)
Python Example	<pre>oEditor.Edit3DComponent("Connector1", ["NAME:EditComponentParametersData", "NewComponentName:=", "Connector2", "GeometryParameters:=", "", "MaterialParameters:=", "", "DesignParameters:=", "",</pre>

	<pre> ["NAME:Component Meshing", "MeshAssembly:=", False], ["NAME:Excitations", "Suppressed:=", []]]) </pre>

VB Syntax	Edit3DComponent <compName>, <Parameters>
VB Example	<pre> oEditor.Edit3DComponent "Connector1", Array("NAME:EditComponentParametersData", "NewComponentName:=", "Connector2", "GeometryParameters:=", "", "MaterialParameters:=", "", "DesignParameters:=", "", Array("NAME:Component Meshing", "MeshAssembly:=", false), Array("NAME:Excitations", "Suppressed:=", Array())) </pre>

Edit3DComponentDefinition

Edits definitions of a specified 3D component.

UI Access	N/A		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:EditComponentParametersData", "OriginalComponentName:=", <string>, "NewComponentName:=", <string>, "GeometryParameters:=", <string>, "MaterialParameters:=", <string>, "DesignParameters:=", <string>, <ComponentMeshing>, <Excitations>)
	<ComponentMeshing>	Array	Structured array. Array("NAME:Component Meshing", "MeshAssembly:=", <boolean>)
	<Excitations>	Array	Structured array containing array of suppressed excitations. Array("NAME:Excitations", "Suppressed:=", <array>)
Return Value	None.		

Python Syntax	Edit3DComponent (<Parameters>)
Python Example	<pre> oEditor.Edit3DComponent (["NAME:EditComponentParametersData", "OriginalComponentName:=", "Connector1", "NewComponentName:=", "Connector2", "GeometryParameters:=", "", "MaterialParameters:=", "", "DesignParameters:=", "", ["NAME:Component Meshing", "MeshAssembly:=", False], ["NAME:Excitations", "Suppressed:=", []]]) </pre>
VB Syntax	Edit3DComponent <Parameters>
VB Example	<pre> oEditor.Edit3DComponent Array ("NAME:EditComponentParametersData", "OriginalComponentName:=", "Connector1", "NewComponentName:=", "Connector2", "GeometryParameters:=", "", "MaterialParameters:=", "", </pre>

	<pre> "DesignParameters:=", "", Array("NAME:Component Meshing", "MeshAssembly:=", false), Array("NAME:Excitations", "Suppressed:=", Array())) </pre>

EditPolyline

Modifies a specified polyline. See: [CreatePolyline](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. <pre> Array("NAME:PolylineParameters", "IsPolylineCovered:=", <bool>, "IsPolylineClosed:=", <bool>, <PolylinePointsArray>, <PolylineSegmentsArray>) </pre>
	<PolylinePointsArray>	Array	Array("NAME:PolylinePoints", <OnePointArray>, <OnePointArray>, ...)
	<OnePointArray>	Array	Array("NAME:PLPoint", "X:=", <value>,

			"Y:=", <value>, "Z:=", <value>))
	<PolylineSegmentsArray>	Array	<PolylineSegmentsArray> Array("NAME:PolylineSegments", <OneSegmentArray>, <OneSegmentArray>, ...)
	<OneSegmentArray>	Array	Array("NAME:PLSegment", "SegmentType:=", <"Line", "Arc", "Spline", or "AngularArc">, "StartIndex:=", <value>, "NoOfPoints:=", <value>)
Return Value	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:=" , "20000mm",</pre>

```
"Y:="                , "-20000mm",
"Z:="                , "0mm"
],
["NAME:PLPoint",
  "X:="                , "-90000mm",
  "Y:="                , "20000mm",
  "Z:="                , "0mm"
],
["NAME:PLPoint",
  "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"
]]
)

```

VB Syntax	EditPolyline <SelectionsArray>, <Parameters>
VB Example	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, "PartCoordin-
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)
```

Get3DComponentDefinitionNames

Gets names of 3D component definitions.

UI Access	N/A
Parameters	None.

Return Value	Array of strings containing component definition names.

Python Syntax	Get3DComponentDefinitionNames()
Python Example	<code>oEditor.Get3DComponentDefinitionNames()</code>

VB Syntax	Get3DComponentDefinitionNames()
VB Example	<pre>Dim defNames defNames = oEditor.Get3DComponentDefinitionNames()</pre>

Get3DComponentInstanceNames

Returns instance names of 3D component definitions.

UI Access	N/A		
Parameters	Name	Type	Description
	<DefinitionName>	String	Definition name.
Return Value	Array containing instance names.		

Python Syntax	Get3DComponentInstanceNames(<DefinitionName>)
Python Example	<code>oEditor.Get3DComponentInstanceNames("Connector")</code>

VB Syntax	Get3DComponentInstanceNames <DefinitionName>

VB Example	<code>oEditor.Get3DComponentInstanceNames "Connector"</code>

Get3DComponentMaterialNames

Returns material names for a specified *.a3dcomp format 3D component.

UI Access	N/A		
Parameters	Name	Type	Description
	<InstanceName>	String	Component instance name.
Return Value	Array containing material names.		

Python Syntax	<code>Get3DComponentMaterialNames(<InstanceName>)</code>
Python Example	<code>oEditor.Get3DComponentMaterialNames("Connector1.a3dcomp")</code>

VB Syntax	<code>Get3DComponentMaterialNames <InstanceName></code>
VB Example	<code>oEditor.Get3DComponentMaterialNames "Connector1.a3dcomp"</code>

Get3DComponentMaterialProperties

Returns material properties for a specified 3D component.

UI Access	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>

Get3DComponentParameters

Returns parameters for a specified 3D component.

UI Access	N/A		
Parameters	Name	Type	Description
	<compName>	String	3D component name.
Return Value	Array containing component parameters.		

Python Syntax	Get3DComponentParameters(<compName>)
Python Example	<code>oEditor.Get3DComponentParameters('Connector')</code>

VB Syntax	Get3DComponentParameters <compName>
VB Example	<code>oEditor.Get3DComponentParameters "Connector"</code>

Get3DComponentPartNames

Returns part names for a specified 3D component.

UI Access	N/A		
Parameters	Name	Type	Description
	<InstanceName>	String	3D component instance.
Return Value	Array containing part names.		

Python Syntax	Get3DComponentParameters(<InstanceName>)
Python Example	<code>oEditor.Get3DComponentPartNames('Connector')</code>

VB Syntax	Get3DComponentParameters <InstanceName>
VB Example	<code>oEditor.Get3DComponentPartNames "Connector"</code>

Insert3DComponent

Inserts a 3D component.

UI Access	Draw > 3D Component Library > Browse > [Component].		
Parameters	Name	Type	Description
	<ComponentData>	Array	Structured array. <code>Array("NAME:InsertComponentData", "Parameters:=", <string>,</code>

			"TargetCS:=", <string>, "ComponentFile:=", <string filepath>)
Return Value	None.		

Python Syntax	Insert3DComponent(<ComponentData>)
Python Example	<pre>oEditor.Insert3DComponent(["NAME:InsertComponentData", "Parameters:=", "", "TargetCS:=", "Global", "ComponentFile:=", "C:\tmp\Connector.a3dcomp"])</pre>

VB Syntax	Insert3DComponent <ComponentData>
VB Example	<pre>oEditor.Insert3DComponent Array("NAME:InsertComponentData", "Parameters:=", "", "TargetCS:=", "Global", "ComponentFile:=", "C:\tmp\Connector.a3dcomp")</pre>

InsertComponent

Inserts a component.

UI Access	N/A

Parameters	Name	Type	Description
	<i><ComponentData></i>	Array	Structured array. Array("NAME:InsertComponentData", "Parameters:=", <string>, "TargetCS:=", <string>, "ComponentFile:=", <string filepath>)
Return Value	None.		

Python Syntax	InsertComponent(<i><ComponentData></i>)
Python Example	<pre>oEditor.InsertComponent(["NAME:InsertComponentData", "Parameters:=", "", "TargetCS:=", "Global", "ComponentFile:=", "C:\tmp\Connector.a3dcomp"])</pre>

VB Syntax	InsertComponent <i><ComponentData></i>
VB Example	<pre>oEditor.InsertComponent Array("NAME:InsertComponentData", "Parameters:=", "", "TargetCS:=", "Global",</pre>

```
"ComponentFile:=", "C:\tmp\Connector.a3dcomp")
```

InsertPolylineSegment

Inserts a polyline segment before or after a specified existing segment.

UI Access	Draw > Line Segment > [Selection].		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre>Array("NAME:Insert Polyline Segment", "Selections:=" , <string>, "Segment Indices:=" , <array containing integers>, "At Start:=" , <boolean>, "SegmentType:=" , <string "Line", "Arc", "Spline", or "AngularArc">, <PolylinePointsArray></pre>
	<PolylinePointsArray>	Array	Structured array. See: CreatePolyline .
Return Value	None.		

Python Syntax	InsertPolylineSegment(<Parameters>)
Python Example	<pre>oEditor.InsertPolylineSegment(["NAME:Insert Polyline Segment", "Selections:=" , "Polyline1:CreatePolyline:1", "Segment Indices:=" , [0],</pre>

```
"At Start:="          , True,
"SegmentType:="       , "Line",
["NAME:PolylinePoints",
  ["NAME:PLPoint",
    "X:="              , "1.1mm",
    "Y:="              , "0.8mm",
    "Z:="              , "0mm"
  ],
  ["NAME:PLPoint",
    "X:="              , "0.6mm",
    "Y:="              , "-0.8mm",
    "Z:="              , "0mm"
  ]
]
]
])
```

VB Syntax	InsertPolylineSegment <Parameters>
VB Example	<pre>oEditor.InsertPolylineSegment Array("NAME:Insert Polyline Segment", "Selections:=", _ "Polyline1:CreatePolyline:1", "Segment Indices:=", Array(1), "At Start:=", _</pre>

```

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")))

```

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.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<PathSweepParametersArray>	Array	Array("NAME:PathSweepParameters", "DraftAngle:=", <value>, "DraftType:=", <string>, "CheckFaceFaceIntersection:=", <bool>, "TwistAngle:=", <value>) Possible values for DraftType are "Extended", "Round", and "Natural".
Return Value	None.		

Python Syntax	SweepAlongPath(<SelectionsArray>, <PathSweepParametersArray>)

Python Example	<pre>oEditor.SweepAlongPath(["NAME:Selections", "Selections:=" , "Rectangle1,Polyline1", "NewPartsModelFlag:=" , "Model"], ["NAME:PathSweepParameters", "DraftAngle:=" , "0deg", "DraftType:=" , "Round", "CheckFaceFaceIntersection:=", False, "TwistAngle:=" , "0deg"])</pre>

VB Syntax	SweepAlongPath <SelectionsArray>, <PathSweepParametersArray>
VB Example	<pre>oEditor.SweepAlongPath Array("NAME:Selections", "Selections:=", "Polygon1,Polyline1"),_ Array("NAME:PathSweepParameters", _</pre>

	<pre>"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.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<VecSweepParametersArray>	Array	Array("NAME:VectorSweepParameters", "DraftAngle:=", <value>, "DraftType:=", <string>, "CheckFaceFaceIntersection:=", <bool>, "SweepVectorX:=", <value> "SweepVectorY:=", <value> "SweepVectorZ:=", <value>) Possible values for DraftType are "Extended", "Round", and "Natural".
Return Value	None.		

Python Syntax	SweepAlongVector(<SelectionsArray>, <VecSweepParametersArray>)
Python Example	<pre>oEditor.SweepAlongVector(["NAME:Selections", "Selections:=" , "Rectangle1", "NewPartsModelFlag:=" , "Model"], ["NAME:VectorSweepParameters", "DraftAngle:=" , "0deg", "DraftType:=" , "Round", "CheckFaceFaceIntersection:=", False, "SweepVectorX:=" , "0mm" "SweepVectorY:=" , "0mm" "SweepVectorZ:=" , "12mm"])</pre>

VB Syntax	SweepAlongVector <SelectionsArray>, <VecSweepParametersArray>
VB Example	<pre>oEditor.SweepAlongPath</pre>

```

Array("NAME:Selections",_
    "Selections:=", "Rectangle1",_
    "NewPartsModelFlag:=" , "Model")
Array("NAME:VectorSweepParameters", _
    "DraftAngle:=", "0deg",_
    "DraftType:=", "Round",_
    "CheckFaceFaceIntersection:=", False,_
    "SweepVectorX:=" , "0mm",_
    "SweepVectorY:=" , "0mm",_
    "SweepVectorZ:=" , "12mm")

```

SweepAroundAxis

Sweeps the specified 1D or 2D parts around an axis.

UI Access	Draw > Sweep > Around Axis.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<AxisSweepParametersArray>	Array	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	<pre> oEditor.SweepAroundAxis(["NAME:Selections", "Selections:=" , "Rectangle1", "NewPartsModelFlag:=" , "Model"], ["NAME:AxisSweepParameters", "DraftAngle:=" , "0deg", "DraftType:=" , "Round", "CheckFaceFaceIntersection:=", False, "SweepAxis:=" , "X" "SweepAngle:=" , "360deg"]) </pre>

	<pre>"NumOfSegments:=" , "12"])</pre>

VB Syntax	SweepAroundAxis <SelectionsArray>, <AxisSweepParametersArray>
VB Example	<pre>oEditor.SweepAroundAxis Array("NAME:Selections",_ "Selections:=", "Rectangle1",_ "NewPartsModelFlag:=" , "Model") Array("NAME:AxisSweepParameters", _ "DraftAngle:=", "0deg",_ "DraftType:=", "Round",_ "CheckFaceFaceIntersection:=", False,_ "SweepAxis:=" , "X",_ "SweepAngle:=" , "360deg",_ "NumOfSegments:=" , "12")</pre>

SweepFacesAlongNormal

Sweep the specified face(s) along normal.

UI Access	Modeler > Surface > Sweep Faces Along Normal		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .

	<code><parameters></code>	Array	Structured array. Array("NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", <faceIDarray>, "LengthOfSweep:=", "<value><units>")
Return Value	None		

Python Syntax	SweepFacesAlongNormal(<SelectionsArray> <parameters>)
Python Example	<pre>oEditor.SweepFacesAlongNormal(["NAME:Selections", "Selections:=", "Rectangle1", "NewPartsModelFlag:=", "Model"], ["NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", [183], "LengthOfSweep:=", "0.1mm"])</pre>

VB Syntax	SweepFacesAlongNormal <SelectionsArray> <parameters>

VB Example	<pre> oEditor.SweepFacesAlongNormal Array("NAME:Selections", "Selections:=", "Rectangle1", "NewPartsModelFlag:=", "Model"), Array("NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", Array(57), "LengthOfSweep:=", "0.5mm")) </pre>

SweepFacesAlongNormalWithAttributes

Sweep a face along normal, and specify attributes of the new object.

UI Access	Modeler > Surface > Sweep Faces Along Normal		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<parameters>	Array	Array("NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", <faceIDarray>, "LengthOfSweep:=", "<value><units>")
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None		

Python Syntax	SweepFacesAlongNormalWithAttributes(<SelectionsArray>, <parameters>, <AttributesArray>)
Python Example	<pre>oEditor.SweepFacesAlongNormalWithAttributes(["NAME:Selections", "Selections:=", "Rectangle1", "NewPartsModelFlag:=", "Model"], ["NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", [183], "LengthOfSweep:=", "0.1mm"], ["NAME:Attributes", "Name:=", "Box3", "Flags:=", "", "Color:=", "(143 175 143)", "Transparency:=", 0, "PartCoordinateSystem:=", "Global", "UDMId:=", "", "MaterialValue:=", "\"copper\"", "SurfaceMaterialValue:=", "\"\"", "SolveInside:=", False, "ShellElement:=", False,</pre>

	<pre>"ShellElementThickness:=", "0mm", "IsMaterialEditable:=", True, "UseMaterialAppearance:=", False, "IsLightweight:=", False])</pre>

VB Syntax	SweepFacesAlongNormalWithAttributes <SelectionsArray>, <parameters>, <AttributesArray>
VB Example	<pre>oEditor.SweepFacesAlongNormalWithAttributes Array("NAME:Selections", "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",</pre>

	<pre>"UDMId:=" , "", "MaterialValue:=" , "\"copper\"", "SurfaceMaterialValue:=", "\"\"", "SolveInside:=" , false, "ShellElement:=" , false, "ShellElementThickness:=", "0mm", "IsMaterialEditable:=" , true, "UseMaterialAppearance:=", false, "IsLightweight:=" , false))</pre>

UpdateComponentDefinition

Updates a 3D component's definition.

UI Access	Draw > 3D Component Library > Definitions.		
Parameters	Name	Type	Description
	<data>	Array	Structured array. Array("NAME:UpdateDefinitionData", "DefinitionNames:=", <string>, "Passwords:=", <array of strings>)
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	<pre>oEditor.UpdateComponentDefinition Array("NAME:UpdateDefinitionData", "DefinitionNames:=", " Connector, Magic_Tee", "Passwords:=", Array("", ""))</pre>

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		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	Copy(<SelectionsArray>)		
Python Example	<pre>oEditor.Copy(["NAME:Selections", "Selections:=", "Box1"])</pre>		

VB Syntax	Copy <SelectionsArray>		
VB Example	<pre>oEditor.Copy Array("NAME:Selections",</pre>		

	"Selections:=", "Box1")

DeletePolylinePoint

Deletes either a start point or an end point from an existing polyline segment.

UI Access	Edit > Delete [Start/End] Point		
Parameters	Name	Type	Description
	<DeletePointArray>	Array	Structured array. Array("NAME:Delete Point", "Selections:=", <string "<PolylineName>:<PolylineAction>:<int>">, "Segment Index:=", <integer>, "At Start:=", <bool True for start point; False for end point>)
Return Value	None.		

Python Syntax	DeletePolylinePoint (<DeletePointArray>)
Python Example	<pre>oEditor.DeletePolylinePoint(["NAME:Delete Point", "Selections:=", "Polyline1:CreatePolyline:1", "Segment Index:=", 1, "At Start:=", True])</pre>

VB Syntax	DeletePolylinePoint <DeletePointArray>

VB Example	<pre>oEditor.DeletePolylinePoint Array("NAME:Delete Point", "Selections:=", "Polyline1:CreatePolyline:1", "Segment Index:=", 1, "At Start:=", True)</pre>

DuplicateAlongLine

Duplicates specified parts along a line.

UI Access	Edit > Duplicate > Along Line.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. <pre>Array("NAME:Selections", "Selections:=" , <string>, "NewPartsModelFlag:=" , <string>)</pre>
	<ParametersArray>	Array	Structured array. <pre>Array("NAME:DuplicateToAlongLineParameters", "CreateNewObjects:=" , <boolean>, "XComponent:=" , <string>, "YComponent:=" , <string>, "ZComponent:=" , <string>, "NumClones:=" , <string containing number greater than 1>)</pre>
	<OptionsArray>	Array	Structured array. <pre>Array("NAME:Options",</pre>

			"DuplicateAssignments:=", <boolean>)
	<CreateGroup>	Array	Optional. Structured array. Array("CreateGroupsForNewObjects:=", <boolean>)
Return Value	None.		

Python Syntax	DuplicateAlongLine (<SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>)
Python Example	<pre> oEditor.DuplicateAlongLine(["NAME:Selections", "Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model"], ["NAME:DuplicateToAlongLineParameters", "CreateNewObjects:=" , False, "XComponent:=" , "1mm", "YComponent:=" , "-0.7mm", "ZComponent:=" , "0mm", "NumClones:=" , "2"], ["NAME:Options", "DuplicateAssignments:=", False], </pre>

	<pre>["CreateGroupsForNewObjects:=", False])</pre>

VB Syntax	DuplicateAlongLine <SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>
VB Example	<pre>oEditor.DuplicateAlongLine Array("NAME:Selections", "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)</pre>

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", "CreateNewObjects:=" , <boolean>, "WhichAxis:=" , <string>, "AngleStr:=" , <string>, "NumClones:=" , <string containing number greater than 1>)
	<OptionsArray>	Array	Structured array. Array("NAME:Options", "DuplicateAssignments:=" , <boolean>)
	<CreateGroup>	Array	Optional. Structured array. Array("CreateGroupsForNewObjects:=" , <boolean>)
Return Value	None.		

Python Syntax	DuplicateAroundAxis (<SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>)
Python Example	oEditor.DuplicateAroundAxis(

	<pre>["NAME:Selections", "Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model"], ["NAME:DuplicateAroundAxisParameters", "CreateNewObjects:=" , True, "WhichAxis:=" , "Z", "AngleStr:=" , "90deg", "NumClones:=" , "2"], ["NAME:Options", "DuplicateAssignments:=", False], ["CreateGroupsForNewObjects:=", False]]</pre>

VB Syntax	<code>DuplicateAroundAxis <SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup></code>
VB Example	<pre>oEditor.DuplicateAroundAxis Array("NAME:Selections",</pre>

	<pre> "Selections:=", "Box1", "NewPartsModelFlag:=", "Model") Array("NAME:DuplicateAroundAxisParameters", "CreateNewObjects:=", true, "WhichAxis:=", "Z", "AngleStr:=", "90deg", "NumClones:=", "2") Array("NAME:Options", "DuplicateAssignments:=", false) Array("CreateGroupsForNewObjects:=", false) </pre>

DuplicateMirror

Duplicates specified parts according to a mirror plane.

UI Access	Edit > Duplicate > Mirror.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. <pre> Array("NAME:Selections", "Selections:=" , <string>, "NewPartsModelFlag:=" , <string>) </pre>
	<ParametersArray>	Array	Structured array. <pre> Array("NAME:DuplicateToMirrorParameters", "DuplicateMirrorBaseX:=" , <string>, </pre>

			"DuplicateMirrorBaseY:=", <string>, "DuplicateMirrorNormalX:=", <string>, "DuplicateMirrorNormalY:=", <string>, "DuplicateMirrorNormalZ:=", <string>)
	<OptionsArray>	Array	Structured array. Array("NAME:Options", "DuplicateAssignments:=", <boolean>)
	<CreateGroup>	Array	Optional. Structured array. Array("CreateGroupsForNewObjects:=", <boolean>)
Return Value	None.		

Python Syntax	DuplicateMirror (<SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>)
Python Example	<pre> oEditor.DuplicateMirror(["NAME:Selections", "Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model"], ["NAME:DuplicateToMirrorParameters", "DuplicateMirrorBaseX:=", "-0.4mm", "DuplicateMirrorBaseY:=", "-1.2mm", </pre>

	<pre> "DuplicateMirrorBaseZ:=", "0mm", "DuplicateMirrorNormalX:=", "0.124034734589208mm", "DuplicateMirrorNormalY:=", "0.992277876713668mm", "DuplicateMirrorNormalZ:=", "0mm"], ["NAME:Options", "DuplicateAssignments:=", False], ["CreateGroupsForNewObjects:=", False]) </pre>

VB Syntax	DuplicateMirror <SelectionsArray>, <ParametersArray>, <OptionsArray>, <CreateGroup>
VB Example	<pre> oEditor.DuplicateMirror Array("NAME:Selections", "Selections:=", "Box1", "NewPartsModelFlag:=", "Model") Array("NAME:DuplicateToMirrorParameters", "DuplicateMirrorBaseX:=", "-0.4mm", "DuplicateMirrorBaseY:=", "-1.2mm", "DuplicateMirrorBaseZ:=", "0mm", "DuplicateMirrorNormalX:=", "0.124034734589208mm", </pre>

	<pre>"DuplicateMirrorNormalY:=", "0.992277876713668mm", "DuplicateMirrorNormalZ:=", "0mm") Array("NAME:Options", "DuplicateAssignments:=", false) Array("CreateGroupsForNewObjects:=", false)</pre>

Mirror

Mirrors specified part(s).

UI Access	Edit > Arrange > Mirror.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<MirrorParameters>	Array	Structured array. <pre>Array("NAME:MirrorParameters", "MirrorBaseX:=" , <string>, "MirrorBaseY:=" , <string>, "MirrorBaseZ:=" , <string>, "MirrorNormalX:=" , <string>, "MirrorNormalY:=" , <string>, "MirrorNormalZ:=" , <string>)</pre>
Return Value	None.		

Python Syntax	Mirror(<SelectionsArray>, <MirrorParameters>)
Python Example	<pre> oEditor.Mirror(["NAME:Selections", "Selections:=", "Box1_1", "NewPartsModelFlag:=", "Model"], ["NAME:MirrorParameters", "MirrorBaseX:=", "-0.2mm", "MirrorBaseY:=", "-1.2mm", "MirrorBaseZ:=", "0mm", "MirrorNormalX:=", "-0.316227766016838mm", "MirrorNormalY:=", "0.948683298050514mm", "MirrorNormalZ:=", "0mm"]) </pre>

VB Syntax	Mirror <SelectionsArray> <MirrorParameters>
VB Example	<pre> oEditor.Mirror Array("NAME:Selections", "Selections:=", "Box1_1", "NewPartsModelFlag:=", "Model") Array("NAME:MirrorParameters", </pre>

	<pre>"MirrorBaseX:=", "-0.2mm", "MirrorBaseY:=", "-1.2mm", "MirrorBaseZ:=", "0mm", "MirrorNormalX:=", "-0.316227766016838mm", "MirrorNormalY:=", "0.948683298050514mm", "MirrorNormalZ:=", "0mm")</pre>

Move

Moves specified part(s).

UI Access	Edit > Arrange > Move.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<TranslateParameters>	Array	Structured array. Array(["NAME:TranslateParameters", "TranslateVectorX:=" , <string>, "TranslateVectorY:=" , <string>, "TranslateVectorZ:=" , <string>)
Return Value	None.		
Python Syntax	Move(<SelectionsArray>, <TranslateParameters>)		

Python Example	<pre> oEditor.Move(["NAME:Selections", "Selections:=", "Box1_1", "NewPartsModelFlag:=", "Model"], ["NAME:TranslateParameters", "TranslateVectorX:=", "-0.5mm", "TranslateVectorY:=", "0.1mm", "TranslateVectorZ:=", "0mm"]) </pre>

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).

UI Access	Edit > Arrange > Offset.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<OffsetParameters>	Array	Structured array. Array("NAME:OffsetParameters", "OffsetDistance:=" , <string>)
Return Value	None.		

Python Syntax	OffsetFaces (<SelectionsArray>, <OffsetParameters>)
Python Example	<pre>oEditor.OffsetFaces(["NAME:Selections", "Selections:=", "Box1_1", "NewPartsModelFlag:=", "Model"], ["NAME:OffsetParameters", "OffsetDistance:=", "16mm"])</pre>

VB Syntax	OffsetFaces <SelectionsArray> <OffsetParameters>
VB Example	<pre> oEditor.OffsetFaces Array("NAME:Selections", "Selections:=", "Box1_1", "NewPartsModelFlag:=", "Model") Array("NAME:OffsetParameters", "OffsetDistance:=", "16mm") </pre>

Paste (Model Editor)

Pastes previously copied object(s). See: [Copy](#).

UI Access	Edit > Paste.
Parameters	None.
Return Value	None.

Python Syntax	Paste()
Python Example	<pre> oEditor.Copy(["NAME:Selections", "Selections:=" , "Box1_2"]) oEditor.Paste() </pre>

VB Syntax	Paste
VB Example	<pre>oEditor.Copy Array("NAME:Selections", "Selections:=" , "Box1_2") oEditor.Paste</pre>

Rotate

Rotates specified object(s).

UI Access	Edit > Arrange > Rotate.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<RotateParameters>	Array	Structured array. <pre>Array("NAME:RotateParameters", "RotateAxis:=" , <string "X", "Y", or "Z">, "RotateAngle:=" , <string>)</pre>
Return Value	None.		

Python Syntax	Rotate(<SelectionsArray>, <RotateParameters>)
Python Example	<pre>oEditor.Rotate(["NAME:Selections", "Selections:=", "Box1_1", "NewPartsModelFlag:=", "Model"],</pre>

	<pre>["NAME:RotateParameters", "RotateAxis:=", "Z", "RotateAngle:=", "90deg"])</pre>

VB Syntax	Rotate <SelectionsArray> <RotateParameters>
VB Example	<pre>oEditor.Rotate Array("NAME:Selections", "Selections:=", "Box1_1", "NewPartsModelFlag:=", "Model") Array("NAME:RotateParameters", "RotateAxis:=", "Z", "RotateAngle:=", "90deg")</pre>

Scale

Scales specified object(s).

UI Access	Edit > Scale.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<ScaleParameters>	Array	Structured array. <pre>Array("NAME:ScaleParameters", "ScaleX:=" , <string containing scale factor>,</pre>

	"ScaleY:=" , <string containing scale factor>, "ScaleZ:=" , <string containing scale factor>)		
Return Value	None.		

Python Syntax	Scale (<SelectionsArray>, <ScaleParameters>)
Python Example	<pre>oEditor.Scale(["NAME:Selections", "Selections:=", "Box1", "NewPartsModelFlag:=", "Model"], ["NAME:ScaleParameters", "ScaleX:=", "2", "ScaleY:=", "2", "ScaleZ:=", "2"])</pre>

VB Syntax	Scale <SelectionsArray> <ScaleParameters>
VB Example	<pre>oEditor.Scale Array("NAME:Selections", "Selections:=", "Box1",</pre>

	<pre>"NewPartsModelFlag:=", "Model") Array("NAME:ScaleParameters", "ScaleX:=", "2", "ScaleY:=", "2", "ScaleZ:=", "2")</pre>

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](#)

AlignFaces

Aligns the adjacent selected faces of imported objects which have only one operation in their History Tree.

UI Access	Modeler > Model Preparation > Align Faces		
Parameters	Name	Type	Description
	<argFaceList>	Array	[["NAME:<SpecifiedName>",

			"BaseFaces:=", <FaceNumberArray>, "SnapFaces:=", <FaceNumberArray>]
	<FaceNumberArray>	Array	Array of number represents specified faces.
Return Value	None.		

Python Syntax	AlignFaces(<argFaceList>)		
Python Example	<pre>oEditor.AlignFaces(["NAME:Entity List", "BaseFaces:=" , [9], "SnapFaces:=" , [18]])</pre>		

VB Syntax	AlignFaces <argFaceList>		
VB Example	<pre>oEditor.AlignFaces Array("NAME:Entity List", "BaseFaces:=" , [9], "SnapFaces:=" , [18])</pre>		

AssignMaterial

Assigns a material to specified object(s).

UI Access	Modeler > Assign Material.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<AttributesArray>	Array	Structured array. See: AttributesArray . This script supports the following attributes: • MaterialValue • SolveInside • ShellElement • ShellElementThickness • IsMaterialEditable • UseMaterialAppearance • IsLightweight
Return Value	None.		
Python Syntax	AssignMaterial(<SelectionsArray>, <AttributesArray>)		

Python Example	<pre>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])</pre>
VB Syntax	AssignMaterial <SelectionsArray> <AttributesArray>

VB Example	<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: SelectionsArray .
	<Parameters>	Array	Structured array. <pre> Array("NAME:Parameters", Array("NAME:ChamferParameters", </pre>

			<pre>"Edges:=" , <array containing integer>, "Vertices:=" , <array>, "LeftDistance:=" , <string>, "RightDistance:=" , <string>, "ChamferType:=" , <string "Symmetric", "Left Distance-Angle", "Right Distance-Angle", or "Left Distance-Right Distance">)</pre>
Return Value	None.		

Python Syntax	Chamfer(<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.Chamfer(["NAME:Selections", "Selections:=" , "Box2", "NewPartsModelFlag:=" , "Model"], ["NAME:Parameters", ["NAME:ChamferParameters", "Edges:=" , [42], "Vertices:=" , [], "LeftDistance:=" , "0.1mm",</pre>

```

"RightDistance:="      , "0.1mm",
"ChamferType:="        , "Symmetric"
]
l)

```

VB Syntax	Chamfer <SelectionsArray> <Parameters>
VB Example	<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>

CleanUpModel

Cleans up history tree operations.

UI Access	Modeler > Cleanup Model History.

Parameters	None.
Return Value	None.

Python Syntax	CleanUpModel()
Python Example	<code>oEditor.CleanUpModel()</code>

VB Syntax	CleanUpModel
VB Example	<code>oEditor.CleanUpModel</code>

Connect

Connects two or more 1D polyline objects or 2D sheet objects.

UI Access	Modeler > Surface > Connect.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	Connect(<SelectionsArray>)
Python Example	<pre>oEditor.Connect(["NAME:Selections", "Selections:=", "Polyline2,Polyline1"])</pre>

VB Syntax	Connect <SelectionsArray>
VB Example	<pre>oEditor.Connect Array("NAME:Selections", "Selections:=", "Polyline2,Polyline1")</pre>

CoverLines

Covers two or more 1D objects to form a sheet.

UI Access	Modeler > Surface > Cover Lines.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	CoverLines(<SelectionsArray>)
Python Example	<pre>oEditor.CoverLines(["NAME:Selections", "Selections:=", "Polyline3,Polyline4", "NewPartsModelFlag:=", "Model"])</pre>

VB Syntax	CoverLines <SelectionsArray>
VB Example	<pre>oEditor.CoverLines Array("NAME:Selections",</pre>

	<pre>"Selections:=", "Polyline3,Polyline4", "NewPartsModelFlag:=", "Model")</pre>

CoverSurfaces

Covers two or more faces to form a solid object.

UI Access	Modeler > Surface > Cover Faces.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	CoverSurfaces (<SelectionsArray>)		
Python Example	<pre>oEditor.CoverSurfaces (["NAME:Selections", "Selections:=", "Obj1_Face1,Obj2_Face2", "NewPartsModelFlag:=", "Model"]) </pre>		

VB Syntax	CoverSurfaces <SelectionsArray>		
VB Example	<pre>oEditor.CoverSurfaces Array("NAME:Selections", </pre>		

	<pre>"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	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:GeometryEntityListParameters", "EntityType:=" , <string "Object" or "Face">, "EntityList:=" , <string of object names or IDs>) See GetObjectIDByName for returning object IDs.
	<AttributesArray>	Array	Structured array. See: AttributesArray . CreateEntityList takes only the "Name" parameter.
Return Value	None.		

Python Syntax	CreateEntityList(<Parameters>,<AttributesArray>)
Python Example	<pre>oEditor.CreateEntityList(["NAME:GeometryEntityListParameters", "EntityType:=", "Object", "EntityList:=", "Bondwire1,Bondwire2"])</pre>

	<pre>], ["NAME:Attributes", "Name:=", "Objectlist1"])</pre>

VB Syntax	CreateEntityList<Parameters> <AttributesArray>
VB Example	<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.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:FaceCSParameters", <OriginArray>, "MoveToEnd=", <boolean>,

			<pre>"FaceID:=", <integer>, <AxisPosnArray>, "WhichAxis:=", <string "X", "Y", or "Z">, "ZRotationAngle:=", <string>, "XOffset:=", <string>, "YOffset:=", <string>, "AutoAxis:=", <boolean></pre>
	<OriginArray>	Array	Structured array. <pre>Array("NAME:Origin", "IsAttachedToEntity:=", <boolean>, "EntityID:=", <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=", <integer>, "PositionType:=", <string "FaceCenter", "EdgeCenter", "OnVertex", "OnEdge", or "OnFace">, "UParam:=", <float between 0 and 1 representing the relative position of the point on the edge or face>, "VParam:=", <float between 0 and 1 representing the relative position of the point on the edge or face>, "XPosition:=", <string>, "YPosition:=", <string>,</pre>

			<code>"ZPosition:=" , <string>)</code> 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.
	<code><AxisPosnArray></code>	Array	Structured array. <code>Array("NAME:AxisPosn", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "FaceCenter", "EdgeCenter", "OnVertex", "OnEdge", or "OnFace">, "UParam:=" , <float>, "VParam:=" , <float>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>)</code>
	<code><AttributesArray></code>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	<code>CreateFaceCS(<Parameters>,<AttributesArray>)</code>

Python Example

```
oEditor.CreateFaceCS(  
  ["NAME:FaceCSParameters",  
    ["NAME:Origin",  
      "IsAttachedToEntity:=" , True,  
      "EntityID:=" , 46,  
      "FacetedBodyTriangleIndex:=", -1,  
      "TriangleVertexIndex:=" , -1,  
      "PositionType:=" , "FaceCenter",  
      "UParam:=" , 0,  
      "VParam:=" , 0,  
      "XPosition:=" , "0",  
      "YPosition:=" , "0",  
      "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"  
])
```

**VB Syn-
tax****CreateFaceCS <Parameters> <AttributesArray>**

VB Example	<pre> oEditor.CreateFaceCS Array("NAME:FaceCSParameters", Array("NAME:Origin", "IsAt- tachedToEntity:=", _ 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	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:GroupParameter", "ParentGroupID:=" , <string>, "Parts:=" , <string>,

	<table><tr><td></td><td></td><td>"SubmodelInstances:=" , <string>, "Groups:=" , <string>)</td></tr></table>					"SubmodelInstances:=" , <string>, "Groups:=" , <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>

VB Syntax	CreateGroup <Parameters>
VB Example	<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	Modeler > Coordinate System > Create > Object > [Offset / Rotated / Both].		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre>Array("NAME:ObjectCSParameters", <OriginArray> "MoveToEnd:=" , <boolean>, "ReverseXAxis:=" , <boolean>, "ReverseYAxis:=" , <boolean>, <xAxisArray / xAxisPosArray> <yAxisArray / yAxisPosArray>)</pre> Note: 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:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "OnVertex", "EdgeCenter", "FaceCenter", "OnEdge", or "AbsolutePosition">, "UParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or</pre>

		<pre>face>, "VParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>) 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	<pre>Structured array for absolute position: Array("NAME:xAxis", "DirectionType:=" , "AbsoluteDirection", "EdgeID:=" , <integer>, "FaceID:=" , <integer>, "xDirection:=" , <string>, "yDirection:=" , <string>, "zDirection:=" , <string>, "UParam:=" , <float>, "VParam:=" , <float>)</pre>
<xAxisPosArray>	Array	<pre>Structured array for relative position: Array("NAME:xAxisPos",</pre>

		<pre> "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "OnVertex", "EdgeCenter", "FaceCenter", or "OnEdge">, "UParam:=" , <float>, "VParam:=" , <float>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string> </pre>
<yAxisArray>	Array	Structured array for absolute position: <pre> Array("NAME:yAxis", "DirectionType:=" , "AbsoluteDirection", "EdgeID:=" , <integer>, "FaceID:=" , <integer>, "xDirection:=" , <string>, "yDirection:=" , <string>, "zDirection:=" , <string>, "UParam:=" , <float>, "VParam:=" , <float>) </pre>
<yAxisPosArray>	Array	Structured array for relative position:

			<pre> Array("NAME:yAxisPos", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "OnVertex", "EdgeCenter", "FaceCenter", or "OnEdge">, "UParam:=" , <float>, "VParam:=" , <float>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>) </pre>
	<AttributesArray>	Array	Structured array. See: AttributesArray .
Return Value	None.		

Python Syntax	CreateObjectCS(<Parameters>,<AttributesArray>)
Python Example	<pre> oEditor.CreateObjectCS(["NAME:ObjectCSParameters", ["NAME:Origin", "IsAttachedToEntity:=" , True, </pre>

```

"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",
  "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"
])
```

**VB Syn-
tax****CreateObjectCS <Parameters> <AttributesArray>**

**VB
Example**

```
oEditor.CreateObjectCS Array("NAME:ObjectCSParameters", Array("NAME:Origin", "IsAt-
tachedToEntity:=", _
false, "EntityID:=", -1, "FacetedBodyTriangleIndex:=", -1, "TriangleVertexIndex:=", _
-1, "PositionType:=", "AbsolutePosition", "UParam:=", 0, "VParam:=", 0, "XPosition:=",
_
"0mm", "YPosition:=", "0mm", "ZPosition:=", "0mm"), "MoveToEnd:=", false, "ReverseXAx-
is:=", _
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:=", _
"Box3")
```

CreateObjectFromEdges

Creates an object from the specified object edge.

UI Access	Modeler > Edge > Create Object From Edge		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .

	<ParametersArray>	Array	Structured array. Array("NAME:Parameters", Array("NAME:BodyFromEdgeToParameters", "Edges:=" , <array containing integer edges>))
	<CreateGroupsForNewObjects>	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	<pre>oEditor.CreateObjectFromEdges(["NAME:Selections", "Selections:=", "Box2", "NewPartsModelFlag:=", "Model"], ["NAME:Parameters", ["NAME:BodyFromEdgeToParameters",</pre>

	<pre> "Edges:=", [41]]], ["CreateGroupsForNewObjects:=", False]) </pre>

VB Syntax	CreateObjectFromEdges <SelectionsArray> <ParametersArray> <CreateGroupsForNewObjects>
VB Example	<pre> oEditor.CreateObjectFromEdges Array("NAME:Selections", "Selections:=", "Box1", "NewPartsModelFlag:=", "Model"), Array("NAME:Parameters", Array("NAME:BodyFromEdgeToParameters", "Edges:=", Array(_ 13))), Array("CreateGroupsForNewObjects:=", false) </pre>

CreateObjectFromFace

Creates 2D objects from specified face(s).

UI Access	N/A		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array("NAME:Parameters", <BodyFromFaceToParameters>)
	<CreateGroupsForNewObjects>	Array	Optional. Structured array. Array("CreateGroupsForNewObjects:=", <boolean>)

Return Value	None.

Python Syntax	CreateObjectFromFace (<SelectionsArray>,<Parameters>,<CreateGroupsForNewObjects>)
Python Example	<pre>oEditor.CreateObjectFromFace(["NAME:Selections", "Selections:=" , "Box3", "NewPartsModelFlag:=" , "Model"], ["NAME:Parameters", ["NAME:BodyFromFaceToParameters", "FacesToDetach:=" , [68]],], ["CreateGroupsForNewObjects:=", False])</pre>

VB Syntax	CreateObjectFromFace <SelectionsArray> <Parameters> <CreateGroupsForNewObjects>
VB Example	<pre>oEditor.CreateObjectFromFace Array("NAME:Selections", "Selections:=", "Box3", "NewPartsModelFlag:=" , "Model") Array("NAME:Parameters", Array("NAME:BodyFromFaceToParameters",</pre>

```
"FacesToDetach:=", Array(68))
Array("CreateGroupsForNewObjects:=", false)
```

CreateObjectFromFaces

Creates 2D objects from specified face(s).

UI Access	Modeler > Surface > Create Object from Face		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array("NAME:Parameters", <BodyFromFaceToParameters>)
	<CreateGroupsForNewObjects>	Array	Structured array. Array("CreateGroupsForNewObjects:=", <boolean>)
Return Value	None.		

Python Syntax	CreateObjectFromFaces (<SelectionsArray>, <Parameters>, <CreateGroupsForNewObjects>)
Python Example	<pre>oEditor.CreateObjectFromFaces (["NAME:Selections", "Selections:=" , "Box3", "NewPartsModelFlag:=" , "Model"], ["NAME:Parameters",</pre>

	<pre>["NAME:BodyFromFaceToParameters", "FacesToDetach:=" , [68]]], ["CreateGroupsForNewObjects:=", False])</pre>

VB Syntax	CreateObjectFromFaces <SelectionsArray> <Parameters> <CreateGroupsForNewObjects>
VB Example	<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	Modeler > Coordinate System > Create > Relative CS > [Offset / Rotated / Both].		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:RelativeCSParameters", "Mode:=" , "Axis/Position",

			<pre> "OriginX:=" , <string>, "OriginY:=" , <string>, "OriginZ:=" , <string>, "XAxisXvec:=" , <string>, "XAxisYvec:=" , <string>, "XAxisZvec:=" , <string>, "YAxisXvec:=" , <string>, "YAxisYvec:=" , <string>, "YAxisZvec:=" , <string> </pre>
	<AttributesArray>	Array	Structured array. See: AttributesArray . CreateRelativeCS supports only the "Name" parameter.
Return Value	None.		

Python Syntax	CreateRelativeCS(<Parameters>,<AttributesArray>)		
Python Example	<pre> oEditor.CreateRelativeCS(["NAME:RelativeCSParameters", "Mode:=" , "Axis/Position", "OriginX:=" , "0.62mm", "OriginY:=" , "-0.7mm", "OriginZ:=" , "0mm", "XAxisXvec:=" , "1mm", "XAxisYvec:=" , "0mm", </pre>		

	<pre>"XAxisZvec:=" , "0mm", "YAxisXvec:=" , "0mm", "YAxisYvec:=" , "1mm", "YAxisZvec:=" , "0mm"], ["NAME:Attributes", "Name:=" , "RelativeCS1"])</pre>

VB Syn- tax	CreateRelativeCS <Parameters> <AttributesArray>
VB Example	<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.

UI Access	Modeler > Group > Delete Empty.

Parameters	Name		
	Type		
	Description		
	<i><Parameters></i>	Array	Structured array.
			Array("Groups:=", <array of string group IDs>)
Return Value	None.		

Python Syntax	DeleteEmptyGroups(<i><Parameters></i>)		
Python Example	<pre>oEditor.DeleteEmptyGroups(["Groups:=", ["Group1", "Group2", "Group3"]])</pre>		

VB Syntax	DeleteEmptyGroups <i><Parameters></i>		
VB Example	oEditor.DeleteEmptyGroups Array("Groups:=", Array("Group1", "Group2", "Group3"))		

DeleteLastOperation

Deletes the last operation performed on the specified object(s).

UI Access	Modeler > Delete Last Operation.		
Parameters	Name		
	Type		
	Description		
	<i><SelectionsArray></i>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	DeleteLastOperation(<i><SelectionsArray></i>)		
Python Example	oEditor.DeleteLastOperation([		

	<pre>"NAME:Selections", "Selections:=", "Box3", "NewPartsModelFlag:=", "Model"]</pre>

VB Syntax	DeleteLastOperation <SelectionsArray>
VB Example	oEditor.DeleteLastOperation Array("NAME:Selections", "Selections:=", "Box3", "NewPartsModelFlag:=", "Model")

DeleteOperation

Deletes specified operation performed on a selected object.

UI Access	Select an operation in the project tree, then press Delete on the keyboard.		
Parameters	Name	Type	Description
	<ParamsArray>	Array	Structured array. <pre>Array("NAME:Parameters", Array("NAME:PartOperations", Array("NAME:<PartName>", "OperationIndices:=", <array of operation indices>)), Array("NAME:UDMOperations"))</pre>

Return Value	None.

Python Syntax	DeleteOperation(<ParamsArray>)
Python Example	<pre>oEditor.DeleteOperation(["NAME:Parameters", ["NAME:PartOperations", ["NAME:Coil_0", "OperationIndices:=", [1]]], ["NAME:UDMOperations"]])</pre>

VB Syntax	DeleteOperation <ParamsArray>
VB Example	<pre>oEditor.DeleteOperation _ Array("NAME:Parameters", _ Array("NAME:PartOperations", _</pre>

	<pre>Array("NAME:Coil_0", _ "OperationIndices:=", Array(1))), _ Array("NAME:UDMOperations"))</pre>

DetachEdges

Detaches the specified edge(s) from an object.

UI Access	Modeler > Edge > Detach Edges.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array("NAME:Parameters", <DetachEdgesArray>)
	<DetachEdgesArray>	Array	Structured array. Array("NAME:DetachEdgesToParameters", "EdgesToDetach:=" , <array containing integer edge IDs>)
Return Value	None.		

Python Syntax	DetachEdges(<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.DetachEdges(["NAME:Selections",</pre>

	<pre> "Selections:=", "Rectangle1", "NewPartsModelFlag:=", "Model"], ["NAME:Parameters", ["NAME:DetachEdgesToParameters", "EdgesToDetach:=", [18,17]]]] </pre>

VB Syntax	DetachEdges <SelectionsArray> <Parameters>
VB Example	<pre> oEditor.DetachEdges Array("NAME:Selections", "Selections:=", "Rectangle1", "NewPartsModelFlag:=", "Model") Array("NAME:Parameters", Array("NAME:DetachEdgesToParameters", "EdgesToDetach:=", [18,17])) </pre>

DetachFaces

Detaches the specified face(s) from an object.

UI Access	Modeler > Surface > Detach Faces.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array("NAME:Parameters",

			<DetachFacesArray>)
	<DetachFacesArray>	Array	Structured array. Array("NAME:DetachFacesToParameters", "FacesToDetach:=" , <array containing integer face IDs>)
Return Value	None.		

Python Syntax	DetachFaces(<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.DetachFaces(["NAME:Selections", "Selections:=", "Box3", "NewPartsModelFlag:=", "Model"], ["NAME:Parameters", ["NAME:DetachFacesToParameters", "FacesToDetach:=", [68,67]]])</pre>

VB Syntax	DetachFaces <SelectionsArray> <Parameters>

VB Example	<pre>oEditor.DetachFaces Array("NAME:Selections", "Selections:=", "Box3", "NewPartsModelFlag:=", "Model") Array("NAME:Parameters", Array("NAME:DetachFacesToParameters", "FacesToDetach:=", [68,67]))</pre>

EditFaceCS

Recreates an existing face coordinate system. See: [CreateFaceCS](#).

UI Access	Modeler > Coordinate System > Edit.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre>Array("NAME:FaceCSParameters", <OriginArray>, "MoveToEnd:=", <boolean>, "FaceID:=", <integer>, <AxisPosnArray>, "WhichAxis:=", <string "X", "Y", or "Z">, "ZRotationAngle:=", <string>, "XOffset:=", <string>, "YOffset:=", <string>, "AutoAxis:=", <boolean>)</pre>
	<OriginArray>	Array	Structured array. <pre>Array("NAME:Origin", "IsAttachedToEntity:=", <boolean>,</pre>

		<pre>"EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "FaceCenter", "EdgeCenter", "OnVertex", "OnEdge", or "OnFace">, "UParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "VParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>)</pre> 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.
	<AxisPosnArray>	Array Structured array. <pre>Array("NAME:AxisPosn", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=", <integer>,</pre>

			"TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "FaceCenter", "EdgeCenter", "OnVertex", "OnEdge", or "OnFace">, "UParam:=" , <float>, "VParam:=" , <float>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>)
	<AttributesArray>	Array	Structured array. See: AttributesArray . Use to select the coordinate system to edit.
Return Value	None.		

Python Syntax	EditFaceCS (<Parameters>, <AttributesArray>)
Python Example	<pre>oEditor.EditFaceCS (["NAME:FaceCSParameters", ["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,
["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",
```

	<pre> "ZRotationAngle:=" , "0deg", "XOffset:=" , "0mm", "YOffset:=" , "0mm", "AutoAxis:=" , False], ["NAME:Attributes", "Name:=" , "FaceCS1", "PartName:=" , "Box1"]) </pre>

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, "VParam:=" , 0, </pre>

```
"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",
```

	<pre> "AutoAxis:=" , False) Array("NAME:Attributes", "Name:=" , "FaceCS1", "PartName:=" , "Box1") </pre>

EditObjectCS

Edits an existing object coordinate system. See: [CreateObjectCS](#).

UI Access	Modeler > Coordinate System > Edit.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre> Array("NAME:ObjectCSParameters", <OriginArray> "MoveToEnd:=" , <boolean>, "ReverseXAxis:=" , <boolean>, "ReverseYAxis:=" , <boolean>, <xAxisArray / xAxisPosArray> <yAxisArray / yAxisPosArray>) </pre> Note: 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:=" , <boolean>, </pre>

		<pre>"EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=" , <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "OnVertex", "EdgeCenter", "FaceCenter", "OnEdge", or "AbsolutePosition">, "UParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "VParam:=" , <float between 0 and 1 representing the relative position of the point on the edge or face>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>)</pre> 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.
	<xAxisArray>	Array Structured array for absolute position: <pre>Array("NAME:xAxis", "DirectionType:=" , "AbsoluteDirection", "EdgeID:=" , <integer>, "FaceID:=" , <integer>,</pre>

		<pre> "xDirection:=" , <string>, "yDirection:=" , <string>, "zDirection:=" , <string>, "UParam:=" , <float>, "VParam:=" , <float>) </pre>
	<xAxisPosArray>	Array Structured array for relative position: <pre> Array("NAME:xAxisPos", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=", <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "OnVertex", "EdgeCenter", "FaceCenter", or "OnEdge">, "UParam:=" , <float>, "VParam:=" , <float>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>) </pre>
	<yAxisArray>	Array Structured array for absolute position: <pre> Array("NAME:yAxis", "DirectionType:=" , "AbsoluteDirection", "EdgeID:=" , <integer>, </pre>

		<pre>"FaceID:=" , <integer>, "xDirection:=" , <string>, "yDirection:=" , <string>, "zDirection:=" , <string>, "UParam:=" , <float>, "VParam:=" , <float>)</pre>
<code><yAxisPosArray></code>	Array	Structured array for relative position: <pre>Array("NAME:yAxisPos", "IsAttachedToEntity:=" , <boolean>, "EntityID:=" , <integer>, "FacetedBodyTriangleIndex:=" , <integer>, "TriangleVertexIndex:=" , <integer>, "PositionType:=" , <string "OnVertex", "EdgeCenter", "FaceCenter", or "OnEdge">, "UParam:=" , <float>, "VParam:=" , <float>, "XPosition:=" , <string>, "YPosition:=" , <string>, "ZPosition:=" , <string>)</pre>
<code><AttributesArray></code>	Array	Structured array. See: AttributesArray . Use to select the coordinate system to edit.

Return Value	None.

Python Syntax	EditObjectCS(<Parameters>,<AttributesArray>)
Python Example	<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, ["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",
    "Name:="                , "ObjectCS1",
    "PartName:="            , "Box2"
]
```

VB Syntax	EditObjectCS <Parameters> <AttributesArray>
VB Example	<pre>oEditor.EditObjectCS Array("NAME:ObjectCSParameters", Array("NAME:Origin", "IsAt- tachedToEntity:=", _ false, "EntityID:=", -1, "FacetedBodyTriangleIndex:=", -1, "TriangleVertexIndex:=", _ -1, "PositionType:=", "AbsolutePosition", "UParam:=", 0, "VParam:=", 0, "XPosition:=", _ "0mm", "YPosition:=", "0mm", "ZPosition:=", "0mm"), "MoveToEnd:=", false, "ReverseXAx- is:=", _ 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")

EditRelativeCS

Edits an existing Relative Coordinate System. See: [CreateRelativeCS](#).

UI Access	Modeler > Coordinate System > Edit.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:RelativeCSParameters", "Mode:=" , "Axis/Position", "OriginX:=" , <string>, "OriginY:=" , <string>, "OriginZ:=" , <string>, "XAxisXvec:=" , <string>, "XAxisYvec:=" , <string>, "XAxisZvec:=" , <string>, "YAxisXvec:=" , <string>, "YAxisYvec:=" , <string>, "YAxisZvec:=" , <string>)
	<AttributesArray>	Array	Structured array. See: AttributesArray . Use to select the coordinate system to edit.
Return Value	None.		

Python Syntax	EditRelativeCS(<Parameters>,<AttributesArray>)
Python Example	<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>

VB Syntax	EditRelativeCS <Parameters> <AttributesArray>
VB	<pre> oEditor.EditRelativeCS Array("NAME:RelativeCSParameters", "Mode:=", "Axis/Position", </pre>

Example	<pre>"OriginX:=", _ "0mm", "OriginY:=", "0mm", "OriginZ:=", "0mm", "XAxisXvec:=", "0.66mm", "XAxisYvec:=", _ "0.28mm", "XAxisZvec:=", "0mm", "YAxisXvec:=", "0.06mm", "YAxisYvec:=", "0.14mm", "YAxisZvec:=", _ "0mm"), Array("NAME:Attributes", "Name:=", "RelativeCS1")</pre>

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	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:ExportParameters", "AllowRegionDependentPartSelectionForPMLCreation:=", <boolean>, "AllowRegionSelectionForPMLCreation:=", <boolean>, "Selections:=" , <string list>, "File Name:=" , <string filepath>, "Major Version:=" , <integer (-1 if not applic-

			able)>, "Minor Version:=" , <integer (-1 if not applicable)>)
Return Value	None.		

Python Syntax	Export(<Parameters>)
Python Example	<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>

VB Syntax	Export <Parameters>
VB Example	<pre>oEditor.Export Array("NAME:ExportParameters", "AllowRegionDependentPartSelectionForPMLCreation:=", True, "AllowRegionSelectionForPMLCreation:=", True,</pre>

	"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 Ensign use *.avz. For export to Ensign in -ng mode, the corresponding version of Ensign 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 Ensign 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 boolean>, "ShowGrid:=" , <string containing boolean>,

		<pre> "ShowRuler:=" , <string containing boolean>, "ShowRegion:=" , <string>, "Selections:=" , <string>, "FieldPlotSelections:=", <string>' # Comma delimited string. #Use to set which field plot to show. "FitToSelections:=" , "", </pre> Note: "FitToSelections" specify geometry objects for the "Fit" operation. <pre> "FitToFieldPlotSelection:=" , "" </pre> Note: "FitToFieldPlotSelections" specifies field plots for the "Fit" operation. <pre> "AutoFit:=" , "True", </pre> Note: If FitToSlections or FitToFieldPlotSelections are used , then then AutoFit is true, it makes sure color key does not overlap field plot. It is true by default. If neither is used, then "AutoFit" will "Fit" to full model. <pre> "Orientation:=" , <string> "ShowOrientationGadget:=" , <False> </pre>

Return Value	None.

Python Syntax	ExportModelImageToFile(<path> <width> <height> <Parameters>)
Python Example	<pre>oEditor.ExportModelImageToFile ("C:/Users/jdoe/Desktop/export.bmp", 1920, 1080, ["NAME:SaveImageParams", "ShowAxis:=" , "True", "ShowGrid:=" , "True", "ShowRuler:=" , "True", "ShowRegion:=" , "Default", "Selections:=" , "", "FieldPlotSelections:=" , "", "FitToSelections:=" , "", "FitToFieldPlotSelections:=" , "", "AutoFit:=" , "True", "Orientation:=" , ""])</pre>

VB Syntax	ExportModelImageToFile <path> <width> <height> <Parameters>
VB Example	<pre> oEditor.ExportModelImageToFile "C:/Users/jdoe/Desktop/export.bmp", 1383, 512, Array("NAME:SaveImageParams", "ShowAxis:=", "True", "ShowGrid:=", "True", "ShowRuler:=", "True", "ShowRegion:=", "Default", "Selections:=", "", "FitToFieldPlotSelections:=", "", "AutoFit:=", "True", "Orientation:=", "" </pre>

ExportModelMeshToFile

Exports geometry model to a 3D model file (e.g. *.obj, *.wrl, etc.).

UI Access	N/A		
Parameters	Name	Type	Description
	<filePath>	String	Full file path, including extension *.obj, *.wrl, etc
	<selections>	Array	Selected parts.

Return Value	None.

Python Syntax	<code>ExportModelMeshToFile <filePath>, <selections></code>
Python Example	<pre>oEditor.ExportModelMeshToFile("E:/MyDir/scriptRun/2Selected-ng.obj", ["BotCover-", "AveragingVolumeAtPeakRMSEfieldLocation"])</pre>

VB Syntax	<code>ExportModelMeshToFile <filePath>, <selections></code>
VB Example	<pre>oEditor.ExportModelMeshToFile("E:/MyDir/scriptRun/2Selected-ng.obj", ["BotCover-", "AveragingVolumeAtPeakRMSEfieldLocation"])</pre>

Fillet

Performs a fillet on specified edge(s).

UI Access	Modeler > Fillet.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array("NAME:Parameters", <FilletParametersArray>)
	<FilletParametersArray>	Array	Structured array. Array("NAME:FilletParameters", "Edges:=" , <array containing integer edge IDs>,

			"Vertices:=" , <empty array>, "Radius:=" , <string>, "Setback:=" , <string>)
Return Value	None.		

Python Syntax	Fillet(<SelectionsArray>, <Parameters>)		
Python Example	<pre> oEditor.Fillet(["NAME:Selections", "Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model"], ["NAME:Parameters", ["NAME:FilletParameters", "Edges:=" , [13], "Vertices:=" , [], "Radius:=" , "1mm", "Setback:=" , "0mm"]]) </pre>		

VB Syntax	Fillet <SelectionsArray> <Parameters>		

VB Example	<pre>oEditor.Fillet Array("NAME:Selections", "Selections:=", "Box1", "NewPartsModelFlag:=", "Model") Array("NAME:Parameters", Array("NAME:FilletParameters", "Edges:=", Array(13), "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	<code>oEditor.FlattenGroup(["Groups:=", ["Group1"]])</code>

VB Syntax	FlattenGroup <GroupID>
VB Example	<code>oEditor.FlattenGroup Array("Groups:=", Array("Group1"))</code>

GenerateHistory

Generates the history for specified 1D object(s).

UI Access	Modeler > Generate History.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	GenerateHistory (<SelectionsArray>)
Python Example	<pre>oEditor.GenerateHistory(["NAME:Selections", "Selections:=" , "Polyline1", "NewPartsModelFlag:=" , "Model", "UseCurrentCS:=" , True]) </pre>

VB Syntax	GenerateHistory <SelectionsArray>
VB Example	<pre>oEditor.GenerateHistory Array("NAME:Selections", "Selections:=" , "Polyline1", "NewPartsModelFlag:=" , "Model", "UseCurrentCS:=" , True)</pre>

GetActiveCoordinateSystem

Returns the active coordinate system.

UI Access	None.
Parameters	None.
Return Value	String name of active coordinate system.

Python Syntax	GetActiveCoordinateSystem()
Python Example	<pre>oEditor.GetActiveCoordinateSystem()</pre>

VB Syntax	GetActiveCoordinateSystem
VB Example	<pre>dim csName csName = oEditor.GetActiveCoordinateSystem</pre>

GetActiveCoordinateSystemTransform

Returns transformation information for the active coordinate system (with respect to the global coordinate system), which consists of the affine matrix for rotation of the coordinate system and a 3D vector with the translation of the coordinate system's origin with respect to the global coordinate system's origin.

UI Access	None
Parameters	None
Return Value	Array of strings containing transformation information

Python Syntax	<code>GetActiveCoordinateSystemTransform()</code>
Python Example	<code>oEditor.GetActiveCoordinateSystemTransform()</code>

VB Syntax	<code>GetActiveCoordinateSystemTransform</code>
VB Example	<code>oEditor.GetActiveCoordinateSystemTransform</code>

GetCoordinateSystems

Returns the names of coordinate systems in the design.

UI Access	None.
Parameters	None.
Return Value	Array containing string names of coordinate systems.

Python Syntax	<code>GetCoordinateSystems()</code>

Python Example	<code>oEditor.GetCoordinateSystems()</code>

VB Syntax	<code>GetCoordinateSystems()</code>
VB Example	<pre>dim csNames csNames = oEditor.GetCoordinateSystems</pre>

HealObject

Heals an imported object.

UI Access	Modeler > Model Preparation > Heal.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array("NAME:ObjectHealingParameters", "Version:=" , <integer>, "AutoHeal:=" , <boolean>, "TolerantStitch:=" , <boolean>, "SimplifyGeom:=" , <boolean>, "TightenGaps:=" , <boolean>, "HealToSolid:=" , <boolean>, "StopAfterFirstStitchError:=" , <boolean>, "MaxStitchTol:=" , <float>,

		<pre> "ExplodeAndStitch:=" , <boolean>, "GeomSimplificationTol:=" , <integer>, "MaximumGeneratedRadiusForSimplification:=" , <integer>, "SimplifyType:=" , <integer>, "TightenGapsWidth:=" , <float>, "RemoveSliverFaces:=" , <boolean>, "RemoveSmallEdges:=" , <boolean>, "RemoveSmallFaces:=" , <boolean>, "SliverFaceTol:=" , <integer>, "SmallEdgeTol:=" , <integer>, "SmallFaceAreaTol:=" , <integer>, "BoundingBoxScaleFactor:=" , <integer>, "RemoveHoles:=" , <boolean>, "RemoveChamfers:=" , <boolean>, "RemoveBlends:=" , <boolean>, "HoleRadiusTol:=" , <integer>, "ChamferWidthTol:=" , <integer>, "BlendRadiusTol:=" , <integer>, "AllowableSurfaceAreaChange:=" , <integer>, "AllowableVolumeChange:=" , <integer>) </pre>

Return Value	None.

Python Syntax	HealObject(<SelectionsArray>, <Parameters>)
Python Example	<pre>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,</pre>

	<pre> "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] </pre>

VB Syntax	HealObject <SelectionsArray> <Parameters>
VB	oEditor.HealObject Array("NAME:Selections", "Selections:=", "Box2",

Example	<pre> "NewPartsModelFlag:=", _ "Model"), Array("NAME:ObjectHealingParameters", "Version:=", 1, "AutoHeal:=", false, "TolerantStitch:=", _ true, "SimplifyGeom:=", true, "TightenGaps:=", true, "HealToSolid:=", false, "StopAfter- FirstStitchError:=", _ 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) </pre>

Import

Imports a 3D model file.

Note:

This script does not import DXF or GDSII models. See: [ImportDXF](#) and [ImportGDSII](#).

UI Access	Modeler > Import.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre> Array("NAME:NativeBodyParameters", "HealOption:=" , <integer>, "Options:=" , <string>, "FileType:=" , <string "UnRecognized">, "MaxStitchTol:=" , <integer>, "ImportFreeSurfaces:=" , <boolean>, "GroupByAssembly:=" , <boolean>, "CreateGroup:=" , <boolean>, "STLFileUnit:=" , <string>, "MergeFacesAngle:=" , <float>, "HealSTL:=" , <boolean>, "ReduceSTL:=" , <boolean>, "ReduceMaxError:=" , <integer>, "ReducePercentage:=" , <integer>, "PointCoincidenceTol:=" , <float>, "CreateLightweightPart:=" , <boolean>, "ImportMaterialNames:=" , <boolean>, "SeparateDisjointLumps:=" , <boolean> </pre>

	<table><tr><td></td><td></td><td>"SourceFile:=" , <string>)</td></tr></table>					"SourceFile:=" , <string>)
		"SourceFile:=" , <string>)				
Return Value	None.					

Python Syntax	Import(<Parameters>)
Python Example	<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>

	<pre> "CreateLightweightPart:=", False, "ImportMaterialNames:=", False, "SeparateDisjointLumps:=", False, "SourceFile:=", "C:\\Users\\jdoe\\Desktop\\export.model"] </pre>

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>

	<pre>"PointCoincidenceTol:=" , 1E-06, "CreateLightweightPart:=", False, "ImportMaterialNames:=" , False, "SeparateDisjointLumps:=", False, "SourceFile:=" , "C:\\Users\\jdoe\\Desktop\\export.model")</pre>

ImportDXF [Modeler]

Imports a DXF model file.

UI Access	Modeler > Import.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:options", "FileName:=" , <string>, "Scale:=" , <float>, "AutoDetectClosed:=" , <boolean>, "SelfStitch:=" , <boolean>, "DefeatureGeometry:=" , <boolean>, "DefeatureDistance:=" , <integer>, "RoundCoordinates:=" , <boolean>, "RoundNumDigits:=" , <integer>, "WritePolyWithWidthAsFilledPoly:=" , <boolean>,

			<pre>"ImportMethod:=" , <integer>, "2DSheetBodies:=" , <boolean>, <layersArray>)</pre>
	<code><layersArray></code>	Array	Structured array. <pre>Array("NAME:LayerInfo", <layer>, <layer>, <layer>,...)</pre>
	<code><layer></code>	Array	Structured array. <pre>Array("NAME:<layerName>", "source:=" , <string>, "display_source:=" , <string>, "import:=" , <boolean>, "dest:=" , <string>, "dest_selected:=" , <boolean>, "layer_type:=" , <string>)</pre>
Return Value	None.		

Python Syntax	<code>ImportDXF(<Parameters>)</code>
Python Example	<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"
],
["NAME:LAYER_2",
"source:="                , "LAYER_2",
"display_source:="        , "LAYER_2",
"import:="                , True,
"dest:="                  , "LAYER_2",
"dest_selected:="        , False,
"layer_type:="            , "signal"
]
]
]
])

```

VB Syntax	ImportDXF < <i>Parameters</i> >
VB Example	<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")))
```

ImportFromClipboard

Imports a model from clipboard.

UI Access	Modeler > Import From Clipboard.
Parameters	None.
Return Value	None.

Python Syntax	ImportFromClipboard ()
Python Example	oEditor.ImportFromClipboard()

VB Syntax	ImportFromClipboard

VB Example	<code>oEditor.ImportFromClipboard</code>

ImportGDSII [Modeler]

Imports a GDSII model file.

UI Access	Modeler > Import.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre>Array("NAME:options", "FileName:=" , <string>, "FlattenHierarchy:=" , <boolean>, "ImportMethod:=" , <integer>, <layerMapArray>, <layerMapArray>, <layerMapArray>,... "OrderMap:=" , ["entry:=" , <entry>, <entry>, <entry>,...])</pre>
	<layerMapArray>	Array	Structured array. <pre>Array("NAME:LayerMapInfo", "LayerNum:=" , <integer>, "DestLayer:=" , <string>, "layer_type:=" , <string>)</pre>

	<entry>	Array	Structured array. Array("order:=" , <integer LayerNum>, "layer:=" , <string DestLayer>)
Return Value	None.		

Python Syntax	ImportGDSII(<Parameters>)
Python Example	<pre>oEditor.ImportGDSII(["NAME:options", "FileName:=", "C:/Users/coil2.gds", "FlattenHierarchy:=" , True, "ImportMethod:=" , 1, ["NAME:LayerMap", ["NAME:LayerMapInfo", "LayerNum:=" , 12, "DestLayer:=" , "Signal12", "layer_type:=" , "signal"], ["NAME:LayerMapInfo", "LayerNum:=" , 13, "DestLayer:=" , "Signal13",</pre>

```
        "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"]
]
])
```

VB Syntax	ImportGDSII < <i>Parameters</i> >
VB Example	oEditor.ImportGDSII Array("NAME:options",

	<pre>"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>

Imprint

Imprints the geometry of one object upon another.

UI Access	Modeler > Boolean > Imprint...		
Parameters	Name	Type	Description
	<Selections>	Array	Structured array. Array("NAME:Selections", "Blank Parts:=", <string, object name>, "Tool Parts:=", <string, object name>)

	<table><tr><td><Parameters></td><td>Array</td><td>Structured array. Array("NAME:ImprintParameters", "KeepOriginals:=", <boolean>)</td></tr></table>	<Parameters>	Array	Structured array. Array("NAME:ImprintParameters", "KeepOriginals:=", <boolean>)
<Parameters>	Array	Structured array. Array("NAME:ImprintParameters", "KeepOriginals:=", <boolean>)		
Return Value	None.			

Python Syntax	Imprint (<Selections>, <Parameters>)
Python Example	<pre>oEditor.Imprint(["NAME:Selections", "Blank Parts:=", "Cylinder1", "Tool Parts:=", "Box1"], ["NAME:ImprintParameters", "KeepOriginals:=", False])</pre>

VB Syntax	Imprint <Selections>, <Parameters>
VB Example	<pre>oEditor.Imprint Array("NAME:Selections", "Blank Parts:=", "Cylinder1", "Tool Parts:=", "Box1"), Array("NAME:ImprintParameters", "KeepOriginals:=", false)</pre>

ImprintProjection

Projects the form of a sheet object onto the face or faces of another object (either solid or sheet).

UI Access	Modeler > Boolean > Imprint Projection.		
Parameters	Name	Type	Description
	<Selections>	Array	Structured array. Array("NAME:Selections", "Selections:=", <string, selected objects>)
	<Parameters>	Array	Structured array. Array("NAME:ImprintProjectionParameters", "KeepOriginals:=", <boolean>, "NormalProjection:=", <boolean>, "Distance:=" , <float>, "DirectionX:=" , <float>, "DirectionY:=" , <float>, "DirectionZ:=" , <float>)
Return Value	None.		

Python Syntax	ImprintProjection (<Selections>, <Parameters>)
Python Example	oEditor.ImprintProjection(

	<pre>["NAME:Selections", "Selections:=", "Rectangle1,Cylinder1"], ["NAME:ImprintProjectionParameters", "KeepOriginals:=", False, "NormalProjection:=", True, "Distance:=", "1.36014705087354mm", "DirectionX:=", "0.882257546512569", "DirectionY:=", "0.294085848837523", "DirectionZ:=", "0.367607311046904"]])</pre>

VB Syntax	ImprintProjection <Selections>, <Parameters>
VB Example	<pre>oEditor.ImprintProjection Array("NAME:Selections", "Selections:=", "Rectangle1,Cylinder1"), Array("NAME:ImprintProjectionParameters", "KeepOriginals:=", false, "NormalProjection:=", true, "Distance:=", "1.36014705087354mm", "DirectionX:=", "0.882257546512569", "DirectionY:=", "0.294085848837523", "DirectionZ:=", "0.367607311046904")</pre>

Intersect

Intersects specified objects.

UI Access	Modeler > Boolean > Intersect.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array("NAME:IntersectParameters", "KeepOriginals:=" , <boolean True to keep original objects; False to delete>)
Return Value	None.		

Python Syntax	Intersect(<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.Intersect(["NAME:Selections", "Selections:=", "Rectangle1,Rectangle2"], ["NAME:IntersectParameters", "KeepOriginals:=", False])</pre>

VB Syntax	<code>Intersect <SelectionsArray> <Parameters></code>
VB Example	<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.

UI Access	Modeler > Coordinate System > Move CS to End.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	<code>MoveCStoEnd (<SelectionsArray>)</code>
Python Example	<pre>oEditor.MoveCStoEnd(["NAME:Selections", "Selections:=" , "ObjectCS1"])</pre>

VB Syntax	<code>MoveCStoEnd <SelectionsArray></code>
VB Example	<code>oEditor.MoveCStoEnd Array("NAME:Selections", "Selections:=", "ObjectCS1")</code>

MoveEntityToGroup

Moves a specified entity or entities to a specified group.

UI Access	Drag item into the group in the history tree.		
Parameters	Name	Type	Description
	<code><Objects></code>	Array	Structured array. Selects the entity/entities to move. <code>Array("Objects:=" , <array containing string object IDs>)</code>
	<code><MoveEntityToGroup></code>	Array	Structured array. <code>Array("ParentGroup:=" , <string group name>)</code>
Return Value	None.		

Python Syntax	<code>MoveEntityToGroup (<Objects>, <MoveEntityToGroup>)</code>
Python Example	<pre>oEditor.MoveEntityToGroup(["Objects:=", ["Box3"]], ["ParentGroup:=", "Box_Group"])</pre>

VB Syntax	<code>MoveEntityToGroup <Objects> <MoveEntityToGroup></code>

VB Example	<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	Modeler > Surface > Move Faces > Along [Normal/Vector].		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<MoveFacesParametersArray>	Array	Structured array. <pre>Array("NAME:Parameters", <FacesOfOneObjToMove>, <FacesOfOneObjToMove>, ...)</pre>
	<FacesOfOneObjToMove>	Array	Structured array. <pre>Array("Name:MoveFacesParameters", "MoveAlongNormalFlag:=", <boolean>, "OffsetDistance:=", <string>, "MoveVectorX:=", <string>, "MoveVectorY:=", <string>, "MoveVectorZ:=", <string>, "FacesToMove:=", <array>)</pre> MoveAlongNormalFlag specifies whether to move along the face normal or along a vector. If false, provide the MoveVectorX, MoveVectorY, and MoveVectorZ parameters.

			FacesToMove is an array of integers (the IDs of the faces to move).
Return Value	None		

Python Syntax	MoveFaces (<SelectionsArray>, <MoveFacesParametersArray>)
Python Example	<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>

VB Syntax	MoveFaces <SelectionsArray>, <MoveFacesParametersArray>
VB Example	<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**.

UI Access	Click Modeler > Surface > Project Sheet .		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Array containing string. Array("NAME:ProjectSheetParameters")

Return Value	None.

Python Syntax	ProjectSheet (<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.ProjectSheet(['NAME:Selections', 'Selections:=','Box1,Box2,Polyline1'], ['NAME:ProjectSheetParameters'])</pre>

VB Syntax	ProjectSheet <SelectionsArray> <Parameters>
VB Example	<pre>oEditor.ProjectSheet Array("NAME:Selections", "Selections:=", "Box1,Box2,Polyline1"), Array("NAME:ProjectSheetParameters")</pre>

PurgeHistory

Purges the history of a specified object.

UI Access	Modeler > Purge History.		
Parameters	Name	Type	Description

	<code><SelectionsArray></code>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	<code>PurgeHistory (<SelectionsArray>)</code>		
Python Example	<pre>oEditor.PurgeHistory(["NAME:Selections", "Selections:=", "Box2", "NewPartsModelFlag:=", "Model"])</pre>		

VB Syntax	<code>PurgeHistory <SelectionsArray></code>		
VB Example	<pre>oEditor.PurgeHistory Array("NAME:Selections", "Selections:=", "Box2", "NewPartsModelFlag:=", "Model")</pre>		

ReplaceWith3DComponent

Replaces the selection with a 3D component.

UI Access	None.		
Parameters	Name	Type	Description

	<CreateData>	Array	Structured array.
	<DesignData>	Array	Structured array.
	<ImageFile>	Array	Structured array.
Return Value	None.		

Python Syntax	ReplaceWith3DComponent(<CreateData>, <DesignData>, <ImageFile>)
Python Example	<pre>oEditor.ReplaceWith3DComponent(["NAME:CreateData", "ComponentName:=", "CoaxBend", "Company:=", "", "Company URL:=", "", "Model Number:=", "", "Help URL:=", "", "Version:=", "1.0", "Notes:=", "", "IconType:=", "", "Owner:=", "Jane Doe", "Email:=", "jdoe@email.com", "Date:=", "3:46:31 PM Jul 26, 2020", "HasLabel:=", False,</pre>

```

    "IncludedParts:=", ["outer","teflon","inner","teflon_1"],
    "HiddenParts:=", [],
    "IncludedCS:=", [],
    "DefaultHandle:=", "Global",
    "IncludedParameters:=", ["bend_angle"],
    "ParameterDescription:=", ["bend_angle:=", ""]
  ],
  ["NAME:DesignData",
    "Excitations:=", ["1","2"]
  ],
  ["NAME:ImageFile",
    "ImageFile:=", ""
  ]
)

```

VB Syntax	ReplaceWith3DComponent <CreateData>, <DesignData>, <ImageFile>
VB Example	<pre> oEditor.ReplaceWith3DComponent Array("NAME:CreateData", "ComponentName:=", _ "ConnectorOnly_wBCs", "Company:=", "", "Company URL:=", "", "Model Number:=", "", "Help URL:=", _ "", "Version:=", "1.0", "Notes:=", "", "IconType:=", "", "Owner:=", _ </pre>


```
"MyName", "Email:=", "My@email.com", "Date:=", "3:07:37 PM Jun 25, 2019", _
"HasLabel:=", false, "IsEncrypted:=", false, "AllowEdit:=", false, _
"SecurityMessage:=", "", "Password:=", "", "EditPassword:=", "", _
"PasswordType:=", "UnknownPassword", "HideContents:=", true, "ReplaceNames:=", true, _
"ComponentOutline:=", "None", "IncludedParts:=", _
Array("pin", "solder1", "ConnectorDielectric", _
"ConnectorOuter", "pcap", "solderr"), "HiddenParts:=", Array(), _
"IncludedCS:=", Array(), "DefaultHandle:=", _
"Global", "IncludedParameters:=", Array("lcon", "lpin", "rpin"), _
"ParameterDescription:=", Array("lcon:=", "", "lpin:=", "", "rpin:=", ""), "IsLi-
censed:=", false, _
"LicensingDllName:=", "", "VendorComponentIdentifier:=", _
"", "PublicKeyFile:=", ""), Array("NAME:DesignData", "Boundaries:=", Array( _
"PinSurface", "Connect"), "Excitations:=", Array("1", "2"), "MeshOperations:=", Array(
_
"Length1", "SkinDepth1")), "", Array("NAME:ImageFile", "ImageFile:=", "")
```

Section

Creates a 2D cross-section of the selection in the specified plane.

UI Access	Modeler > Surface > Section.		
Parameters	Name	Type	Description

	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. <pre>Array("NAME:SectionToParameters", "CreateNewObjects:=" , <boolean>, "SectionPlane:=" , <string "XY", "YZ", or "ZX">, "SectionCrossObject:=" , <boolean>)</pre>
Return Value	None.		

Python Syntax	Section (<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.Section(["NAME:Selections", "Selections:=" , "Cone1", "NewPartsModelFlag:=" , "Model"], ["NAME:SectionToParameters", "CreateNewObjects:=" , True, "SectionPlane:=" , "XY", "SectionCrossObject:=" , False])</pre>

VB Syntax	Section <SelectionsArray> <Parameters>
VB Example	<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.

UI Access	Modeler > Boolean > Separate Bodies.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	SeparateBody (<SelectionsArray>)
Python Example	<pre>oEditor.SeparateBody(["NAME:Selections",</pre>

	<pre>"Selections:=", "Rectangle1,Circle1", "NewPartsModelFlag:=", "Model"])</pre>

VB Syntax	SeparateBody <SelectionsArray>
VB Example	<pre>oEditor.SeparateBody Array("NAME:Selections", "Selections:=", "Rectangle1,Circle1", "NewPartsModelFlag:=", "Model")</pre>

SetModelUnits

Sets the model units.

UI Access	Modeler > Units.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre>Array("NAME:Units Parameter", "Units:=", <string>, "Rescale:=", <boolean True to rescale model; else False>)</pre> To see valid unit strings, select Modeler > Units.
Return Value	None.		

Python Syntax	SetModelUnits (<Parameters>)
Python Example	<pre>oEditor.SetModelUnits(["NAME:Units Parameter", "Units:=", "km", "Rescale:=", False])</pre>

VB Syntax	SetModelUnits <Parameters>
VB Example	<pre>oEditor.SetModelUnits Array("NAME:Units Parameter", "Units:=", "km", "Rescale:=", False)</pre>

SetWCS

Sets the working coordinate system.

UI Access	Modeler > Coordinate System > Set Working CS.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array.

			<pre>Array("NAME:SetWCS Parameter", "Working Coordinate System:=", <string CS name>, "RegionDepCSOk:=" , <boolean True if region-dependent; else False)</pre>
Return Value	None.		

Python Syntax	SetWCS (<Parameters>)
Python Example	<pre>oEditor.SetWCS(["NAME:SetWCS Parameter", "Working Coordinate System:=", "Global", "RegionDepCSOk:=", False])</pre>

VB Syntax	SetWCS <Parameters>
VB Example	<pre>oEditor.SetWCS Array("NAME:SetWCS Parameter", "Working Coordinate System:=", "Global", "RegionDepCSOk:=", False)</pre>

ShowWindow

Opens the active 3D Modeler window.

UI Access	None.
Parameters	None.
Return Value	None.

Python Syntax	ShowWindow
Python Example	<code>oEditor.ShowWindow()</code>

VB Syntax	ShowWindow
VB Example	<code>oEditor.ShowWindow</code>

Simplify

Converts a complex MCAD object into simpler primitives which are easy to mesh and solve.

UI Access	Modeler > Model Preparation > Simplify.		
Parameters	Name	Type	Description
	<Selections>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array.
			<code>Array("NAME:SimplifyParameters", "Type:=", <string fit type>, "ExtrusionAxis:=", <string>,</code>

			"Cleanup:=", <boolean>, "Splitting:=", <boolean>, "SeparateBodies:=", <boolean>, "CloneBody:=", <boolean>, "Generate Primitive History:=", <boolean>, "NumberPointsCurve:=", <integer>, "LengthThresholdCurve:=", <integer>
	<CreateGroup>	Array	Structured array. Array("CreateGroupsForNewObjects:=", <boolean>)
Return Value	None.		

Python Syntax	Simplify (<Selections>, <Parameters>, <CreateGroup>)
Python Example	<pre> oEditor.Simplify(["NAME:Selections", "Selections:=", "Cylinder1,Box2", "NewPartsModelFlag:=", "Model"], ["NAME:SimplifyParameters", "Type:=", "Polygon Fit", "ExtrusionAxis:=", "Auto", "Cleanup:=", True, "Splitting:=", True, </pre>

	<pre> "SeparateBodies:=", False, "CloneBody:=", False, "Generate Primitive History:=", True, "NumberPointsCurve:=", 3, "LengthThresholdCurve:=", 20], ["CreateGroupsForNewObjects:=", True]) </pre>

VB Syntax	Simplify <Selections>, <Parameters>, <CreateGroup>
VB Example	<pre> oEditor.Simplify Array("NAME:Selections", "Selections:=", "Cylinder1,Box2", "NewPartsModelFlag:=", "Model"), Array("NAME:SimplifyParameters", "Type:=", "Polygon Fit", "ExtrusionAxis:=", "Auto", "Cleanup:=", true, "Splitting:=", true, "SeparateBodies:=", false, </pre>

	<pre> CloneBody:=", false, Generate Primitive History:=", true, NumberPointsCurve:=", 3, LengthThresholdCurve:=", 20), Array("CreateGroupsForNewObjects:=", true) </pre>

Split

Splits the specified object(s) along a plane.

UI Access	Modeler > Boolean > Split.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. <pre> Array("NAME:SplitToParameters", "SplitPlane:=" , <string "XY", "YZ", "ZX", or "Dummy">, "WhichSide:=" , <string "PositiveOnly", "Neg- ativeOnly" or "Both">, "ToolType:=" , <string "PlaneTool" or "FaceTool">, "ToolEntityID:=" , <-1 if using SplitPlane; else FaceID>, "SplitCrossingObjectsOnly:=", <boolean>, "DeleteInvalidObjects:=", <boolean>) </pre> You can split an object either along an existing plane , or by creating a plane using an edge.

			For existing plane: • SplitPlane is "XY", "YZ", or "ZX" • ToolType is "PlaneTool" • ToolEntityID is -1 For an edge: • SplitPlane is "Dummy" • ToolType is "EdgeTool" • ToolEntityID is the face ID
Return Value	None.		

Python Syntax	Split(<SelectionsArray>, <Parameters>)		
Python Example	<pre>oEditor.Split(["NAME:Selections", "Selections:=", "NewPartsModelFlag:=",], ["NAME:SplitToParameters", "SplitPlane:=", "WhichSide:=", "ToolType:=" </pre>		

	<pre> "ToolEntityID:=" , -1, "SplitCrossingObjectsOnly:=", False, "DeleteInvalidObjects:=", True l) </pre>

VB Syntax	Split <SelectionsArray> <Parameters>
VB Example	<pre> oEditor.Split Array("NAME:Selections", "Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model") Array("NAME:SplitToParameters", "SplitPlane:=" , "XY", "WhichSide:=" , "PositiveOnly", "ToolType:=" , "PlaneTool", "ToolEntityID:=" , -1, "SplitCrossingObjectsOnly:=", False, "DeleteInvalidObjects:=", True) </pre>

Stitch

Stitches selected sheets.

UI Access	Modeler > Model Preparation > Stitch Sheets.

Parameters	Name	Type	Description
	<Selections>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array("NAME:StitchParameters", "MaxTol:=", <integer maximum tolerance>)
Return Value	None.		

Python Syntax	Stitch (<Selections>, <Parameters>)
Python Example	<pre>oEditor.Stitch(["NAME:Selections", "Selections:=", "Rectangle3,Rectangle4", "NewPartsModelFlag:=", "Model"], ["NAME:StitchParameters", "MaxTol:=", -1])</pre>

VB Syntax	Stitch <Selections>, <Parameters>
VB Example	<pre>oEditor.Stitch Array("NAME:Selections",</pre>

	<pre> "Selections:=", "Rectangle3,Rectangle4", "NewPartsModelFlag:=", "Model"), Array("NAME:StitchParameters", "MaxTol:=", -1) </pre>

Subtract

Subtracts the specified object(s).

UI Access	Modeler > Boolean > Subtract.		
Parameters	Name	Type	Description
	<Selections>	Array	Structured array. <pre> Array("NAME:Selections", "Blank Parts:=" , <string object name(s)>, "Tool Parts:=" , <string object name(s)>) </pre> The Tool Parts object is the object being removed. The Blank Parts object has any Tool Parts overlap removed. Either string can contain more than one object.
	<Parameters>	Array	Structured array. <pre> Array("NAME:SubtractParameters", "KeepOriginals:=" , <boolean>) </pre>
Return Value	None.		

Python Syntax	Subtract(<Selections>, <Parameters>)
Python Example	<pre>oEditor.Subtract(["NAME:Selections", "Blank Parts:=", "Rectangle1", "Tool Parts:=", "Rectangle2"], ["NAME:SubtractParameters", "KeepOriginals:=", False])</pre>

VB Syntax	Subtract <Selections> <Parameters>
VB Example	<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.

UI Access	Modeler > Surface > Sweep Faces Along Normal		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<parameters>	Array	Structured array. <pre>Array("NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", <faceIDarray>, "LengthOfSweep:=", "<value><units>")</pre>
Return Value	None		

Python Syntax	SweepFacesAlongNormal(<SelectionsArray> <parameters>)
Python Example	<pre>oEditor.SweepFacesAlongNormal(["NAME:Selections", "Selections:=", "Rectangle1", "NewPartsModelFlag:=", "Model"], ["NAME:Parameters", "NAME:SweepFaceAlongNormalToParameters", "FacesToDetach:=", [183], "LengthOfSweep:=", "0.1mm"])</pre>

VB Syntax	SweepFacesAlongNormal <SelectionsArray> <parameters>
VB Example	<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.

UI Access	Modeler > Surface > Thicken Sheet		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<parameters>	Array	Structured array. Array("NAME:SheetThickenParameters", "Thickness:=" , <string>, "BothSides:=" , <boolean>)
Return Value	None.		

Python Syntax	ThickenSheet (<SelectionsArray> <parameters>)
Python Example	<pre> oEditor.ThickenSheet(["NAME:Selections", "Selections:=" , "Rectangle3", "NewPartsModelFlag:=" , "Model"], ["NAME:SheetThickenParameters", "Thickness:=" , "0.01mm", "BothSides:=" , False]) </pre>

VB Syntax	ThickenSheet <SelectionsArray> <parameters>
VB Example	<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	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<parameters>	Array	Structured array. <pre>Array("NAME:Parameters", Array("NAME:UncoverFacesParameters", "FacesToUncover:=" , <array of face IDs>))</pre>
Return Value	None.		

Python Syntax	UncoverFaces(<SelectionsArray> <parameters>)
Python Example	<pre>oEditor.UncoverFaces(["NAME:Selections", "Selections:=" , "Box1", "NewPartsModelFlag:=" , "Model"], ["NAME:Parameters", ["NAME:UncoverFacesParameters",</pre>

	<pre> "FacesToUncover:=" , [12,16,18]]]) </pre>

VB Syntax	UncoverFaces <SelectionsArray> <parameters>
VB Example	<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: SelectionsArray .
	<parameters>	Array	Structured array. <pre> Array("NAME:UniteParameters", "KeepOriginals:=" , <boolean>, </pre>

			<code>"TurnOnNBodyBoolean:=", <boolean>)</code> Note: TurnOnNBodyBoolean is set to True by default. Enabled by Parasolid, this unites <i>n</i> bodies in a single step, rather than uniting them one by one.
Return Value	None.		

Python Syntax	<code>Unite(<SelectionsArray>, <parameters>)</code>
Python Example	<pre>oEditor.Unite(["NAME:Selections", "Selections:=", "Rectangle1,Rectangle2"], ["NAME:UniteParameters", "KeepOriginals:=", False, "TurnOnNBodyBoolean:=", True])</pre>

VB Syntax	<code>Unite <SelectionsArray> <parameters></code>

VB Example	<pre>oEditor.Unite Array("NAME:Selections", "Selections:=", "Rectangle1,Rectangle2") Array("NAME:UniteParameters", "KeepOriginals:=", false, "TurnOnNBodyBoolean:=", true)</pre>

Ungroup

Ungroups a specified history tree group.

UI Access	Modeler > Group > Ungroup		
Parameters	Name	Type	Description
	<Groups>	Array	Structured array. Array("Groups:=", <array of group names to ungroup>)
Return Value	None		

Python Syntax	Ungroup(<Groups>)
Python Example	oEditor.Ungroup(["Groups:=", ["Group1"]])

VB Syntax	Ungroup <Groups>
VB Example	oEditor.Ungroup Array("Groups:=", Array("Group1"))

WrapSheet

Wraps a sheet object to another object.

UI Access	None.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<Parameters>	Array	Structured array. Array ("NAME:WrapSheetParameters", "Imprinted:=", <boolean>)
Return Value	None.		

Python Syntax	WrapSheet (<SelectionsArray>, <Parameters>)
Python Example	<pre>oEditor.WrapSheet(["NAME:Selections", "Selections:=", "Rectangle1,Box1"], ["NAME:WrapSheetParameters", "Imprinted:=", True])</pre>

VB Syntax	WrapSheet <SelectionsArray> <Parameters>
VB Example	<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](#)

[GeometryCheckAndAutofix](#)

[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](#)[PageSetup](#)[RenamePart](#)[SetPropValue \[Modeler\]](#)

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.

UI Access	N/A		
Parameters	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.

	<i><defFolderName></i>	String	Library type (by definition folder name).
	<i><newTimeStamp></i>	String	New timestamp string (time_t as integer, number of seconds since 1/1/1970 12:00am), default is current time
	<i><replaceExisting></i>	Boolean	True to replace existing, False to choose a new unique name if an existing definition is found, default is False
Return Value	Property scripting object for the definition.		

Python Syntax	AddDefinitionFromLibFile(<i><FilePath></i> , <i><defName></i> , <i><defFolderName></i> , <i><newTimeStamp></i> , <i><replaceExisting></i>)
Python Example	<pre> oProject = oDesktop.NewProject() oProject.InsertDesign("HFSS", "HFSSDesign1", "DrivenModal", "") oDesign = oProject.SetActiveDesign("HFSSDesign1") oEditor = oDesign.SetActiveEditor("3D Modeler") oEditor.CreateBox(["NAME:BoxParameters", "XPosition:=", "YPosition:=", "ZPosition:=", "-0.4mm", "-1mm", "0mm",]) </pre>

```

        "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
                ]
            ]
        ]
    ]
)
```

	]
	])

AssignSurfaceMaterial

Assigns a material to specified surfaces.

UI Access	N/A		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
	<AttributesArray>	Array	Structured array. See: AttributesArray . This script supports the following attributes: - MaterialValue - SolveInside - ShellElement - ShellElementThickness - IsMaterialEditable - UseMaterialAppearance - IsLightweight
Return Value	None.		

Python Syntax	AssignSurfaceMaterial(<SelectionsArray>, <AttributesArray>)

Python Example	<pre>oEditor.AssignSurfaceMaterial(["NAME:Selections", "AllowRegionDependentPartSelectionForPMLCreation:=", True, "AllowRegionSelectionForPMLCreation:=", True, "Selections:=" , "Rectangle1"], ["NAME:Attributes", "MaterialValue:=" , "diamond", "SolveInside:=" , False, "ShellElement:=" , False, "ShellElementThickness:=" , "nan", "IsMaterialEditable:=" , True, "UseMaterialAppearance:=" , False, "IsLightweight:=" , False])</pre>
VB Syntax	AssignSurfaceMaterial <SelectionsArray> <AttributesArray>

VB Example	<pre>oEditor.AssignSurfaceMaterial Array("NAME:Selections", "AllowRegionDependentPartSelectionForPMLCreation:=", true, "AllowRegionSelectionForPMLCreation:=", true, "Selections:=" , "Rectangle1") Array("NAME:Attributes", "MaterialValue:=" , "diamond", "SolveInside:=" , false, "ShellElement:=" , false, "ShellElementThickness:=" , "nan", "IsMaterialEditable:=" , true, "UseMaterialAppearance:=" , false, "IsLightweight:=" , false)</pre>

BreakUDMConnection

Breaks a current UDM connection to Discovery.

UI Access	Break Connection.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray.
Return Value	None.		

Python Syntax	BreakUDMConnection (<SelectionsArray>)
Python Example	<pre>oEditor.BreakUDMConnection(["NAME:Selections", "Selections:=", "Discovery1"])</pre>

VB Syntax	BreakUDMConnection <SelectionsArray>
VB Example	<pre>oEditor.BreakUDMConnection Array("NAME:Selections", "Selections:=", "")</pre>

ChangeProperty

Changes the properties of an object in the history tree.

UI Access	Right-click an object in the History Tree and select Properties .		
Parameters	Name	Type	Description
	<propertyArgs>	Array	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 Automation tab.
Return Value	None.		

Python Syntax	ChangeProperty(<propertyArgs>)
Python Example	Example: Changing the Position of a Box <pre>oEditor.ChangeProperty(["NAME:AllTabs", ["NAME:Geometry3DCmdTab", ["NAME:PropServers" , "Box1:CreateBox:1"], ["NAME:ChangedProps", ["NAME:Position", "X:" , "0.35in", "Y:" , "0.55in", "Z:" , "0in"]]]])</pre> Example: Changing a Box's Material and Wireframe Display <pre>oEditor.ChangeProperty(["NAME:AllTabs", ["NAME:Geometry3DAttributeTab", [</pre>

```

        "NAME:PropServers" ,
        "Box1"
    ],
    [
        "NAME:ChangedProps",
        [
            "NAME:Material",
            "Value:="          , "\"vacuum\""
        ],
        [
            "NAME:Display Wireframe",
            "Value:="          , True
        ]
    ]
]
])
Example: Change Default Handle (reference coordinate system)
oEditor.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:General",
        [
            "NAME:PropServers",
            "HFSSDesign1_1"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Handle",
                "Value:="          , " NotchOutY"
            ]
        ]
    ]
]

```

	<pre>]]])</pre>

VB Syntax	ChangeProperty <propertyArgs>
VB Example	Example: Changing the Position of a Box <pre>oEditor.ChangeProperty Array("NAME:AllTabs", Array("NAME:Geometry3DCmdTab", Array("NAME:PropServers" , "Box1: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 <pre>oEditor.ChangeProperty(Array("NAME:AllTabs", Array("NAME:Geometry3DAttributeTab", Array("NAME:PropServers",</pre>

```

        "Box1"
    ),
    Array("NAME:ChangedProps",
        Array("NAME:Material",
            "Value:=", "\"vacuum\""
        ),
        Array("NAME:Display Wireframe",
            "Value:=", True
        )
    )
)
)
)

```

Example: Change Default Handle (reference coordinate system)

```

oEditor.ChangeProperty
Array("NAME:AllTabs",
    Array("NAME:General",
        Array("NAME:PropServers",
            "HFSSDesign1_1"
        ),
        Array("NAME:ChangedProps",
            Array("NAME:Handle",
                "Value:=", " NotchOutY"
            )
        )
    )
)
)

```

CloseAllWindows[Editor]

Closes all windows belong to current 3D Modeler editor.

UI Access	N/A
Parameters	None.
Return Value	None.

Python Syntax	CloseAllWindows()
Python Example	<code>oEditor.CloseAllWindows()</code>

VB Syntax	CloseAllWindows
VB Example	<code>oEditor.CloseAllWindows</code>

Delete

Deletes the specified object(s).

UI Access	Edit > Delete.		
Parameters	Name	Type	Description
	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	None.		

Python Syntax	Delete(<SelectionsArray>)
Python Example	<pre>oEditor.Delete(["NAME:Selections", "Selections:=", "Rectangle1,Rectangle2"])</pre>

VB Syntax	Delete <SelectionsArray>
VB Example	oEditor.Delete Array("NAME:Selections", "Selections:=", "Rectangle1,Rectangle2")

FitAll

Fits the design to the modeling area.

UI Access	View > Fit All > All Views.
Parameters	None.
Return Value	None.

Python Syntax	FitAll()
Python Example	oEditor.FitAll()

VB Syntax	FitAll()

VB Example	<pre>Set oEditor = oDesign.SetActiveEditor("3D Modeler") oEditor.FitAll()</pre>

GenerateAllUserDefinedModels

Generates all user defined models.

UI Access	N/A
Parameters	None.
Return Value	Array of models generated.

Python Syntax	GenerateAllUserDefinedModels ()
Python Example	<pre>oEditor.GenerateAllUserDefinedModels ()</pre>

VB Syntax	GenerateAllUserDefinedModels
VB Example	<pre>oEditor.GenerateAllUserDefinedModels</pre>

GenerateUserDefinedModel

Generates specified user defined model(s).

UI Access	N/A		
Parameters	Name	Type	Description

	<SelectionsArray>	Array	Structured array. See: SelectionsArray .
Return Value	Array of models generated.		

Python Syntax	GenerateUserDefinedModel (<SelectionsArray>)
Python Example	<pre>oEditor.GenerateUserDefinedModel(["NAME:Selections", "Selections:=", "model1", model2"])</pre>

VB Syntax	GenerateAllUserDefinedModels
VB Example	<pre>oEditor.GenerateUserDefinedModel Array("NAME:Selections", "Selections:=", "model1", model2")</pre>

GeometryCheckAndAutofix

Use: Runs Geometry Check and optionally applies autofixes.

Command: HFSS 3D Layout > Geometry Check

Syntax: GeometryCheckAndAutofix <ChecksArray>,

minimum_area_meters_squared,

<FixesArray>

Return Value: None

Parameters: <ChecksArray> - Array("NAME:checks", <check 1>, <check 2>, ..., <check n>)

Specify the checks that should be included. Specifying fewer checks may speed up execution but may also result in less problems reported in the message manager (or the logfile) and consequently less problems that can be fixed by autofixes.

The following are valid checks that can be specified:

- "Self-Intersecting Polygons"
- "Disjoint Nets (Floating Nodes)"
- "DC-Short Errors"
- "Identical/Overlapping Vias"
- "Misalignments"

There may be no checks, all 5 of the checks, or anything in between. The order that checks are specified in is not relevant.

minimum_area_meters_squared

Specify a decimal value for the minimum area (e.g. .000015) optionally in scientific notation (e.g. 2E-006). Cutouts smaller than this minimum area may be ignored during validation check.

<FixesArray> - Array("NAME:fixes", <fix 1>, <fix 2>, ..., <fix n>)

Specify the autofixes that should be applied if they are found by a check.

The following are valid fixes that can be specified:

- "Self-Intersecting Polygons"
- "Disjoint Nets",
- "Identical/Overlapping Vias"

- "Traces-Inside-Traces Errors"
- "Misalignments (Planes/Traces/Vias)"

There may be no fixes specified, all 5 fixes specified, or anything in between. The order that fixes are specified in is not relevant.

Example:

```
oEditor.GeometryCheckAndAutofix _
  Array("NAME:checks", "Self-Intersecting Polygons", _
    "Disjoint Nets (Floating Nodes)", "DC-Short Errors", _
    "Identical/Overlapping Vias", "Misalignments"), _
    "minimum_area_meters_squared:=", 2E-006, _
  Array("NAME:fixes", "Self-Intersecting Polygons", _
    "Disjoint Nets", "Identical/Overlapping Vias", _
    "Traces-Inside-Traces Errors", _
    "Misalignments (Planes/Traces/Vias)")
```

GetBodyNamesByPosition

Returns the names of objects that contact a specified point.

UI Access	N/A		
Parameters	Name	Type	Description
	<positionParameters>	Array	Structured array containing position coordinates for active coordinate system. Array("NAME:Parameters", "XPosition:=", <string>,

	"YPosition:=", <string>, "ZPosition:=", <string>)		
Return Value	Array containing string object names.		

Python Syntax	GetBodyNamesByPosition (< <i>positionParameters</i> >)
Python Example	<pre>oEditor.GetBodyNamesByPosition(["NAME:Parameters", "XPosition:=", "0mm", "YPosition:=", "15mm", "ZPosition:=", "0mm"])</pre>

VB Syntax	GetBodyNamesByPosition < <i>positionParameters</i> >
VB Example	<pre>oEditor.GetBodyNamesByPosition Array("NAME:Parameters", _ "XPosition:=", "0mm", _ "YPosition:=", "15mm", _ "ZPosition:=", "0mm")</pre>

GetChildNames [Modeler]

Returns the names of children for a specified input. For 3D Components and UDMs, these commands do not return parts, coordinate systems, plans, as top-level modeler children.

Note:

This command is not supported by the EMIT and Circuit design types.

UI Access	N/A		
Parameters	Name	Type	Description
	<category>	String	<i>Optional.</i> Passing no input returns the list of possible strings: "AllParts" – Returns the names of all parts. "CoordinateSystems" – Returns the names of all coordinate systems. "Groups" – Returns the names of all groups. "Lists" – Returns the names of all lists. "ModelParts" – Returns names of model parts. "NonModelParts" – Returns the names of non-model parts. "Planes" – Returns the names of all planes. "Points" – Returns the names of all points. "SubmodelDefinitions" – Returns the names of sub-model definitions.
Return Value	Array containing object names in the selected category.		

Python Syntax	GetChildNames(<category>)
Python Example	Standalone Example: <pre>oEditor.GetChildNames("ModelParts") oEditor.GetChildNames("Points")</pre> Example used in Conjunction with GetChildObject and GetPropNames: <pre>oDesign = oProject.GetActiveDesign() oModel = oDesign.GetChildObject("3D Modeler") oModel.GetChildTypes()</pre> - This returns an array containing strings: "ModelParts", "AllParts", "NonModelParts", "CoordinateSystems", "Points", "Planes", "SubmodelDefinitions", "Groups", and "Lists". <pre>oModel.GetChildNames("ModelParts")</pre> - This returns an array containing string model parts. For example: "WG_Interior", "WG", "MT_Interior", "MT", "Box2", "Cylinder1". <pre>oWGi = oModel.GetChildObject("WG_Interior")</pre> - This sets the object WG_Interior to variable oWGi. <pre>oWGi.GetPropNames()</pre> - This returns property names from child object assigned to oWGi. In this case: "Name", "Material", "Material/SIValue", "Material/EvaluatedValue", "Solve Inside", "Orientation", "Orientation/Choices", "Model", "Display Wireframe", "Material Appearance", "Color", "Color/Red", "Color/Green", "Color/Blue", and "Transparent".

VB Syntax	GetChildNames <category>
VB Example	Standalone Example: <pre>oEditor.GetChildNames "ModelParts" oEditor.GetChildNames "Points"</pre> Example used in Conjunction with GetChildObject and GetPropNames: <pre>Dim oAnsoftApp Dim oDesktop Dim oProject Dim oDesign Dim oEditor Set oDesign = oProject.GetActiveDesign() Set oModel = oDesign.GetChildObject "3D Modeler" oModel.GetChildTypes() • This returns an array containing strings: "ModelParts", "AllParts", "NonModelParts", "CoordinateSystems", "Points", "Planes", "SubmodelDefinitions", "Groups", and "Lists". oModel.GetChildNames "ModelParts" • This returns an array containing string model parts. For example: "WG_Interior", "WG", "MT_Interior", "MT", "Box2", "Cylinder1". Set oWGi = oModel.GetChildObject "WG_Interior" • This sets the object WG_Interior to variable oWGi. oWGi.GetPropNames()</pre>

	This returns property names from child object assigned to oWGi. In this case: "Name", "Material", "Material/SIValue", "Material/EvaluatedValue", "Solve Inside", "Orientation", "Orientation/Choices", "Model", "Display Wireframe", "Material Appearance", "Color", "Color/Red", "Color/Green", "Color/Blue", and "Transparent".

GetChildObject [Modeler]

Returns a 3D modeler child object, which can be assigned to a variable. Will return normally if there are no active objects. For 3D Components and UDMs, these commands do not return parts, coordinate systems, plans, as top-level modeler children.

Note: This command is not supported by the EMIT and Circuit design types.

UI Access	N/A		
Parameters	Name	Type	Description
	<object>	String	In this case, "3D Modeler".
Return Value	Returns the 3D Modeler object. In the examples below, it is assigned to the variable oEditor.		

Python Syntax	GetChildObject(<object>)
Python Example	Standalone Example: <pre>oDesign = oProject.GetActiveDesign() oEditor = oDesign.GetChildObject("3D Modeler")</pre> Example used in Conjunction with GetChildNames and GetPropNames:

	<pre>oDesign = oProject.GetActiveDesign() oModel = oDesign.GetChildObject("3D Modeler") oModel.GetChildTypes()</pre> • This returns an array containing strings: "ModelParts", "AllParts", "NonModelParts", "CoordinateSystems", "Points", "Planes", "SubmodelDefinitions", "Groups", and "Lists". <pre>oModel.GetChildNames("ModelParts")</pre> • This returns an array containing string model parts. For example: "WG_Interior", "WG", "MT_Interior", "MT", "Box2", "Cylinder1". <pre>oWGi = oModel.GetChildObject("WG_Interior")</pre> • This sets the object WG_Interior to variable oWGi. <pre>oWGi.GetPropNames()</pre> • This returns property names from child object assigned to oWGi. In this case: "Name", "Material", "Material/SIValue", "Material/EvaluatedValue", "Solve Inside", "Orientation", "Orientation/Choices", "Model", "Display Wireframe", "Material Appearance", "Color", "Color/Red", "Color/Green", "Color/Blue", and "Transparent".

VB Syntax	GetChildObject<object>
VB Example	Standalone Example: <pre>Dim oAnsoftApp Dim oDesktop Dim oProject Dim oDesign</pre>

```
Dim oEditor
Set oDesign = oProject.GetActiveDesign()
Set oEditor = oDesign.GetChildObject "3D Modeler"
```

Example used in Conjunction with [GetChildNames](#) and [GetPropNames](#):

```
Dim oAnsoftApp
```

```
Dim oDesktop
```

```
Dim oProject
```

```
Dim oDesign
```

```
Dim oEditor
```

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

```
Set oModel = oDesign.GetChildObject "3D Modeler"
```

```
oModel.GetChildTypes()
```

- This returns an array containing strings: "ModelParts", "AllParts", "NonModelParts", "CoordinateSystems", "Points", "Planes", "SubmodelDefinitions", "Groups", and "Lists".

```
oModel.GetChildNames "ModelParts"
```

- This returns an array containing string model parts. For example: "WG_Interior", "WG", "MT_Interior", "MT", "Box2", "Cylinder1".

```
Set oWGi = oModel.GetChildObject "WG_Interior"
```

- This sets the object WG_Interior to variable oWGi.

```
oWGi.GetPropNames()
```

	This returns property names from child object assigned to oWGi. In this case: "Name", "Material", "Material/SIValue", "Material/EvaluatedValue", "Solve Inside", "Orientation", "Orientation/Choices", "Model", "Display Wireframe", "Material Appearance", "Color", "Color/Red", "Color/Green", "Color/Blue", and "Transparent".

GetChildTypes [Modeler]

Gets child types of queried designs or editors obtained by [GetActiveProject\(\)](#) and [GetActiveDesign\(\)](#) commands. For 3D Components and UDMs, these commands do not return parts, coordinate systems, plans, as top-level modeler children.

Note:

This command is not supported by the EMIT and Circuit design types.

UI Access	N/A
Parameters	None.
Return Value	Array containing the names of child types.

Python Syntax	GetChildTypes()
Python Example	Standalone Example: <pre>oDesign = oProject.GetActiveDesign() oEditor = oDesign.SetActiveEditor("3D Modeler") oEditor.GetChildTypes()</pre> This returns an array containing strings: "ModelParts", "AllParts", "NonModelParts", "CoordinateSystems", "Points", "Planes", "SubmodelDefinitions", "Groups", and "Lists".

	Example Used in Conjunction with GetChildNames: <pre>oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() oDesign.GetChildNames() • This returns an array containing names of child objects for the design. For example: "Boundaries", "Nets", "Analysis", "Optimetrics", "Radiation", "Results", and "3D Modeler". oDesign.GetChildTypes() • This returns an array containing the child types. For example: "Module", "Editor", and "Variable".</pre>

VB Syntax	GetChildTypes()
VB Example	Standalone Example: <pre>Dim oDesign Dim oProject Dim oEditor Set oDesign = oProject.GetActiveDesign() Set oEditor = oDesign.SetActiveEditor "3D Modeler" oEditor.GetChildTypes() • This returns an array containing strings: "ModelParts", "AllParts", "NonModelParts", "CoordinateSystems", "Points", "Planes", "SubmodelDefinitions", "Groups", and "Lists".</pre> Example Used in Conjunction with GetChildNames:

	<pre>Dim oDesign Dim oProject Dim oDesign Set oDesign = oProject.GetActiveDesign() Set oDesign.GetChildNames()</pre> • This returns an array containing names of child objects for the design. For example: "Boundaries", "Nets", "Analysis", "Optimetrics", "Radiation", "Results", and "3D Modeler". <pre>Set oDesign.GetChildTypes()</pre> • This returns an array containing the child types. For example: "Module", "Editor", and "Variable".

GetEdgeByPosition

Returns the ID for edge(s) that contact a specified point.

UI Access	N/A		
Parameters	Name	Type	Description
	<positionParameters>	Array	Structured array. <pre>Array("NAME:EdgeParameters", "BodyName:=", <string object name>, "Xposition:=", <string>, "YPosition:=", <string>, "ZPosition:=", <string>)</pre> Note:

	<table><tr><td></td><td>For 2D XY Designs, ZPosition should be set to "0". For 2D RZ Designs, YPosition should be set to "0".</td></tr></table>		For 2D XY Designs, ZPosition should be set to "0". For 2D RZ Designs, YPosition should be set to "0".
	For 2D XY Designs, ZPosition should be set to "0". For 2D RZ Designs, YPosition should be set to "0".		
Return Value	Array containing string edge IDs.		

Python Syntax	GetEdgeByPosition (<positionParameters>)
Python Example	<pre>oEditor.GetEdgeByPosition(["NAME:EdgeParameters", "BodyName:", "Box1", "Xposition:", "10mm", "YPosition:", "0mm", "ZPosition:", "10mm"])</pre>

VB Syntax	GetEdgeByPosition <positionParameters>
VB Example	<pre>oEditor.GetEdgeByPosition Array("NAME:EdgeParameters", "BodyName:", "Box1", "Xposition:", "10mm", "YPosition:", "0mm",</pre>

	"ZPosition:=", "10mm")

GetEdgeIDsFromFace

Returns the edge IDs for a specified Face ID.

UI Access	N/A		
Parameters	Name	Type	Description
	<faceID>	Integer	ID of the specified face.
Return Value	Array containing string edge IDs.		

Python Syntax	GetEdgeIDsFromFace (<faceID>)
Python Example	oEditor.GetEdgeIDsFromFace (20)

VB Syntax	GetEdgeIDsFromFace <faceID>
VB Example	oEditor.GetEdgeIDsFromFace 20

GetEdgeIDsFromObject

Returns the edge IDs for a specified object.

UI Access	N/A		
Parameters	Name	Type	Description
	<object>	String	Object name.
Return Value	Array containing string edge IDs.		

Python Syntax	GetEdgeIDsFromObject (< <i>object</i> >)
Python Example	<code>oEditor.GetEdgeIDsFromObject ("Box1")</code>

VB Syntax	GetEdgeIDsFromObject < <i>object</i> >
VB Example	<code>oEditor.GetEdgeIDsFromObject "Box1"</code>

GetEdgeLength

Returns the length of edges for a specified Edge ID.

UI Access	N/A		
Parameters	Name	Type	Description
	< <i>edgeID</i> >	Integer	ID of the specified edge.
Return Value	Integer containing the length of found edges.		

Python Syntax	GetEdgeLength (< <i>EdgeID</i> >)
Python Example	<code>oEditor.GetEdgeLength (20)</code>

VB Syntax	GetEdgeLength < <i>edgeID</i> >

VB Example	<code>oEditor.GetEdgeLength 20</code>

GetEntityListIDByName

Returns a list of IDs in a given entity list (either object or face). See: [CreateEntityList](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<listName>	String	List name.
Return Value	Array containing string object IDs.		

Python Syntax	<code>GetEntityListIDByName (<listName>)</code>		
Python Example	<code>oEditor.GetEntityListIDByName ("MyList")</code>		

VB Syntax	<code>GetEntityListIDByName <listName></code>		
VB Example	<code>oEditor.GetEntityListIDByName "MyList"</code>		

GetFaceArea

Returns the area of a specified face.

UI Access	N/A		
Parameters	Name	Type	Description
	<faceID>	Integer	ID of specified face.
Return Value	Long integer of face area.		

Python Syntax	GetFaceArea (<faceID>)
Python Example	oEditor.GetFaceArea(19)

VB Syntax	GetFaceArea <faceID>
VB Example	oEditor.GetFaceArea 19

GetFaceCenter

Returns the center position of a specified face.

UI Access	N/A		
Parameters	Name	Type	Description
	<planarFaceID>	Long	Face ID
Return Value	Array containing string X, Y, Z coordinates.		

Python Syntax	GetFaceCenter (<planarFaceID>)
Python Example	oEditor.GetFaceCenter(12)

VB Syntax	GetFaceCenter <planarFaceID>

VB Example

```
oEditor.GetFaceCenter 12
```

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.

UI Access	N/A		
Parameters	Name	Type	Description
	<FaceParameters>	Array	Structured Array. Array("NAME:FaceParameters", "BodyName:=", <string>, "XPosition:=", <string>, "YPosition:=", <string>, "ZPosition:=", <string>)
Return Value	Integer Face ID.		

Python Syntax	GetFaceByPosition(<FaceParameters>)
Python Example	<pre>oEditor.GetFaceByPosition(["NAME:FaceParameters", "BodyName:=", "Box1",</pre>

	<pre>"XPosition:=", "0.2mm", "YPosition:=", "-0.2mm", "ZPosition:=", "0.4mm" l)</pre>

VB Syntax	GetFaceByPosition <FaceParameters>
VB Example	<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.

UI Access	N/A		
Parameters	Name	Type	Description
	<objectName>	String	Object name.
Return Value	Array containing string face IDs.		

Python Syntax	GetFaceIDs (<objectName>)
Python Example	oEditor.GetFaceIDs ('Box1')

VB Syntax	GetFaceIDs <objectName>
VB Example	oEditor.GetFaceIDs "Box1"

GetFaceIDsOfSheet

Returns the face IDs associated with a specified sheet object.

UI Access	N/A		
Parameters	Name	Type	Description
	<sheetName>	String	Name of specified sheet object.
Return Value	Array containing string face IDs.		

Python Syntax	GetFaceIDsOfSheet(<sheetName>)
Python Example	oEditor.GetFaceIDsOfSheet ('Sheet1')

VB Syntax	GetFaceIDsOfSheet <sheetName>
VB Example	oEditor.GetFaceIDsOfSheet "Sheet1"

GetMatchedObjectName

Returns all object names containing the input text string.

UI Access	N/A		
Parameters	Name	Type	Description
	<wildcardText>	String	Text string present in object name(s).
Return Value	Array containing string object names.		

Python Syntax	GetMatchedObjectName (<wildcardText>)		
Python Example	oEditor.GetMatchedObjectName ('Box*')		
	oEditor.GetMatchedObjectName ('?ox?')		

VB Syntax	GetMatchedObjectName <wildcardText>		
VB Example	oEditor.GetMatchedObjectName "Box*"		

GetModelBoundingBox

Returns the bounding box of the current model using the global coordinate system as reference. The outputs are in the defined modeler units. Only modeled objects are considered for this computation (non-modeled objects are ignored).

UI Access	N/A		
Parameters	None.		
Return Value	Array containing string Xmin, Ymin, Zmin, Xmax, Ymax, and Zmax values of the bounding box in the global coordinate system		

Python Syntax	GetModelBoundingBox()
Python Example	<code>oEditor.GetModelBoundingBox()</code>

VB Syntax	GeModelBoundingBox
VB Example	<code>oEditor.GeModelBoundingBox</code>

GetModelUnits

Returns the model's unit of measure.

UI Access	N/A
Parameters	None.
Return Value	String containing unit of measure.

Python Syntax	GetModelUnits()
Python Example	<code>oEditor.GetModelUnits()</code>

VB Syntax	GetModelUnits
VB Example	<code>oEditor.GetModelUnits</code>

GetNumObjects

Returns the number of objects in a design.

UI Access	N/A
Parameters	None.
Return Value	Integer number of objects.

Python Syntax	GetNumObjects()
Python Example	<code>oEditor.GetNumObjects()</code>

VB Syntax	GetNumObjects
VB Example	<code>oEditor.GetNumObjects(</code>

GetObjectIDByName

Given an object's name, returns its ID. IDs are used with [CreateEntityList](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<code><objectName></code>	String	Object name.
Return Value	Integer object ID.		

Python Syntax	GetObjectIDByName(<code><objectName></code>)

Python Example	<code>oEditor.GetObjectIDByName('Box1')</code>

VB Syntax	<code>GetObjectIDByName <objectName></code>
VB Example	<code>oEditor.GetObjectIDByName "Box1"</code>

GetObjectName

Returns an object's name from its specified base index (creation order).

UI Access	N/A		
Parameters	Name	Type	Description
	<index>	Integer	Base index (where '0' is the first item created)
Return Value	String containing object name.		

Python Syntax	<code>GetObjectName(<index>)</code>
Python Example	<code>oEditor.GetObjectName(3)</code>

VB Syntax	<code>GetObjectName <index></code>
VB Example	<code>oEditor.GetObjectName 3</code>

GetObjectNameByEdgeID

Returns an object name given an edge ID.

UI Access	N/A		
Parameters	Name	Type	Description
	<EdgeID>	Integer	The edge ID.
Return Value	String containing object name.		

Python Syntax	GetObjectByNameByEdgeID(<EdgeID>)		
Python Example	<code>oEditor.GetObjectByNameByEdgeID(88)</code>		

VB Syntax	GetObjectByNameByEdgeID <EdgeID>		
VB Example	<code>oEditor.GetObjectByNameByEdgeID 10</code>		

GetObjectByNameByFaceID

Returns an object name given a face ID.

UI Access	N/A		
Parameters	Name	Type	Description
	<FaceID>	Integer	The face ID.
Return Value	String containing object name.		

Python Syntax	GetObjectByNameByFaceID (<FaceID>)
Python Example	<code>oEditor.GetObjectByNameByFaceID(88)</code>

VB Syntax	GetObjectByNameByFaceID <FaceID>
VB Example	<code>oEditor.GetObjectByNameByFaceID 10</code>

GetObjectByNameByID

Returns an object name given an ID.

UI Access	N/A		
Parameters	Name	Type	Description
	<ObjID>	Integer	The object ID.
Return Value	String containing object name.		

Python Syntax	GetObjectByNameByID(<ObjID>)
Python Example	<code>oEditor.GetObjectByNameByID(88)</code>

VB Syntax	GetObjectByNameByID <ObjID>
VB Example	<code>oEditor.GetObjectByNameByID 10</code>

GetObjectByNameByVertexID

Returns an object name given a vertex ID.

UI Access	N/A		
Parameters	Name	Type	Description
	<VertexID>	Integer	The vertex ID.
Return Value	String containing object name.		

Python Syntax	GetObjectByNameByVertexID(<VertexID>)		
Python Example	oEditor.GetObjectByNameByVertexID(88)		

VB Syntax	GetObjectByNameByVertexID <VertexID>		
VB Example	oEditor.GetObjectByNameByVertexID 10		

GetObjectsByMaterial

Returns a list of objects of a specified material.

UI Access	N/A		
Parameters	Name	Type	Description
	<material>	String	Material name.
Return Value	Array containing string object names.		

Python Syntax	<code>GetObjectsByMaterial (<material>)</code>
Python Example	<code>oEditor.GetObjectsByMaterial ('copper')</code>

VB Syntax	<code>GetObjectsByMaterial <material></code>
VB Example	<code>oEditor.GetObjectsByMaterial "copper"</code>

GetObjectShapeType

Returns the shape type of a specified object.

UI Access	N/A		
Parameters	Name	Type	Description
	<objName>	String	Name of specified object.
	<csName>	String	Name of coordinate system of the object.
Return Value	String containing shape type.		

Python Syntax	<code>GetObjectShapeType(<objName>, <csName>)</code>
Python Example	<code>oEditor.GetObjectShapeType ("box1", "Global")</code>

VB Syntax	<code>GetObjectShapeType <objName>, <csName></code>
VB Example	<code>oEditor.GetObjectShapeType "box1", "Global"</code>

GetObjectsInGroup

Returns a list of objects in a specified group.

UI Access	N/A		
Parameters	Name	Type	Description
	<group>	String	Group name. One of: • "<materialName>" • "<assignmentName>" • "Non Model" • "Solids" • "Unclassified" • "Sheets" • "Lines"
Return Value	Array containing string object names.		

Python Syntax	GetObjectsInGroup (<group>)
Python Example	oEditor.GetObjectsInGroup ('Sheets')

VB Syntax	GetObjectsInGroup <group>

VB Example	<code>oEditor.GetObjectsInGroup "Sheets"</code>

GetObjectVolume

Returns an object's volume from its name).

UI Access	N/A		
Parameters	Name	Type	Description
	<name>	String	Object name
Return Value	.Real		

Python Syntax	<code>GetObjectVolume(<name>)</code>
Python Example	<code>oEditor.GetObjectVolume("Box1")</code>

VB Syntax	<code>GetObjectVolume <name></code>
VB Example	<code>oEditor.GetObjectVolume "Box1"</code>

GetObjPath [Editor]

Obtains the path to the 3D modeler.

UI Access	N/A
Parameters	None.
Return Value	String containing the path.

Python Syntax	GetObjPath()
Python Example	<code>oEditor.GetObjPath()</code>

VB Syntax	GetObjPath
VB Example	<code>oEditor.GetObjPath</code>

GetPartsForUserDefinedModel

Obtains parts from a specified user defined model.

UI Access	N/A		
Parameters	Name	Type	Description
	<udmName>	String	Name of user defined model.
Return Value	Array of strings containing associated parts.		

Python Syntax	GetPartsForUserDefinedModel (<udmName>)
Python Example	<code>oEditor.GetPartsForUserDefinedModel ("OnDieSpiralInductor1")</code>

VB Syntax	GetPartsForUserDefinedModel <udmName>
VB Example	<code>oEditor.GetPartsForUserDefinedModel "OnDieSpiralInductor1"</code>

GetPoints [3D Modeler Editor]

Returns all the points defined in current 3D modeler.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing names of points.

Python Syntax	GetPoints ()
Python Example	<code>oEditor.GetPoints()</code>

VB Syntax	GetPoints
VB Example	<code>oEditor.GetPoints</code>

GetPropEvaluatedValue

Returns the Evaluated-Value for Value-Property and Variable. Returns the Property-value as text string for other property types

Note:

This command is not supported by the EMIT and Circuit design types.

UI Access	N/A		
Parameters	Name	Type	Description
	<PropName>	String	Name of the property.

Return Value	String value of the evaluated value.
Python Syntax	GetPropEvaluatedValue (<PropName>)
Python Example	<pre>oVar = oDesign.GetChildObject(" Variables/var") oVar.GetPropEvaluatedValue()</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.

UI Access	N/A		
Parameters	Name	Type	Description
	<PropTab>	String	One of the following, where tab titles are shown in parentheses: - PassedParameterTab ("Parameter Values") - DefinitionParameterTab (Parameter Defaults") - LocalVariableTab ("Variables" or "Local Variables") - ProjectVariableTab ("Project variables") - ConstantsTab ("Constants")

			• BaseElementTab ("Symbol" or "Footprint") • ComponentTab ("General") • Component("Component") • CustomTab ("Intrinsic Variables") • Quantities ("Quantities") • Signals ("Signals")
	<PropServer>	String	An object identifier, generally returned from another script method, such as <code>CompInst@R;2;3</code>
	<PropName>	String	Name of the property.
Return Value	String value of the property.		

Python Syntax	<code>GetPropertyValue (<PropTab>, <PropServer>, <PropName>)</code>
Python Example	<pre> selectionArray = oEditor.GetSelections() for k in selectionArray: val = oEditor.GetPropertyValue("PassedParameterTab", k, "R") ... </pre>

VB Syntax	<code>GetPropertyValue (<PropTab>, <PropServer>, <PropName>)</code>

VB Example	<pre> selectionArray = oEditor.GetSelections for k in selectionArray: val = oEditor.GetPropertyValue("PassedParameterTab", k, "R") ... </pre>

GetPropNames [Modeler]

Returns the property names for the active model object, or specified property values.

Note:

This command is not supported by the EMIT and Circuit design types.

UI Access	N/A		
Parameters	Name	Type	Description
	<IncludeReadOnly>	Boolean	Optional. • True - includes all properties. • False - returns only property names that can be changed. True by default.
Return Value	Returns property names of the current 3D Model object.		

Python Syntax	<code>GetPropNames(<IncludeReadOnly>)</code>
Python Example	<code>oEditor.GetPropNames(True)</code>

VB Syntax	<code>GetPropNames(<IncludeReadOnly>)</code>
VB Example	<code>oEditor.GetPropName True</code>

GetPropSIValue

Returns the SI-Value for Value-Property and Variable. Return NAN for other property type if its value is not able to convert to be a double-floating point value.

Note:

This command is not supported by the EMIT and Circuit design types.

UI Access	N/A		
Parameters	Name	Type	Description
	<PropName>	String	Name of the property.
Return Value	Property value as a double floating value, or NAN if the property value cannot be converted to double floating point.		

Python Syntax	<code>GetPropSIValue (<PropName>)</code>
Python Example	<pre>oCreateBox = oDesign.GetChildObject("3D Modeler/Box1/CreateBox:1") oCreateBox.GetPropValue("xSize")</pre>

	<pre>return "length / 2" oCreateBox.GetPropEvaluatedValue ("xSize") return '0.4mm' oCreateBox.GetPropSIValue ("xSize") return 0.0004</pre>

GetPropValue [Modeler]

Returns the property value for the active model object, or specified property values.

Note:

This command is not supported by the EMIT and Circuit design types.

UI Access	N/A		
Parameters	Name	Type	Description
	<propPath>		Optional. Child object's property path. See: Object Property Function Summary .
Return Value	The property value.		

Python Syntax	GetPropValue(<propPath>)
Python Example	<pre>Rh1 = oDesign.GetChildObject('Box1') Rh1.GetPropValue('Orientation')</pre>

	Returns the specified Property Value: <code>'Global'</code>

VB Syntax	<code>GetPropValue(<propPath>)</code>
VB Example	<pre>Rh1 = oDesign.GetChildObject("Box1") GetPropValue("Orientation")</pre>

GetRelativeCoordinateSystems

Returns the relative coordinate systems in the current 3D modeler.

UI Access	N/A
Parameters	None.
Return Value	Array containing name of relative coordinate systems.

Python Syntax	<code>GetRelativeCoordinateSystems()</code>
Python Example	<code>oEditor.GetRelativeCoordinateSystems()</code>

VB Syntax	<code>GetRelativeCoordinateSystems</code>
VB Example	<code>oEditor.GetRelativeCoordinateSystems</code>

GetSelections [Model Editor]

Returns an array of currently selected objects.

UI Access	N/A
Parameters	None.
Return Value	Array containing object IDs

Python Syntax	GetSelections()
Python Example	<code>oEditor.GetSelections()</code>

VB Syntax	GetSelections
VB Example	<code>oEditor.GetSelections</code>

GetSubGroupsInGroup

Returns subgroup names in a specified group.

UI Access	N/A		
Parameters	Name	Type	Description
	<group>	String	Group name. One of: "<materialName>"

			• "<assignmentName>" • "Non Model" • "Solids" • "Unclassified" • "Sheets" • "Lines"
Return Value	Array containing subgroup names.		

Python Syntax	GetSubGroupsInGroup(<group>)
Python Example	<code>oEditor.GetSubGroupsInGroup('Sheets')</code>

VB Syntax	GetSubGroupsInGroup<group>
VB Example	<code>oEditor.GetSubGroupsInGroup "Sheets"</code>

GetUserPosition

Returns a user's current coordinates in the 3D Modeler window.

UI Access	N/A		
Parameters	Name	Type	Description
	<prompt>	String	"Enter a point." Then click a point in the 3D Modeler window.
Return Value	Array containing X, Y, Z coordinates.		

Python Syntax	<code>GetPosition(<prompt>)</code>
Python Example	<code>oEditor.GetPosition("Enter a point.")</code>

VB Syntax	<code>GetPosition <prompt></code>
VB Example	<code>oEditor.GetPosition "Enter a point."</code>

GetVertexIDFromNameForFirstOperation

Returns vertex ID associated with a specified vertex belongs to the first operation of a part.

UI Access	N/A		
Parameters	Name	Type	Description
	<PartName>	String	Name of specified part.
	<VertexName>	String	Name of specified vertex.
Return Value	Integer representing vertex ID.		

Python Syntax	<code>GetVertexIDFromNameForFirstOperation(<PartName>, <VertexName>)</code>
Python Example	<code>oEditor.GetVertexIDFromNameForFirstOperation("Part1", "Vertex1")</code>

VB Syntax	<code>GetVertexIDFromNameForFirstOperation <PartName>, <VertexName></code>
VB Example	<code>oEditor.GetVertexIDFromNameForFirstOperation "Part1", "Vertex1"</code>

GetVertexIDsFromEdge

Returns vertex IDs associated with a specified edge.

UI Access	N/A		
Parameters	Name	Type	Description
	<edgeID>	Integer	Edge ID.
Return Value	Array containing string vertex IDs.		

Python Syntax	GetVertexIDsFromEdge(<edgeID>)
Python Example	oEditor.GetVertexIDsFromEdge(10)

VB Syntax	GetVertexIDsFromEdge <edgeID>
VB Example	oEditor.GetVertexIDsFromEdge 10

GetVertexIDsFromFace

Returns vertex IDs associated with a specified face.

UI Access	N/A		
Parameters	Name	Type	Description
	<faceID>	Integer	Face ID.
Return Value	Array containing string vertex IDs.		

Python Syntax	GetVertexIDsFromFace(<faceID>)
Python Example	oEditor.GetVertexIDsFromFace(10)

VB Syntax	GetVertexIDsFromFace <faceID>
VB Example	oEditor.GetVertexIDsFromFace 10

GetVertexIDsFromObject

Returns vertex IDs associated with a specified object.

UI Access	N/A		
Parameters	Name	Type	Description
	<objectName>	String	Object name.
Return Value	Array containing string vertex IDs.		

Python Syntax	GetVertexIDsFromObject(<objectName>)
Python Example	oEditor.GetVertexIDsFromObject('Box1')

VB Syntax	GetVertexIDsFromObject <objectName>
VB Example	oEditor.GetVertexIDsFromObject "Box1"

GetVertexPosition

Returns an array of coordinates for a specified vertex.

UI Access	N/A		
Parameters	Name	Type	Description
	<vertexID>	Integer	Vertex ID.
Return Value	Array containing X, Y, Z coordinates.		

Python Syntax	GetVertexPosition(<vertexID>)
Python Example	<code>oEditor.GetVertexPosition(1)</code>

VB Syntax	GetVertexPosition <vertexID>
VB Example	<code>oEditor.GetVertexPosition 1</code>

GetWireBodyNames

Returns the wire body names in current 3D modeler.

UI Access	N/A
Parameters	None.
Return Value	Array containing string of names.

Python Syntax	GetWireBodyNames()
Python Example	<code>oEditor.GetWireBodyNames()</code>

VB Syntax	GetWireBodyNames
VB Example	<code>oEditor.GetWireBodyNames</code>

PageSetup

Specifies page settings for printing.

UI Access	File > Page Setup.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. ["NAME:PageSetupData", "margins:=", <SetupArray>]
	<SetupArray>	Array	Structured Array ["left:=", <string>, "right:=", <string>, "top:=", <string>,]

	<table><tr><td></td><td></td><td>"bottom:=", <string>)</td></tr><tr><td></td><td></td><td>]</td></tr></table>			"bottom:=", <string>)			]
		"bottom:=", <string>)					
		]					
Return Value	None.						

Python Syntax	PageSetup(<Parameters>)
Python Example	<pre>oEditor.PageSetup(["NAME:PageSetupData", "margins:=", ["left:=", "10mm", "right:=", "10mm", "top:=", "10mm", "bottom:=", "10mm"]])</pre>

VB Syntax	PageSetup <Parameters>
VB Example	<pre>oEditor.PageSetup Array("NAME:PageSetupData", "margins:=", Array("left:=", "10mm", "right:=", "10mm", "top:=", "10mm",</pre>

	<pre>"bottom:=", "10mm"))</pre>

RemoveBadEdges

Removes bad edges from specified list.

UI Access	N/A		
Parameters	Name	Type	Description
	<EdgeList>	Array	Array containing edge IDs.
Return Value	None.		

Python Syntax	RemoveBadEdges (<EdgeList>)
Python Example	<code>oEditor.RemoveBadEdges ([18, 30])</code>

VB Syntax	RemoveBadEdges <EdgeList>
VB Example	<code>oEditor.RemoveBadEdges Array(18, 30)</code>

RemoveBadFaces

Removes bad faces from specified list.

UI Access	N/A

Parameters	Name	Type	Description
	<FaceList>	Array	Array containing face IDs.
Return Value	None.		

Python Syntax	RemoveBadFaces (<FaceList>)
Python Example	<code>oEditor.RemoveBadFaces ([328, 333])</code>

VB Syntax	RemoveBadFaces <FaceList>
VB Example	<code>oEditor.RemoveBadFaces Array(328, 333)</code>

RemoveBadVertices

Removes bad vertices from specified list.

UI Access	N/A		
Parameters	Name	Type	Description
	<VertexList>	Array	Array containing vertex IDs.
Return Value	None.		

Python Syntax	RemoveBadVertices (<VertexList>)
Python Example	<code>oEditor.RemoveBadVertices ([324, 325])</code>

VB Syntax	RemoveBadVertices <VertexList>
VB Example	<code>oEditor.RemoveBadVertices Array(324, 325)</code>

RenamePart

Renames an object.

UI Access	Enter new name in Name field.		
Parameters	Name	Type	Description
	<renameParametersArray>	Array	Structured array. Array("NAME:Rename Data", "Old Name:=", <string>, "New Name:=", <string>)
Return Value	None.		

Python Syntax	<code>oEditor.RenamePart(<renameParametersArray>)</code>
Python Example	<pre>oEditor.RenamePart(['NAME:Rename Data', 'Old Name:=' , 'partname', 'New Name:=' , 'newpartname',</pre>

	<pre>]) </pre>

VB Syntax	oEditor.RenamePart <RenameParametersArray>
VB Example	<pre> oEditor.RenamePart Array("NAME:Rename Data", "Old Name:=", "partname", "New Name:=", "newpartname",) </pre>

SetPropValue [Modeler]

Sets the property value for the active model property.

Note:

This command is not supported by the EMIT and Circuit design types.

UI Access	Edit Properties on History Tree objects		
Parameters	Name	Type	Description
	<propPath>	String	Child object's property path. See: Object Property Function Summary .
	<value>	Varies	New property value.
Return Value	Boolean: • True – property found. • False – property not found.		

Python Syntax	SetPropValue(<propPath>, <value>)
Python Example	oEditor.SetPropValue("Color/r", 111)

VB Syntax	SetPropValue <propPath>, <value>
VB Example	oEditor.SetPropValue "Color/r", 111)

SetTopDownViewDirectionForActiveView

Sets active view to top-down view direction.

UI Access	N/A		
Parameters	Name	Type	Description
	<RelativeCS>	String	Name of relative coordinate system
Return Value	None.		

Python Syntax	SetTopDownViewDirectionForActiveView (<RelativeCS>)
Python Example	oEditor.SetTopDownViewDirectionForActiveView("Global")

VB Syntax	SetTopDownViewDirectionForActiveView <RelativeCS>
VB Example	oEditor.SetTopDownViewDirectionForActiveView "Global"

SetTopDownViewDirectionForAllViews

Sets all views to top-down view direction.

UI Access	N/A		
Parameters	Name	Type	Description
	< <i>RelativeCS</i> >	String	Name of relative coordinate system
Return Value	None.		

Python Syntax	SetTopDownViewDirectionForAllViews (< <i>RelativeCS</i> >)		
Python Example	<code>oEditor.SetTopDownViewDirectionForAllViews("Global")</code>		

VB Syntax	SetTopDownViewDirectionForAllViews < <i>RelativeCS</i> >		
VB Example	<code>oEditor.SetTopDownViewDirectionForAllViews "Global"</code>		

UpdatePriorityList

Updates specified priority lists.

UI Access	N/A		
Parameters	Name	Type	Description
	< <i>Lists</i> >	Array	Array of priority list names.
Return Value	None.		

Python Syntax	UpdatePriorityList (<Lists>)
Python Example	<code>oEditor.UpdatePriorityList(["PriorityList1", "PriorityList2"])</code>

VB Syntax	UpdatePriorityList <Lists>
VB Example	<code>oEditor.UpdatePriorityList Array("PriorityList1", "PriorityList2")</code>

UpgradeVersion

Upgrades legacy geometry to current version.

UI Access	Right-click on an operation icon in the history tree, select Upgrade Version .		
Parameters	Name	Type	Description
	<Selections>	Array	Structured array. <code>Array("NAME:Parameters", Array("NAME:PartOperations", Array("NAME:<string>"), "OperationIndices:=", <array of integers>), Array("NAME:UDMOperations"))</code>
Return Value	None.		

Python Syntax	UpgradeVersion (<Selections>)

Python Example	<pre>oEditor.UpgradeVersion(["NAME:Parameters", ["NAME:PartOperations", ["NAME:source", "OperationIndices:=", [0]]], ["NAME:UDMOperations"]])</pre>

VB Syntax	UpgradeVersion <Lists>
VB Example	<pre>oEditor.UpgradeVersion Array("NAME:Parameters", Array("NAME:PartOperations", Array("NAME:source", "OperationIndices:=", Array(0))), Array("NAME:UDMOperations"))</pre>

Validate3DComponent

Validates a 3D component.

UI Access	N/A		
Parameters	Name	Type	Description
	<componentPath>	String	Path to a 3D component.
	<password>	String	Optional. Password to access the component.
Return Value	Boolean: • 1 - component is valid. • 0 - component is not valid.		

Python Syntax	Validate3DComponent (<componentPath>, <password>)		
Python Example	<pre>oEditor.Validate3DComponent ("C:/temp/component.3dcomp", "")</pre>		

VB Syntax	Validate3DComponent <componentPath>, <password>		
VB Example	<pre>oEditor.Validate3DComponent "C:/temp/component.3dcomp", ""</pre>		

WriteHistoryTreeLayoutForTest

Writes history tree layout to a file.

UI Access	N/A		
Parameters	Name	Type	Description
	<filePath>	String	Path to the file.

	<OrgByMaterial>	Integer	• 1 - organize objects by material. • 0 - do not organize by material.
	<OrgByAssignment>	Integer	• 1 - organize sheets by assignment. • 0 - do not organize by assignment.
	<OrgByCompDefinition>	Integer	• 1 - organize components by definition. • 0 - do not organize by definition.
	<DonotOrgUnderGroup>	Integer	• 1 - do not organize within group. • 0 - organize within group.
	<ShowGroup>	Integer	Optional. • 1 - show group. • 0 - do not show.
Return Value	None.		

Python Syntax	WriteHistoryTreeLayoutForTest (<filePath>, <OrgByMaterial>, <OrgByAssignment>, <OrgByCompDefinition>, <DonotOrgUnderGroup>, <ShowGroup>)
Python Example	<pre>oEditor.WriteHistoryTreeLayoutForTest ('C:\temp', 1, 0, 0, 0, 1)</pre>

VB Syntax	WriteHistoryTreeLayoutForTest <filePath>, <OrgByMaterial>, <OrgByAssignment>, <OrgByCompDefinition>, <DonotOrgUnderGroup>, <ShowGroup>
VB Example	<pre>oEditor.WriteHistoryTreeLayoutForTest "C:\temp", 1, 0, 0, 0, 1</pre>

Cable Modeling Commands

Cable Modeling is a Beta Feature to 3D Component Modeling for Release 21.2. Cables are created and edited as design objects using the Cable Setup module, and then inserted as native components. For example, elements of a cable are created:

```
oModule = oDesign.GetModule("CableSetup")  
oModule.CreateStraightWireCable(...
```

And then, when a Harness has been created from various components, it is inserted to the Modeler:

```
oEditor = oDesign.SetActiveEditor("3D Modeler")  
oEditor.InsertNativeComponent(  
    [  
        "NAME:InsertNativeComponentData",  
        "TargetCS:=" , "Global",  
        "SubmodelDefinitionName:=", "MyHarness",  
    ...
```

[AddCableToBundle](#)

[CreateCableBundle](#)

[CreateCableHarness](#)

[CreateClockSource](#)

[CreatePWLSource](#)

[CreateStraightWireCable](#)

[CreateTwistedPairCable](#)[ExportCableLibrary](#)[ImportCableLibrary](#)[RemoveCable](#)[UpdateCableHarness](#)

AddCableToBundle

Add a cable to an existing bundle using the Beta Cable Modeling feature. Right click on the 3D Component icon and select **Cables>Edit Cables** to open the **Cable Editor**.

UI Access	3D Components>Cables > Edit Cables
Parameters	None
Return Value	None

Python Syntax	AddCableToBundle(<parameters>))
Python Example	<pre>oModule.AddCableToBundle("bundle1", "tpwire2", 1, ["NAME:CableInstParams", "XPos:=" , "0mm", "YPos:=" , "0mm", "RotX:=" , "0deg"</pre>

	<pre>], ["NAME:CableInstAttribs", "Name:=", "tpwire2"])</pre>

VB Syntax	AddCableToBundle <name> Array(<parameters>)
VB Example	<pre>oModule.AddCableToBundle "bundle2", "bundle1", 1, Array("NAME:CableInstParams", "XPos:=", _ "0mm", "YPos:=", "0mm", "RotX:=", "0deg"), Array("NAME:CableInstAttribs", "Name:=", _ "bundle1")</pre>

CreateCableBundle

Creates a cable bundle using the Beta Cable Modeling feature. Right click on the 3D Component icon and select **Cables>Edit Cables** to open the **Cable Editor**. Then **Add>Bundle...**

UI Access	3D Components>Cables > Edit Cables
Parameters	None
Return Value	None

Python Syntax	CreateCableBundle([<parameters>]), [<name>])
Python Example	<pre> oModule.CreateCableBundle (["NAME:BundleParams", "AutoPack:=" , True, ["NAME:InsulationJacketParams", "InsThickness:=" , "0.25mm", "JacketMaterial:=" , "PVC plastic", "InnerDiameter:=" , "2.5mm"]], ["NAME:BundleAttribs", "Name:=" , "bundle1"]) </pre>

VB Syntax	CreateCableBundle Array(<parameters> , <name>)
VB Example	<pre> oModule.CreateCableBundle Array("NAME:BundleParams", "AutoPack:=", true, Array("NAME:In- sulationJacketParams", "InsThickness:=", _ </pre>

	"0.25mm", "JacketMaterial:=", "PVC plastic", "InnerDiameter:=", "2.5mm")), Array("NAME:BundleAttribs", "Name:=", _ "bundle3")

CreateCableHarness

Creates a cable harness using the Beta Cable Modeling feature. Right click on the 3D Component icon and select **Cables>New Cable Harness** to open the **Insert Cable Harness Component Editor**.

UI Access	3D Components>Cables > New Cable Harness...
Parameters	None
Return Value	None

Python Syntax	CreateCableHarness(<parameters>))
Python Example	<pre>oModule.CreateCableHarness(["NAME:HarnessParams", "Bundle:=" , "bundle1", "TwistAlongPath:=" , "720deg", "Route:=" , "Polyline1", "AutoOrient:=" , False, "Origin:=" , ["0mm", "0mm", "0mm"],</pre>

```

        "XAxisEnd:="                , [ "-6mm", "0mm", "0mm" ],
        "PlaneFlip:="                , True,
        [
            "NAME:RefConductors",
            "tpwire2/w1"
        ],
        [
            "NAME:InputTerminations",
            "tpwire2/w1:="                , [ "Source:="
            "tpwire2/w2:="                , [ "Imped:="
        ],
        [
            "NAME:OutputTerminations",
            "tpwire2/w1:="                , [ "Imped:="
            "tpwire2/w2:="                , [ "Source:="
        ]
    ],
    [
        "NAME:HarnessAttribs",
        "Name:="                , "MyHarness"
    ]
)

```


VB Syntax	CreateCableHarness Array(<parameters>)
VB Example	<pre>oModule.CreateCableHarness Array("NAME:HarnessParams", "Bundle:=", "bundle1", "TwistAlongPath:=", _ "360deg", "Route:=", "Polyline2", "AutoOrient:=", false, "Origin:=", Array(_ "-24mm", "24mm", "0mm"), "XAxisEnd:=", Array("-30mm", "24mm", "0mm"), "PlaneFlip:=", _ true, Array("NAME:RefConductors", "tpwire2/w1"), Array("NAME:InputTerminations", "tpwire2/w1:=", Array("Source:=", _ "5V", "Imped:=", "50ohm"), "tpwire2/w2:=", Array("Source:=", "5V", "Imped:=", _ "50ohm")), Array("NAME:OutputTerminations", "tpwire2/w1:=", Array("Source:=", "5V", "Imped:=", _ "50ohm"), "tpwire2/w2:=", Array("Imped:=", "50ohm"))), Array("NAME:Har- nessAttribs", "Name:=", _ "MyNewHarness")</pre>

CreateClockSource

Creates a clock source using the Beta Cable Modeling feature. Right click on the 3D Component icon and select **Cables>Edit Cable Sources...** to open the **Cable Time Domain Sources Editor**.

UI Access	3D Components>Cables > Edit Cables
Parameters	None
Return Value	None

Python Syntax	CreateClockSource(<parameters>))
Python Example	<pre> oModule.CreateClockSource(["NAME:ClockSignalParams", "Period:=" , "35us", "LowPulseVal:=" , "0V", "HighPulseVal:=" , "1V", "Risetime:=" , "5us", "Falltime:=" , "5us", "PulseWidth:=" , "20us"], ["NAME:TDSOURCEAttribs", "Name:=" , "clock1"]) </pre>

VB Syntax	CreateClockSourceArray(<parameters>)
VB Example	<pre> oModule.CreateClockSource Array("NAME:ClockSignalParams", "Period:=", "35us", "LowPulseVal:=", _ </pre>

	<pre>"0V", "HighPulseVal:=", "1V", "Risetime:=", "5us", "Falltime:=", "5us", "PulseWidth:=", _ "20us"), Array("NAME:TDSourceAttribs", "Name:=", "clock2")</pre>

CreatePWLSource

Creates a PWL source using the Beta Cable Modeling feature. Right click on the 3D Component icon and select **Cables > Edit Cable Sources...** to open the **Cable Time Domain Sources Editor**.

UI Access	3D Components>Cables > Edit Cable Sources...
Parameters	None
Return Value	None

Python Syntax	CreatePWLSource(<parameters>))
Python Example	<pre>oModule.CreatePWLSource(["NAME:PWLSignalParams", ["NAME:SignalValues", "0V", "0.5V", "0V"</pre>

	<pre>], ["NAME:TimeValues", "0ns", "1ns", "2ns"]], ["NAME:TDSOURCEATTRIBS", "Name:=" , "pw13"]) </pre>

VB Syntax	CreatePWLSourceArray(<parameters>)
VB Example	<pre> oModule.CreatePWLSource Array("NAME:PWLSignalParams", Array("NAME:SignalValues", "0V", _ "0.5V", "0V"), Array("NAME:TimeValues", "0ns", "1ns", "2ns")), Array ("NAME:TDSOURCEATTRIBS", "Name:=", _ "pw11") </pre>

CreateStraightWireCable

Creates a straight Wire Cable using the Beta Cable Modeling feature. Right click on the 3D Component icon and select **Cables > Edit Cables** to open the **Cable Editor**. Then **Add Straight Wire Cable**.

UI Access	3D Components>Cables > Edit Cables
Parameters	None
Return Value	None

Python Syntax	CreateStraightWireCable([<parameters>]), [<name>])
Python Example	<pre>oModule = oDesign.GetModule("CableSetup") oModule.CreateStraightWireCable(["NAME:StWireParams", "WireStandard:=" , "ISO", "WireGauge:=" , "0.13", "CondDiameter:=" , "0.55mm", "CondMaterial:=" , "copper", "InsThickness:=" , "0.25mm", "InsMaterial:=" , "PVC plastic", "InsType:=" , "Thin Wall"</pre>

	<pre>], ["NAME:StWireAttribs", "Name:=" , "stwire1"]) </pre>

VB Syntax	Paste
VB Example	<pre> oModule.CreateStraightWireCable Array("NAME:StWireParams", "WireStandard:=", "ISO", "WireGauge:=", _ "0.13", "CondDiameter:=", "0.55mm", "CondMaterial:=", "copper", "InsThickness:=", _ "0.25mm", "InsMaterial:=", "PVC plastic", "InsType:=", "Thin Wall"), Array ("NAME:StWireAttribs", _ "Name:=", "stwire3") </pre>

CreateTwistedPairCable

Creates a twisted pair cable using the Beta Cable Modeling feature. Right click on the 3D Component icon and select **Cables>Edit Cables** to open the **Cable Editor**. Then **Add Twisted Pair Cable**.

UI Access	3D Components>Cables > Edit Cables
Parameters	None
Return Value	None

Python Syntax	CreateTwisted Pair Cable([<parameters>]), [<name>])
Python Example	<pre>oModule.CreateTwistedPairCable("stwire1", ["NAME:TwistedPairParams", ["NAME:VirtualJacketParams", "JacketMaterial:=" , "", "InnerDiameter:=" , "0"], "IsLayLengthSpecified:=", False, "LayLength:=" , "13.8888888888889mm", "TurnsPerMeter:=" , "72"], ["NAME:TwistedPairAttribs", "Name:=" , "tpwire1"]])</pre>

VB Syntax	CreateTwistedPairCable "<name>" , Array(<parameters>)

VB Example	<pre>oModule.CreateTwistedPairCable "stwire1", Array("NAME:TwistedPairParams", _ Array("NAME:VirtualJacketParams", "JacketMaterial:=", _ "", "InnerDiameter:=", "0"), "IsLayLengthSpecified:=", false, "LayLength:=", _ "13.8888888888889mm", "TurnsPerMeter:=", "72"), Array("NAME:TwistedPairAttribs", "Name:=", _ "tpwire3")</pre>

ExportCableLibrary

Exports a cable library created using the Beta Cable Modeling feature. Right click on the 3D Component icon and select **Cables > Export Cable Library**. to open a browser window.

UI Access	3D Components > Cables > Export Cable Library...
Parameters	None
Return Value	None

Python Syntax	ExportCableLibrary(<parameters>)
Python Example	<pre>oModule = oDesign.GetModule("CableSetup") oModule.ExportCableLibrary("D:\\Users\\JaneDoe\\PersonalLib\\MyCables")</pre>

VB Syntax	ExportCableLibrary(<parameters>)
VB Example	<pre>oModule = oDesign.GetModule("CableSetup") oModule.ExportCableLibrary("D:\\Users\\JaneDoe\\PersonalLib\\MyCables")</pre>

ImportCableLibrary

Imports an existing cable library created using the Beta Cable Modeling feature. Right click on the 3D Component icon and select **Cables>Export Cable Library**. to open a browser window.

UI Access	3D Components>Cables > Import Cable Library...
Parameters	None
Return Value	None

Python Syntax	ImportCableLibrary(<parameters>))
Python Example	<pre>oModule = oDesign.GetModule("CableSetup") oModule.ExportCableLibrary("D:\\Users\\JaneDoe\\PersonalLib\\MyCables")</pre>

VB Syntax	ImportCableLibrary(<parameters>)
VB Example	<pre>oModule = oDesign.GetModule("CableSetup") oModule.ExportCableLibrary("D:\\Users\\JaneDoe\\PersonalLib\\MyCables")</pre>

RemoveCable

Removes an existing cable created using the Beta Cable Modeling feature. Right click on the 3D Component icon and select **Cables>Edit Cables....** to open a **Cable Editor** window.

UI Access	3D Components>Cables > Edit Cables...
Parameters	None
Return Value	None

Python Syntax	<code>RemoveCable(<parameters>))</code>
Python Example	<pre>oModule = oDesign.GetModule("CableSetup") oModule.RemoveCable("bundle4")</pre>

VB Syntax	<code>ImportCableLibrary(<parameters>)</code>
VB Example	<pre>oModule = oDesign.GetModule("CableSetup") oModule.RemoveCable("bundle4")</pre>

UpdateCableHarness

Creates a cable harness using the Beta Cable Modeling feature. Right click on the 3D Component icon and select **Cables>New Cable Harness** to open the **Insert Cable Harness Component Editor**.

UI Access	3D Components>Cables > New Cable Harness...
Parameters	None
Return Value	None

Python Syntax	UpdateCableHarness(<parameters>))
Python Example	<pre>oModule.UpdateCableHarness(["NAME:HarnessParams", "Bundle:=" , "bundle1", "TwistAlongPath:=" , "720deg", "Route:=" , "Polyline1", "AutoOrient:=" , False, "Origin:=" , ["0mm", "0mm", "0mm"], "XAxisEnd:=" , ["-6mm", "0mm", "0mm"], "PlaneFlip:=" , True, ["NAME:RefConductors", "tpwire2/w1"], ["NAME:InputTerminations", "tpwire2/w1:=" , ["Source:=", "tpwire2/w2:=" , ["Imped:="</pre>

	<pre> ["NAME:OutputTerminations", "tpwire2/w1:=" , ["Imped:=" "tpwire2/w2:=" , ["Source:="]], ["NAME:HarnessAttribs", "Name:=" , "MyHarness"]) </pre>

VB Syntax	UpdateCableHarness Array(<parameters>)
VB Example	<pre> oModule.UpdateCableHarness Array("NAME:HarnessParams", "Bundle:=", "bundle1", "TwistAlongPath:=", _ "360deg", "Route:=", "Polyline2", "AutoOrient:=", false, "Origin:=", Array(_ "-24mm", "24mm", "0mm"), "XAxisEnd:=", Array("-30mm", "24mm", "0mm"), "PlaneFlip:=", _ true, Array("NAME:RefConductors", "tpwire2/w1"), Array("NAME:InputTerminations", "tpwire2/w1:=", Array("Source:=", _ "5V", "Imped:=", "50ohm"), "tpwire2/w2:=", Array("Source:=", "5V", "Imped:=", _ "50ohm")), Array("NAME:OutputTerminations", "tpwire2/w1:=", Array("Source:=", "5V", "Imped:=", _ </pre>

	<pre>"50ohm"), "tpwire2/w2:=", Array("Imped:=", "50ohm"))), Array("NAME:Har- nessAttribs", "Name:=", _ "MyNewHarness")</pre>

9 - 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:

[DeleteOutputVariable](#)

[DoesOutputVariableExist](#)

[EditOutputVariable](#)

[ExportOutputVariables](#)

[GetOutputVariableValue](#)

[GetOutputVariables](#)

[ImportOutputVariables](#)

CreateOutputVariable

Adds a new output variable. Different forms of this command are executed for different design types.

Note:

Output variables are associated with a name and an expression. The name is not permitted to collide with design variable, sim value, or other output variable names. It cannot have spaces or any arithmetic or other operators in it. Expression definitions cannot be cyclic. For example, $A = 2*B$, $B=3*A$ is not allowed.

UI Access	HFSS 3D Layout > Results > Output Variables.		
Parameters	Name	Type	Description
	<OutputVarName>	String	Name of the new output variable.
	<Expression>	Value	Value to assign to the variable.
	<SolutionName>	String	Name of the solution, as seen in the output variable UI.
	<reportType> or <solutionType>	String	The name of the report type as seen in the output variable UI. For Layout Editor, use the solution type.
	<ContextArray> or <DomainArray>	Array	Structured array containing context for which the output variable expression is being evaluated. Array("Context:=", <string>) For Layout Editor, use Domain array: Array("Domain:=", <string>)
Return Value	None.		

Python Syntax	CreateOutputVariable (<OutputVarName>, <Expression>, <SolutionName>, <reportType solutionType>, <contextArray domainArray>)
Python Example	HFSS Example: oModule.CreateOutputVariable("Var" & OutputQuantity, OutputQuantity, Solution, "Far Fields", ["Context:=", InfiniteSphere, "Domain:=", "Sweep"])

	Layout Editor Example : <pre>Set oModule = oDesign.GetModule("OutputVariable") oModule.CreateOutputVariable("test","mag(S(WavePort1,WavePort1))", "Setup1 : LastAdaptive", "Modal Solution Data", ["Domain:=", "Sweep"])</pre>

VB Syntax	CreateOutputVariable <OutputVarName>, <Expression>, <SolutionName>, <reportType solutionType>, <contextArray domainArray>
VB Example	HFSS Example: <pre>oModule.CreateOutputVariable "Var_" & OutputQuantity, OutputQuantity, Solution, "Far Fields", Array("Context:=", InfiniteSphere, "Domain:=", "Sweep")</pre> Layout Editor Example : <pre>Set oModule = oDesign.GetModule("OutputVariable") oModule.CreateOutputVariable "test","mag(S(WavePort1,WavePort1))", "Setup1 : LastAdaptive", "Modal Solution Data", Array("Domain:=", "Sweep")</pre>

DeleteOutputVariable

Deletes an existing output variable. The variable can only be deleted if it is not in use by any traces.

UI Access	HFSS 3D Layout > Results > Output Variables. In the Output Variables window, click Delete .		
Parameters	Name	Type	Description
	<OutputVarName>	String	Name of the output variable.

Return Value	None.

Python Syntax	DeleteOutputVariable (<OutputVarName>)
Python Example	<pre>oModule = oDesign.GetModule("OutputVariable") oModule.DeleteOutputVariable ("testNew")</pre>

VB Syntax	DeleteOutputVariable <OutputVarName>
VB Example	<pre>Set oModule = oDesign.GetModule("OutputVariable") oModule.DeleteOutputVariable "testNew"</pre>

DoesOutputVariableExist

Verifies whether or not a named output variable exists.

UI Access	N/A		
Parameters	Name	Type	Description
	<outputVariableName>	String	The output variable name.
Return Value	Boolean True if the variable exists; False if it does not.		

Python Syntax	DoesOutputVariableExist(<outputVariableName>)

Python Example	<pre>oProject = oDesktop.GetActiveProject() oDesign = oProject.GetActiveDesign() oModule = oDesign.GetModule("OutputVariable") oModule.DoesOutputVariableExist("MyTestVar")</pre>

VB Syntax	DoesOutputVariableExist(<outputVariableName>)
VB Example	<pre>Set oModule = oDesign.GetModule "OutputVariable" oModule.DoesOutputVariableExist "MyTestVar"</pre>

EditOutputVariable

Changes the name or expression of an existing output variable.

UI Access	N/A		
Parameters	Name	Type	Description
	<OrigVarName>	String	Name of the original output variable.
	<NewExpression>	String	New value to assign to the variable.
	<NewVarName>	String	New name of the variable if any, else pass empty string.
	<SolutionName>	String	Name of the solution as seen in the output variable UI. For example, "Setup1 : Last Adaptive".
	<ReportType> or <SolutionType>	String	The name of the report type as seen in the output variable UI. For Layout Editor, use the solution type.
	<ContextArray> or <DomainArray>	Array	Structured array containing context for which the out-

			put variable expression is being evaluated. <code>Array("Context:=", <string>)</code> For Layout Editor, use Domain array: <code>Array("Domain:=", <string>)</code>
Return Value	None		

Python Syntax	<code>EditOutputVariable (<OrigVarName>, <NewExpression>, <NewVarName>, <SolutionName>, <ReportType SolutionType>, <ContextArray DomainArray>)</code>
Python Example	<pre>oModule = oDesign.GetModule("OutputVariable") oModule.EditOutputVariable ("test", "normalize(R1_0.V)", "testNew", "TR", "Standard", [])</pre>

VB Syntax	<code>EditOutputVariable <OrigVarName>, <NewExpression>, <NewVarName>, <SolutionName>, <ReportType SolutionType>, <ContextArray DomainArray></code>
VB Example	<pre>Set oModule = oDesign.GetModule("OutputVariable") oModule.EditOutputVariable "test", "dB(S(WavePort1,WavePort1)) ", "testNew", "Setup1 : LastAdaptive", "Modal Solution Data", Array("Domain:=", "Sweep")</pre>

ExportOutputVariables

Exports output variables to a file.

UI Access	Click on Export in the Output Variables dialog.		
Parameters	Name	Type	Description
	<FileName>	String	Name of the file include path.
Return Value	Boolean: • True - output variable successfully exported. • False - error when export output variables.		

Python Syntax	ExportOutputVariables(<FileName>)
Python Example	oModule.ExportOutputVariables("C:/output_var.aoutvar")

VB Syntax	ExportOutputVariables <FileName>
VB Example	oModule.ExportOutputVariables "C:/output_var.aoutvar"

GetOutputVariables

Returns the list of output variables.

UI Access	N/A
Parameters	None.
Return Value	Array containing all output variables.

Python Syntax	GetOutputVariables()
Python Example	<code>oDesign.GetOutputVariables()</code>

VB Syntax	GetOutputVariables
VB Example	<code>oDesign.GetOutputVariables</code>

GetOutputVariableValue

Returns the double value of an output variable. Only expressions that return a double value are supported. The expression is evaluated only for a single point.

UI Access	N/A		
Parameters	Name	Type	Description
	<OutputVarName>	String	Name of the output variable.
	<IntrinsicVariation>	String	A set of intrinsic variable value pairs to use when evaluating the output expression. HFSS example: "Freq='20GHz' Theta='20deg' Phi='30deg'"
	<SolutionName>	String	Name of the solution as listed in the output variable UI. For example, "Setup1 : Last Adaptive".
	<ReportType>	String	The name of the report type as seen in the output variable UI. Possible values in Layout/Schematic Editor are: • "Standard" • "Load Pull"

			• "Constellation" • "Data Table" • "Eye Diagram" • "Statistical"
	<ContextArray>	Array	Structured array containing context for which the output variable expression is being evaluated. Can be empty. Array("Context:=", <string>)
Return Value	Double value of the output variable.		

Python Syntax	GetOutputVariableValue(<OutputVarName>, <IntrinsicVariation>, <SolutionName>, <ReportTypeName>, <ContextArray>)
Python Example	<pre>Val = oDesign.GetOutputVariableValue("test","Freq = '20Ghz' Theta='20deg' Phi='30deg'", "TR", "Standard", [])</pre>

VB Syntax	GetOutputVariableValue <OutputVarName>, <IntrinsicVariation>, <SolutionName>, <ReportTypeName>, <ContextArray>
VB Example	<pre>Val = oDesign.GetOutputVariableValue "test","Freq = '20Ghz' Theta='20deg' Phi='30deg'", "TR", "Standard", Array()</pre>

ImportOutputVariables

Imports output variables from a file.

UI Access	Click on Import in the Output Variables dialog.		
Parameters	Name	Type	Description
	<FileName>	String	Name of the file include path.
Return Value	Boolean: • True - output variable successfully imported. • False - error when import output variables.		

Python Syntax	ImportOutputVariables(<FileName>)
Python Example	oModule.ImportOutputVariables("C:/output_var.aoutvar")

VB Syntax	ImportOutputVariables <FileName>
VB Example	oModule.ImportOutputVariables "C:/output_var.aoutvar"

SimValueContext

SimValueContext holds context information for a trace, and describes how data for a trace should be extracted from the simulation. SimValueContext contains a list of 14 required initial values:

```
SimValueContext (  
Domain ID, Calculation Type, Number of Cycles, Rise Time,  
Step, Impulse, Context ID, Window Width,
```

Window Type, TDR Kaiser Parameter, Hold Time, DeviceName,
TDR Step Time, DR Maximum Time)

For example, the following indicates a trace in the Time Domain, Standard Calculation with the number of cycles being 2:

```
"SimValueContext:=", Array(1, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0)
```

Additional, context-specific values may follow the required values, as described in subsection 15 below.

1. Domain ID

No Domain	0
Time Domain	1
Spectrum Domain	2
Sweep Domain	3
Device Domain	4
SinglePt Domain	5
LoadPull Domain	6
Transient Domain	7
Budget Domain	8
NetworkFunction Domain	9
Oscillator Domain	55802
Noise Domain	55803
Transfer Function Domain	55807
Time Frequency Domain	55808
Transient Time Domain	55809
Periodic AC Domain	55818

UI Domain	55819
Eye Measurement Domain	55823
Initial Response Domain	55824
Peak Distortion Domain	55825

2. Calculation Type

Standard Calculation	0
Device2_DCIV	1
Device3_DCIV_Output	2
Device3_DCIV_Input	3
Device3_DCIV_Transfer	4
Device3_DCIV_Reverse	5
Device2_ACLoad	6
Device3_ACLoad_Output	7
Device3_ACLoad_Input	8
Device3_ACLoad_Transfer	9
Device3_ACLoad_Reverse	10
Constellation	11
EyeDiagram	12
FreeX (Statistic Report)	13

3. Number of Cycles — Used in Time Domain in HarmonicBalance analysis.

4. **Rise Time** — Not used by Designer/Nexxim.
5. **Step** — Not used by Designer/Nexxim.
6. **Impulse** — Not used by Designer/Nexxim.
7. **Context ID** — Not used by Designer/Nexxim.
8. **Window Width** — Not used by Designer/Nexxim.
9. **Window Type** — Not used by Designer/Nexxim.
10. **TDR Kaiser Parameter** — Not used by Designer/Nexxim.
11. **Hold Time** — Not used by Designer/Nexxim.
12. **DeviceName** — Not used by Designer/Nexxim.
13. **TDR Step Time** — Not used by Designer/Nexxim.
14. **TDR Maximum Time** — Not used by Designer/Nexxim.
15. **Context-specific values** — Used in Time Domain in HarmonicBalance analysis.

Context-specific values are entered in the format "key, true/false, keyvalue", where:

- **"key"** is the name of the key being set.
- **"true/false"** indicates whether the key is a string value.
- **"keyvalue"** is the value of the key.
- The order of the context keys is not significant.
- Context keys have software defaults that will be used if not provided in the script.

VB Example:

```
"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, "10ns",  
"WM", false, "10ns",  
"WN", false, "0ns",  
"WS", false, "0ns"))
```

a. Plotting Range for Time domain in Transient and QuickEye analysis:

Description	Key Name	Is a string?	Key Value
Start Time	WS	False	0ns
Stop Time	WE	False	10ns
Minimum Time	WM	False	0ns
Maximum Time	WN	False	10ns
Is Thinning Enabled?	DE	False	0
Dy/dx Tolerance	DT	False	0.001
Number of points	DP	False	20000000

b. Transient report context for Spectral domain in Transient analysis:

Description	Key Name	Is a string?	Key Value
Start Time	TS	False	0ns
Stop Time	TE	False	10ns
Max Harmonics	MH	False	100
Max Frequency	MF	False	*
Window type	WT	False	0
Width Percentage	WW	False	100
Kaiser Parameter	KP	False	0
Adjust Coherent Gain	CG	False	0

* Script can specify either MH or MF. If neither is specified, Max Harmonics is set to 100. If both are specified, MF is used.

Window Type	ID
Rectangular	0
Bartlett	1
Blackman	2
Hamming	3
Hanning	4
Kaiser	5
Welch	6
Weber	7
Lanzcos	8

c. Eyeprobe index context for UI domain, Time domain, Eye Measurement domain in VeriEye and QuickEye analysis:

Description	Key Name	Is a string?	Key Value
Eyeprobe compinst ID	PCID	False	0

d. Eyesource index context for Initial Response domain and Peak Distortion domain in VerifEye and QuickEye analysis:

Description	Key Name	Is a string?	Key Value
Eyesource compinst ID	SCID	False	0

e. UI domain context in VerifEye and QuickEye analysis:

Description	Key Name	Is a string?	Key Value
Use midpoint?	MIDPOINT	False	0 - Don't use midpoint. 1 - Use midpoint of amplitude. 2 - Use midpoint of UI.
Minimum latch overdrive	MLO	False	0

f. Distribution Context for UI Domain in VerifEye and QuickEye analysis:

Description	Key Name	Is a string?	Key Value
Use distribution?	USE_DIST	False	0 - No 1 - Yes
Distribution type	DIST	False	0 - Receiver Jitter 1 - Receiver Noise 2 - User Defined

Receiver Jitter Parameters

Description	Key Name	Is a string?	Key Value
DLL standard deviation	DSD	False	0
Distribution type	DIST	False	0
DLL taps	DMN	False	0
Static Offset	SOFF	False	0
Number of Gaussian data sets	NUMG	False	0
Gaussian std deviation	GS0,GS1...	False	0
Offset mean	GM1,GM1...	False	0
Number of Uniform data sets	NUMU	False	0
Uniform width	UW0,UW1...	False	0
Uniform mean	UM1,UM1...	False	0

Receiver Noise Parameters

Description	Key Name	Is a string?	Key Value
Number of Gaussian data sets	NUMG	False	0
Gaussian std deviation	GS0,GS1...	False	0
Number of Uniform data sets	NUMU	False	0
Uniform width	UW0,UW1...	False	0

User Defined Parameters

Description	Key Name	Is a string?	Key Value
Number of XY data pairs	NUMG	False	0
X data	X0,X1,X2...	False	0
Y data	Y0,Y1,Y2...	False	0
Cutoff probability	CP	False	0

10 - Reporter Editor Script Commands

Reporter commands should be executed by the oDesign object.

For example:

```
Set oDesign = Project.SetActiveDesign("EMDesign1")
```

```
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:

[AddAllEyeMeasurements](#)

[AddCartesianLimitLine](#)

[AddCartesianLimitLineFromCurve](#)

[AddCartesianLimitLineFromEquation](#)

[AddCartesianXMarker](#)

[AddCartesianYMarker](#)

[AddCartesianYMarkerToStack](#)

[AddDeltaMarker](#)

[AddMarker](#)

[AddNote](#)

[AddTraceCharacteristics](#)

[AddTraces](#)

AddVerifEyeAnalysis

[ApplyReportTemplate](#)

[ChangeProperty](#)

[ClearAllMarkers](#)

[ClearAllTraceCharacteristics](#)

[CloneReportsFromDatasetSolution](#)

[CopyPlotSettings](#)

[CopyReportDefinitions](#)

[CopyReportData](#)

[CopyTraceDefinitions](#)

[CopyTracesData](#)

CreateReportFromFile

[CreateReportFromTemplate](#)

[CreateReportOfAllQuantities](#)

[DeleteAllReports](#)

[DeleteReports](#)

[DeleteTraceCharacteristics](#)

[DeleteTraces](#)

[DoesSupportTraceCharacteristics](#)

[DumpAllReportsData](#)

[EditCartesianXMarker](#)

[EditCartesianYMarker](#)

[EditMarker](#)

EditQuickEyeAnalysis

EditVerifEyeAnalysis

[ExportEyeMaskViolation](#)

[ExportImageToFile](#)

[ExportReportDataToFile](#)

[ExportTableToFile](#)

[ExportToFile \[Reporter\]](#)

[GetAllCategories](#)

[GetAllQuantities](#)

[GetAllReportNames](#)

[GetAvailableDisplayTypes](#)

[GetAvailableReportTypes](#)

[GetAvailableSolutions](#)

[GetChildNames](#)

[GetChildObject](#)

[GetChildTypes](#)

[GetCurvePropServerName](#)

[GetDisplayType](#)

[GetDynLinkIntrinsicVariables](#)

[GetDynLinkQtyValueState](#)

[GetDynLinkLinkTraces](#)

[GetDynLinkVariableValues](#)

[GetName](#)

[GetObjPath](#)

[GetPropNames](#)

[GetPropSIValue](#)

[GetPropertyValue](#)

[GetQtyExpressionsForSourceTrace](#)

[GetReportSummaryForRegressionTesting](#)

[GetReportTraceNames](#)

[GetSolutionContexts](#)

[GetSolutionDataPerVariation](#)

[GroupPlotCurvesByGroupingStrategy](#)

[ImportIntoReport](#)

[ImportReportDataIntoReport](#)

[MovePlotCurvesToGroup](#)

[MovePlotCurvesToNewGroup](#)

[OpenWindowForAllReports](#)

[OpenWindowForReports](#)

[PastePlotSettings](#)

[PasteReports](#)

[PasteReportsWithLegacyNames](#)

[PasteTraces](#)

[PasteTracesWithLegacyNames](#)

[RenameReport](#)

[RenameTrace](#)

[ResetPlotSettings](#)

[SavePlotSettingsAsDefault](#)

[SetLinkOutputTraces](#)

[SetPropValue](#)

[UngroupPlotCurvesInGroup](#)

[UpdateAllFieldsPlots](#)

[UpdateAllReports](#)

[UpdateReports](#)

[UpdateTraces](#)

[UpdateTracesContextandSweeps](#)

AddAllEyeMeasurements

Displays all the eye measurements in tabular format

UI Access	Right-click the report and select Trace Characteristics > Add All Eye Measurements		
Parameters	Name	Type	Description
	<ReportName>	String	Name of the report.
Return Value	None.		

Python Syntax	AddAllEyeMeasurements(<ReportName>)
Python Example	<code>oModule.AddAllEyeMeasurements("DQS Eye")</code>

VB Syntax	AddAllEyeMeasurements <ReportName>
VB Example	<code>oModule.AddAllEyeMeasurements "DQS Eye"</code>

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.
	<Def>	Array	Structured array:

			<pre> Array("NAME:CartesianLimitLine", Array("NAME:XValues", <integer X values>), "XUnits:=", <string unit of measure for X>, Array("NAME:YValues", <integer Y values>), "YUnits:=", "<string unit of measure for Y>", "YAxis:=", <string name of associated Y axis>) </pre>
Return Value	None.		

Python Syntax	AddCartesianLimitLine (<ReportName>, <Def>)		
Python Example	<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>		

VB Syntax	AddCartesianLimitLine <ReportName>, <Def>		

VB Example	<pre>oModule.AddCartesianLimitLine "Project Outputs", 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")</pre>

AddCartesianLimitLineFromCurve

Adds a limit line to a report from selected curve on the plot.

UI Access	Report2D > Add Limit Line > From Selected Curve...		
Parameters	Name	Type	Description
	<ReportName>	String	Name of the report.
	<LimitLineParams>	String	Structured array.
			<pre>Array("NAME:CartesianLimitLineFromCurve", "TraceName:=", <string name of selected trace>, "CurveName:=", <string name of selected curve>, "Start:=", "<value><unit>", "Stop:=", "<value><unit>", "YAxis:=", <integer Y-Axis number>, "YOffset:=", <value offset from Y-Axis>,</pre>

			"CreateMode:=", "<AboveCurve BelowCurve Above and Below Curve>", "YShiftPercent:=", "<value percentage to shift from Y>"
Return Value	None.		

Python Syntax	AddCartesianLimitLineFromCurve(<ReportName>, <LimitLineParams>)
Python Example	<pre> oModule.AddCartesianLimitLineFromCurve("Variables Plot 2", ["NAME:CartesianLimitLineFromCurve", "TraceName:=" , "Phase", "CurveName:=" , "", "Start:=" , "0deg", "Stop:=" , "375deg", "YAxis:=" , 1, "YOffset:=" , 0, "CreateMode:=" , "AboveCurve", "YShiftPercent:=" , 10]) </pre>

VB Syntax	AddCartesianLimitLineFromCurve<ReportName>, <LimitLineParams>

VB Example	<pre>oModule.AddCartesianLimitLineFromCurve _ "Variables Plot 2", _ Array("NAME:CartesianLimitLineFromCurve", _ "TraceName:=", "Phase", "CurveName:=", "", _ "Start:=", "0deg", "Stop:=", "375deg", _ "YAxis:=", 1, "YOffset:=", 0, _ "CreateMode:=", "AboveCurve", _ "YShiftPercent:=", 10)</pre>

AddCartesianLimitLineFromEquation

Adds a limit line to a report from a specified equation.

UI Access	Report2D > Add Limit Line > Specify Equation...		
Parameters	Name	Type	Description
	<ReportName>	String	Name of the report.
	<LimitLineParams>	Array	Structured array. <pre>Array("NAME:CartesianLimitLineFromEquation", "YAxis:=", <integer Y-Axis number>, "Start:=", <string start frequency with unit>, "Stop:=", <string end frequency with unit>, "Step:=", <string frequency resolution with unit>, "Equation:=", <string specified equation>)</pre>

	"XValuesUnit:=", "GHz"
Return Value	None.

Python Syntax	AddCartesianLimitLineFromEquation(<ReportName>, <LimitLineParams>)
Python Example	<pre>oModule.AddCartesianLimitLineFromEquation("S Parameter Plot 1", ["NAME:CartesianLimitLineFromEquation", "YAxis:=" , 1, "Start:=" , "9GHz", "Stop:=" , "11GHz", "Step:=" , "0.2GHz", "Equation:=" , "x+1" "XValuesUnit:=" , "GHz"])</pre>

VB Syntax	AddCartesianLimitLineFromEquation <ReportName>, <LimitLineParams>
VB Example	<pre>oModule.AddCartesianLimitLineFromEquation _ "S Parameter Plot 1", _ Array("NAME:CartesianLimitLineFromEquation", _ "YAxis:=", 1, _</pre>

	"Start:=", "9GHz", "Stop:=", "11GHz", "Step:=", "0.2GHz", _ "Equation:=", "x+1", "XValuesUnit:=", "GHz")

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)

VB Syntax	AddCartesianXMarker <ReportName>, <MarkerName>, <XValue>
VB Example	oModule.AddCartesianXMarker "XY Plot1", "MX1", 0

AddCartesianYMarker

Adds a marker to a report on the Y axis.

UI Access	Report2D > Marker > Add Y Marker.		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<MarkerName>	String	Marker name, including any trailing number.
	<AxisName>	String	Name of axis.
	<YValue>	Double	Y coordinate.
	<CurveName>	String	Name of curve.
Return Value	None		

Python Syntax	AddCartesianYMarker (<ReportName>, <MarkerName>, <AxisName>, <YValue>, <CurveName>)		
Python Example	<pre>oModule.AddCartesianYMarker("XY Plot 1", "MY1", "Y1", 0, "dB() : Setup1 : Sweep1")</pre>		

VB Syntax	AddCartesianYMarker <ReportName>, <MarkerName>, <AxisName>, <YValue>, <CurveName>		
VB Example	<pre>oModule.AddCartesianYMarker "XY Plot 1", "MY1", "Y1", 0, _ "dB() : Setup1 : Sweep1"</pre>		

AddCartesianYMarkerToStack

Adds a marker to a stacked report on the Y axis.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.

	<MarkerName>	String	Marker name, including any trailing number.
	<AxisName>	String	Name of axis.
	<YValue>	Double	Y coordinate.
	<CurveName>	String	Name of curve.
	<StackOption>	Array	"Current" to create marker on the current stack. "All" to create marker on all stack.
Return Value	None		

Python Syntax	AddCartesianYMarkerToStack (<ReportName>, <MarkerName>, <AxisName>, <YValue>, <CurveName>, <StackOption>)
Python Example	<pre>oModule.AddCartesianYMarker("XY Plot 1", "MY1", "Y1", 0, "dB() : Setup1 : Sweep1", ["All"])</pre>

VB Syntax	AddCartesianYMarkerToStack <ReportName>, <MarkerName>, <AxisName>, <YValue>, <CurveName>, <StackOption>
VB Example	<pre>oModule.AddCartesianYMarkerToStack "XY Plot 1", "MY1", "Y1", 0, _ "dB() : Setup1 : Sweep1", Array("All")</pre>

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(A-MII-RXD1_30_SQFP28X28_208_U1 A-MII-RXD1_30_SQFP28X28_208_U1)) : SYZ Sweep 1 : SYZ Sweep 1 : Cartesian", "1.7GHz", "m2", "dB(S(A-MII-RXD1_30_SQFP28X28_208_U1 A-MII-RXD1_30_SQFP28X28_208_U1)) : SYZ Sweep 1 : SYZ Sweep 1 : Cartesian", "5.05GHz")</pre>

VB Syntax	AddDeltaMarker <ReportName>, <MarkerName1>, <CurveName1>, <PrimarySweepValue1>, <MarkerName2>, <CurveName2>, <PrimarySweepValue2>
VB Example	<pre>oModule.AddDeltaMarker "S Parameter Plot 1", "m1", "dB(S(A-MII-RXD1_30_SQFP28X28_208_U1 A-MII-RXD1_30_SQFP28X28_208_U1)) : SYZ Sweep 1 :</pre>

e	<pre>SYZ Sweep 1 : Cartesian", "1.7GHz", "m2", "dB(S(A-MII-RXD1_30_SQFP28X28_208_U1 A-MII-RXD1_30_SQFP28X28_208_U1)) : SYZ Sweep 1 : SYZ Sweep 1 : Cartesian", "5.05GHz"</pre>

AddMarker

Adds a marker to a trace on a report.

UI Access	Report2D > Marker > Add Marker.		
Parameters	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.
Return Value	None		

Python Syntax	AddMarker(<ReportName>, <MarkerName>, <CurveName>, <PrimarySweepValue>)
Python Example	<pre>oModule.AddMarker("XY Plot 1", "m5", "GS1.VAL : TR4 : Cartesian", "3.61599999999997s")</pre>

VB Syntax	<code>AddMarker <ReportName>, <MarkerName>, <CurveName>, <PrimarySweepValue></code>
VB Example	<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.

UI Access	Right-click on the plot and select Add Note .		
Parameters	Name	Type	Description
	<code><ReportName></code>	String	Name of report
	<code><NoteDataArray></code>	Array	Structured array. <code>Array("NAME:<StringDataName>", <NoteArray>)</code>
	<code><NoteArray></code>	Array	Structured array: <code>Array("NAME:<NoteDataSourceName>", "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>)</code>

Return Value	None

Python Syntax	AddNote (<ReportName>, <NoteDataArray>)
Python Example	<pre>oModule.AddNote("XY Plot1", ["NAME:NoteDataSource", "SourceName:=", "Note1", "HaveDefaultPos:=", False, "DefaultXPos:=", 1996, "DefaultYPos:=", 3177, "String:=", "This is a note."])</pre>

VB Syntax	AddNote <ReportName> <NoteDataArray>
VB Example	<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 Add Trace Characteristics dialog box.		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<FunctionName>	String	The function name. See the Functions column of the Add Trace Characteristics dialog box.
	<FunctionArgs>	Array	Array containing string values for any function arguments. Pass empty array if no arguments exist. To see which argument(s) a function takes, see the bottom of the Add Trace Characteristics dialog box. For function with one argument: <pre>Array(<value>)</pre> For function with multiple arguments: <pre>Array(<value>, <value>, ...)</pre>
	<RangeArgs>	Array	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. For example: <pre>Array("Specified", "19.5GHz", "24.4GHz")</pre>
Return Value	None.		

Python Syntax	AddTraceCharacteristics (<ReportName>, <FunctionName>, <FunctionArgs>, <RangeArgs>)
Python Example	<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>

VB Syntax	AddTraceCharacteristics <ReportName>, <FunctionName>, <FunctionArgs>, <RangeArgs>
VB Example	<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 Modify Report dialog box.
	<ContextArray>	Array	Context for which the expression is being evaluated. This can be an

			empty string if there is no context. <pre>Array("Domain:=", <DomainType>) <DomainType> ex. "Sweep" or "Time" Array("Context:=", <GeometryType>) <GeometryType> ex. "Infinite Spheren", "Spheren", "Polylinen"</pre>
	<FamiliesArray>	Array	Contains sweep definitions for the report. <pre>Array("<VariableName>:= ", <ValueArray>) <ValueArray> Array("All") or Array("Value1", "Value2", ... "Valuen") examples of <VariableName> "Freq", "Theta", "Distance"</pre>
	<ReportDataArray>	Array	This array contains the report quantity and X, Y, and (Z) axis definitions. <pre>Array("X Component:=", <VariableName>, "Y Component:=", <VariableName> <ReportQuant- ityArray>) <ReportQuantityArray> ex. Array("dB(S(Port1, Port1))")</pre>
Return Value	None		

Python Syntax	Add Traces(<ReportName>, <SolutionName>, <ContextArray>, <FamiliesArray>, <ReportDataArray>)
Python Example	oModule.AddTraces("XY Plot1", "Setup1 : Sweep1",

	<pre>["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>

VB Syntax	Add Traces <ReportName> <SolutionName> <ContextArray> <FamiliesArray> <ReportDataArray>
VB Example	<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", _ "Y Component:=", Array("TDRZ (WavePort1)")), _</pre>

	<code>Array()</code>

ApplyReportTemplate

Applies settings to a report from a template file.

UI Access	Right-click on a report, select Report Templates > Apply Settings .		
Parameters	Name	Type	Description
	<code><ReportName></code>	String	Name of the report to apply settings to.
	<code><TemplateFile></code>	String	Template file name with path.
	<code><PropertyType></code>	String	Property types to apply. ("Graphical" "Data" "All")
Return Value	None.		

Python Syntax	<code>ApplyReportTemplate(<ReportName>, <TemplateFile>, <PropertyType>)</code>
Python Example	<code>oModule.ApplyReportTemplate("Variables Plot 1", "C:/MyTemplate.rpt", "Graphical")</code>

VB Syntax	<code>ApplyReportTemplate <ReportName>, <TemplateFile>, <PropertyType></code>
VB Example	<code>oModule.ApplyReportTemplate "Variables Plot 1", "C:/MyTemplate.rpt", "Graphical"</code>

ChangeProperty[ReportSetup]

Change the properties for a Report.

UI Access	Double-click on a report to change its properties.		
Parameters	Name	Type	Description
	<PropertyArray>	Array	Varies, depending on the properties associated with the select object command.
Return Value	None.		

Python Syntax	ChangeProperty(<PropertyArray>)
Python Example	<pre>oModule = oDesign.GetModule("ReportSetup") oModule.ChangeProperty(["NAME:AllTabs", ["NAME:General", ["NAME:PropServers", "XY Plot 1:General"],], ["NAME:ChangedProps", [</pre>

```
"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"  
    ]  
]  
]  
])  
oModule.ChangeProperty(  
[  
    "NAME:AllTabs",  
    [  
        "NAME:General",  
        [  
            "NAME:PropServers",  
            "S Parameter Plot 1:General"  
        ],  
    ]  
[  
    "NAME:ChangedProps",  
    [  
        "NAME:Auto Scale Fonts",  
        "Value:=" , False  
    ]  
]
```

```
    ]
  ]
])

### Example For Rectangular Stacked Plot

oProject = oDesktop.SetActiveProject("antenna_test")
oDesign = oProject.SetActiveDesign("Design1")
oModule = oDesign.GetModule("ReportSetup")
oModule.ChangeProperty(
  [
    "NAME:AllTabs",
    [
      "NAME:Scaling",
      [
        "NAME:PropServers",
        "S Parameter Plot 1:AxisY2:Stack1",
        "S Parameter Plot 1:AxisY2:Stack2"
      ],
    ],
    [
      "NAME:ChangedProps",
```

```
        [
            "NAME:Min",
            "Value:="                , "-60"
        ]
    ]
]
```

Example Change eye rect and eye plot:

```
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("test_nport_synth2")
oDesign = oProject.SetActiveDesign("Circuit1")
oModule = oDesign.GetModule("ReportSetup")
oModule.ChangeProperty(
    [
        "NAME:AllTabs",
        [
            "NAME:Grid",
            [
                "NAME:PropServers",
```

```
        "QuickEye Voltage Plot 1:Grid:Eye"
    ],
    [
        "NAME:ChangedProps",
        [
            "NAME:Major grid line style",
            "Value:=" , "Dot"
        ],
        [
            "NAME:Minor grid line style",
            "Value:=" , "Dot"
        ]
    ]
],
[
    "NAME:Axis",
    [
        "NAME:PropServers",
        "QuickEye Voltage Plot 1:AxisX:Eye"
    ],
    [
```

```
        "NAME:ChangedProps",
        [
            "NAME:Axis Color",
            "R:=" , 255,
            "G:=" , 128,
            "B:=" , 128
        ]
    ],
    [
        "NAME:Axis",
        [
            "NAME:PropServers",
            "QuickEye Voltage Plot 1:AxisY1:Rect"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Axis Color",
                "R:=" , 255,
```

```
        "G:=" , 128,  
        "B:=" , 128  
    ]  
]  
]  
])  
oModule.ChangeProperty(  
    [  
        "NAME:AllTabs",  
        [  
            "NAME:Grid",  
            [  
                "NAME:PropServers",  
                "QuickEye Voltage Plot 1:Grid:Rect"  
            ],  
            [  
                "NAME:ChangedProps",  
                [  
                    "NAME:Major grid line style",  
                    "Value:=" , "LongDash"  
                ],  
            ],  
        ],  
    ],  
)
```

	<pre> ["NAME:Minor grid line style", "Value:=", "LongDash"]]]])</pre>

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: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.

UI Access	Report2D > Markers > Clear All.		
Parameters	Name	Type	Description
	<ReportName>	String	Name of Report
Return Value	None		

Python Syntax	ClearAllMarkers(<ReportName>)
Python Example	<code>oModule.ClearAllMarkers("XY Plot 1")</code>

VB Syntax	ClearAllMarkers <ReportName>
VB Example	<code>oModule.ClearAllMarkers "XY Plot 1"</code>

ClearAllTraceCharacteristics

Clears all trace characteristics from the legend in a report.

UI Access	Report2D > Trace Characteristics > Clear All.		
Parameters	Name	Type	Description
	<PlotName>	String	Name of the plot
Return Value	None		

Python Syntax	<code>ClearAllTraceCharacteristics(<PlotName>)</code>
Python Example	<code>oModule.ClearAllTraceCharacteristics("XY Plot 1")</code>

VB Syntax	<code>ClearAllTraceCharacteristics <PlotName></code>
VB Example	<code>oModule.ClearAllTraceCharacteristics "XY Plot 1"</code>

CloneReportsFromDatasetSolution

Clones a report for a solved solution from a dataset solution.

UI Access	Right-click the Project tree on the report and choose Clone from Dataset Solution > [Dataset_SolutionName]		
Parameters	Name	Type	Description
	<code><ReportsToClone></code>	Array	Array of report names to clone.
	<code><SoluNameToUse></code>	String	Name of the dataset solution.
Return Value	None.		

Python Syntax	<code>CloneReportsFromDatasetSolution(<ReportsToClone>, <SoluNameToUse>)</code>
Python Example	<code>oModule.CloneReportsFromDatasetSolution(["Rectangular Plot1"], "DatasetSolution_rsm_5812")</code>

VB Syntax	CloneReportsFromDatasetSolution <ReportsToClone>, <SoluNameToUse>
VB Example	oModule.CloneReportsFromDatasetSolution _ Array("Rectangular Plot1"), "DatasetSolution_rsm_5812"

CopyPlotSettings

Copies settings of a specified plot.

UI Access	Right-click a report, select Copy		
Parameters	Name	Type	Description
	<ReportName>	String	Name of specified report.
Return Value	None.		

Python Syntax	CopyPlotSettings(<ReportName>)
Python Example	oModule.CopyPlotSettings("Plot 1")

VB Syntax	CopyPlotSettings <ReportName>
VB Example	oModule.CopyPlotSettings "Plot 1"

CopyReportDefinitions

Copy the definition of a report for paste operations.

UI Access	Select a report in the Project tree, right-click and select Copy Definition		
Parameters	Name	Type	Description
	<ReportsArray>	Array	Names of reports from which to copy the definitions
Return Value	None		

Python Syntax	CopyReportDefinitions(<ReportsArray>)
Python Example	<pre>oModule.CopyReportDefinitions(["Transmission", "Reflection"])</pre>

VB Syntax	CopyReportDefinitions <ReportsArray>
VB Example	<pre>oModule.CopyReportDefinitions _ Array("Transmission", "Reflection")</pre>

CopyReportsData

Copy all data corresponding to the specified reports.

UI Access	Select a report in the Project tree, right-click and select Copy Data		
Parameters	Name	Type	Description
	<ReportsArray>	Array	Names of reports from which to copy data
Return Value	None		

Python Syntax	CopyReportsData (<ReportsArray>)
Python Example	oModule.CopyReportsData (["Transmission", "Reflection"])

VB Syntax	CopyReportsData <ReportsArray>
VB Example	oModule.CopyReportsData _ Array("Transmission", "Reflection")

CopyTraceDefinitions

Copy trace definitions for a paste operation.

UI Access	Select a trace in the Project tree, right-click and select Copy Definition		
Parameters	Name	Type	Description
	<ReportName>	String	Name of Report
	<TracesArray>	Array	Trace definitions to copy
Return Value	None		

Python Syntax	CopyTraceDefinitions(<ReportName>, <TracesArray>)
Python Example	oModule.CopyTraceDefinitions ("Transmission", ["mag (S (Port1, Port2)) "])

VB Syntax	CopyTraceDefinitions <ReportName>, <TracesArray>
VB Example	<pre>oModule.CopyTraceDefinitions "Transmission", _ Array("mag(S(Port1,Port2))")</pre>

CopyTracesData

Copies trace data for a paste operation.

UI Access	Select a trace in the Project tree, right-click and select Copy Data .		
Parameters	Name	Type	Description
	<ReportName>	String	Name of Report.
	<TraceArray>	Array	Trace definitions from which to copy corresponding data.
Return Value	None.		

Python Syntax	CopyTracesData(<ReportName>, <TracesArray>)
Python Example	<pre>oModule.CopyTracesData ("Transmission", ["mag(S(Port1,Port2))"])</pre>

VB Syntax	CopyTracesData <ReportName>, <TracesArray>
VB Example	<pre>oModule.CopyTracesData _ "C11", Array("mag(S(Port1,Port2))")</pre>

CreateReport

Creates a new report with a single trace and adds it to the **Results** branch in the project tree.

UI Access	Right-click on Results > Create [Type] Report		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<ReportType>	String	Type of report. Possible values are: "Modal S Parameters" - Only for Driven Modal solution-type problems with ports. "Terminal S Parameters" - Only for Driven Terminal solution-type problems with ports. "Eigenmode Parameters" - Only for Eigenmode solution-type problems. "Fields" "Far Fields" - Only for problems with radiation or PML boundaries. "Near Fields" - Only for problems with radiation or PML boundaries. "Emission Test"
	<DisplayType>	String	Type of display. If ReportType is "Modal S Parameters", "Terminal S Parameters", or "Eigenmode Parameters", then set to one of the following: "Rectangular Plot", "Polar Plot", "Radiation Pattern", "Smith Chart", "Data Table", "3D Rectangular Plot", "3D Rectangular Bar Plot", "Rectangular Stacked Plot" or "3D Polar Plot". If <ReportType> is "Fields", then set to one of the following:

		"Rectangular Plot", "Polar Plot", "Radiation Pattern", "Rectangular Contour Plot", "Data Table", "3D Rectangular Plot", "Rectangular Stacked Plot" or "3D Rectangular Bar Plot". If <ReportType> is "Far Fields" or "Near Fields", then set to one of the following: "Rectangular Plot", "Radiation Pattern", "Rectangular Contour Plot", "Data Table", "3D Rectangular Plot", "3D Rectangular Bar Plot" "Rectangular Stacked Plot" or "3D Polar Plot" If <ReportType> is "Emission Test", then set to one of the following: "Rectangular Plot" or "Data Table"
<SolutionName>	String	Name of the solution as listed in the Modify Report dialog box.
<ContextArray>	Array	Context for which the expression is being evaluated. This can be an empty string if there is no context. <pre>Array("Domain:=", <DomainType>)</pre> <DomainType> ex. "Sweep" or "Time" <pre>Array("Context:=", <GeometryType>)</pre> <GeometryType> ex. "Infinite Spheren", "Spheren", "Polylinen"
<FamiliesArray>	Array	Contains sweep definitions for the report. <pre>Array("<VariableName>:= ", <ValueArray>)</pre> <ValueArray> <pre>Array("All") or Array("Value1", "Value2", ... "Valuen")</pre>

	<table border="1"> <tr> <td></td><td>examples of <VariableName> "Freq", "Theta", "Distance"</td></tr> <tr> <td><ReportDataArray></td><td> Array This array contains the report quantity and X, Y, and (Z) axis definitions. <pre>Array("X Component:=", <VariableName>, "Y Component:=", <VariableName> <ReportQuantityArray>)</pre> <ReportQuantityArray> ex. <code>Array("dB(S (Port1, Port1))")</code> </td></tr> </table>		examples of <VariableName> "Freq", "Theta", "Distance"	<ReportDataArray>	Array This array contains the report quantity and X, Y, and (Z) axis definitions. <pre>Array("X Component:=", <VariableName>, "Y Component:=", <VariableName> <ReportQuantityArray>)</pre> <ReportQuantityArray> ex. <code>Array("dB(S (Port1, Port1))")</code>
	examples of <VariableName> "Freq", "Theta", "Distance"				
<ReportDataArray>	Array This array contains the report quantity and X, Y, and (Z) axis definitions. <pre>Array("X Component:=", <VariableName>, "Y Component:=", <VariableName> <ReportQuantityArray>)</pre> <ReportQuantityArray> ex. <code>Array("dB(S (Port1, Port1))")</code>				
Return Value	None.				

Python Syntax	CreateReport(<ReportName>, <ReportType>, <DisplayType>, <SolutionName>, <ContextArray>, <FamiliesArray>, <ReportDataArray>, <ExtendedInfo>)
Python Example	Three examples are given below: (nominal, Optimetrics, spectral) Nominal Example: <pre>oModule.CreateReport("XY Plot 2", "Standard", "Rectangular Plot", "TR", ["NAME:Context", "SimValueContext:=", [], 1, 0, 2, 0, false, false, -1, 1, 0, 1, 1, "", 0, 0]], ["Time:=" , ["All"], "aaa:=" , ["Nominal"]], ["X Component:=", "Time",</pre>


```
        "Y Component:=", ["E1.I"]
    ],
    [])
```

Optimetrics Example:

```
oModule.CreateReport ("XY Plot 3", "Standard",
    "Rectangular Plot", "TR",
    [
        "NAME:Context",
        "OptiSetup:=", "ParametricSetup1", "SimValueContext:=",
        [
            1, 0, 2, 0, false, false, 0, 1,0, 1, 1, "", 0, 0
        ]
    ],
    [
        "Time:=", ["All"], "aaa:=", ["Nominal"]
    ],
    [
        "X Component:=" , "Time",
        "Y Component:=" , ["E1.I"]
    ],
    [])
```

Spectral Example:

```
oModule.CreateReport ("XY Plot 4", "Standard",
    "Rectangular Plot", "TR",
    [
        "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, "40ms", "TH", false,
            "40", "TS", false, "0ns", "UF", false, "0",
            "WT", false, "0", "WW", false, "100"
        ]
    ],
    [
        "Spectrum:=", ["All"], "aaa:=", ["Nominal"]
    ],
    [
        "X Component:=", "Spectrum",
        "Y Component:=", ["mag(E1.I)"]
    ],
    [])

```

Partial Discharge Example

```

oModule.CreateReport("Partial Discharge Plot 2",
    "Partial Discharge", "Rectangular Plot",
    "PartialDischarge1 : PartialDischarge", [],
    [
        "GasPressure:=", ["All"],
        "Freq:=", ["All"],
        "bend_angle:=", ["All"]
    ],
    [
        "X Component:=", "GasPressure",
        "Y Component:=", ["Power"]
    ])

```

Example 3D Rectangular Bar Plot with Change Bar properties

```
oModule = oDesign.GetModule("ReportSetup")
oModule.CreateReport("S Parameter Plot 4", "Modal Solution Data",
    "3D Rectangular Bar Plot", "Setup1 : Sweep", [],
    [
        "Freq:="          , ["All"],
        "UU:="            , ["All"],
        "ZZ:="            , ["Nominal"]
    ],
    [
        "X Component:=", "Freq",
        "Y Component:=", "UU",
        "Z Component:=", ["dB(S(wDipole1_1_p1,wDipole1_1_p1))"]
    ])

oModule.ChangeProperty(
    [
        "NAME:AllTabs",
        [
            "NAME:Bar",
            [
                "NAME:PropServers"
                "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
            ],
        ],
    ],
```

```

        [
            "NAME:ChangedProps",
            [
                "NAME:Transparency",
                "Value:=", "0.5"
            ]
        ]
    ]
))

oModule.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:Bar",
        [
            "NAME:PropServers",
            "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Filled",
                "Value:=", False
            ]
        ]
    ]
]

```

```
ChangeProperty(  
    [  
        "NAME:AllTabs",  
        [  
            "NAME:Bar",  
            [  
                "NAME:PropServers",  
                "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"  
            ],  
            [  
                "NAME:ChangedProps",  
                [  
                    "NAME:Outline Color",  
                    "R:=", 0,  
                    "G:=", 0,  
                    "B:=", 160  
                ]  
            ]  
        ]  
    ]  
)  
  
oModule.ChangeProperty(  
    [  
        "NAME:AllTabs",  
        [  
            "NAME:Bar",  
            [  
                "NAME:PropServers",  
                "S Parameter Plot 4:Traces-dB(S(wDipole1_1_p1,wDipole1_1_p1))"
```


VB Syntax	CreateReport <ReportName>, <ReportType>, <DisplayType>, <SolutionName>, <ContextArray>, <FamiliesArray>, <ReportDataArray>, <ExtendedInfo
VB Example	<pre> oModule.CreateReport "3D Cartesian Plot1", "Far Fields", _ "3D Cartesian Plot", "Setup1 : LastAdaptive", _ Array("Context:=", "Infinite Spherel", "Domain:=", "Sweep"), _ Array("Theta:=", Array("All"), "Phi:=", Array("All"), _ "Freq:=", Array("10GHz")), _ Array("X Component:=", "Theta", _ "Y Component:=", "Phi", _ "Z Component:=", Array("rETotal")), _ Array() oModule.CreateReport "ReptSmithFreq", _ "Modal Solution Data", "Smith Plot", "Setup1 : Sweep1", _ Array("Domain:=", "Sweep"), _ Array("Freq:=", Array("All")), _ Array("Polar Component:=", _ Array("ln(Y(LumpPort1,LumpPort1))")), _ Array() oModule.CreateReport "Rectangular Contour Plot 2", "Far Fields", _ "Rectangular Contour Plot", "Setup1 : LastAdaptive", Array("Context:=", _ "Infinite Spherel"), Array("_u:=", Array("All"), "_v:=", Array("All"), "Freq:=", Array(_ "5GHz")), Array("X Component:=", "_u", "Y Component:=", "_v", "Z Component:=", Array (_ </pre>

	<pre>"rETotal")) , Array() oModule.CreateReport "Rept2DRectFreq", _ "Modal Solution Data", "XY Plot", _ "Setup1 : Sweep1", _ Array("Domain:=", "Sweep"), _ Array("Freq:=", Array("All")), _ Array("X Component:=", "Freq", _ "Y Component:=", _ Array("dB(S(LumpPort1,LumpPort1))"), _ Array()</pre>

CreateReportFromTemplate

Creates a report from a saved template.

UI Access	[product] > Results > Report Templates > PersonalLib > [TemplateName]		
Parameters	Name	Type	Description
	<TemplatePath>	String	Path to report template
Return Value	None		

Python Syntax	CreateReportFromTemplate(<TemplatePath>)
Python Example	<pre>oModule.CreateReportFromTemplate("C:/MyHFSSProjects/PersonalLib/ReportTemplates/TestTemplate.rpt")</pre>

VB Syntax	CreateReportFromTemplate <TemplatePath>
VB Example	oModule.CreateReportFromTemplate "C:/MyHFSSProjects/PersonalLib/" _ "ReportTemplates/TestTemplate.rpt"

CreateReportOfAllQuantities

Creates a report including all quantities in a category. Cannot create a report with expressions.

UI Access	NA		
Parameters	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
	<SimValueCtxt>	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)
Return Value	None.		

Python Syntax	CreateReportOfAllQuantities(<ReportNameArg>, <ReportType>, <DisplayType>, <SolutionName>, <SimValueCtxt>, <CategoryName>, <PointSet>, <CommonComponentsOfTraces>, <ExtTraceInfo>)
Python Example	oModule.CreateReportOfAllQuantities("Smith Chart all", "Modal Solution Data", "Smith Chart", "Setup1 : LastAdaptive",

	<pre>[],"S Parameter",["Freq=", ["All"], "offset=", ["All"],"a=", ["Nominal"],"b=", ["Nominal"]],[],[])</pre>

VB Syntax	<pre>CreateReportOfAllQuantities <ReportNameArg>, <ReportType>, <DisplayType>, <SolutionName>, <SimValueCtxt>, <CategoryName>, <PointSet>, <CommonComponentsOfTraces>, <ExtTraceInfo></pre>
VB Example	<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.

UI Access	[product] > Fields > Fields > Marker > Delete Marker .		
Parameters	Name	Type	Description
	<MarkerName>	String	Name of the marker.

Return Value	None.

Python Syntax	DeleteMarker(<MarkerName>)
Python Example	<code>oModule.DeleteMarker("m1")</code>

VB Syntax	DeleteMarker <MarkerName>
VB Example	<code>oModule.DeleteMarker "m1"</code>

DeleteAllReports

Deletes all existing reports.

UI Access	Right-click the report to delete in the project tree, and then click Delete All Reports on the shortcut menu.
Parameters	None.
Return Value	None.

Python Syntax	DeleteAllReports()
Python Example	<code>oModule.DeleteAllReports()</code>

VB Syntax	DeleteAllReports
VB Example	<code>oModule.DeleteAllReports</code>

DeleteReports

Deletes an existing report or reports.

UI Access	Right-click the report to delete in the project tree, and then click Delete on the shortcut menu		
Parameters	Name	Type	Description
	<ReportNameArray>	Array	Array of report names to be deleted
Return Value	None.		

Python Syntax	DeleteReports(<ReportNameArray>)
Python Example	oModule.DeleteReports (["Rept2DRectFreq"])

VB Syntax	DeleteReports <ReportNameArray>
VB Example	oModule.DeleteReports Array ("Rept2DRectFreq")

DeleteTraces

Deletes an existing traces or traces.

UI Access	Right-click the Trace to delete in the project tree, and then click Delete on the shortcut menu		
Parameters	Name	Type	Description
	<TraceSelection>	Array	Structured array define selections.

			Array("<ReportName>:=", <TracesArray>, <TracesArray>, ...)
	<ReportName>	String	Name of Report
	<TracesArray>	Array	Contains the traces to delete within a report Array(<Trace>, <Trace>, ...)
	<Trace>	String	A specific trace that the user wishes to delete
Return Value	None.		

Python Syntax	DeleteTraces(<TraceSelection>)
Python Example	<pre>oModule.DeleteTraces (["XY Plot 1:=", ["dB(S(LumpPort1,LumpPort1))"], "XY Plot 2:=", ["Mag_E"]])</pre>

VB Syntax	DeleteTraces <TraceSelection>
VB Example	<pre>oModule.DeleteTraces Array("XY Plot 1:=", _ Array("dB(S(LumpPort1,LumpPort1))"),) "XY Plot 2:=", Array("Mag_E"))</pre>

DoesSupportTraceCharacteristics

Determines whether trace characteristics is supported in a specified display type.

UI Access	N/A

Parameters	Name	Type	Description
	<DisplayType>	String	Specified display type to check.
Return Value	Integer • 1 - trace characteristics is supported. • 0 - trace characteristics not supported.		

Python Syntax	DoesSupportTraceCharacteristics(<DisplayType>)
Python Example	<code>oModule.DoesSupportTraceCharacteristics("Rectangular Plot")</code>

VB Syntax	DoesSupportTraceCharacteristics <DisplayType>
VB Example	<code>oModule.DoesSupportTraceCharacteristics "Rectangular Plot"</code>

EditCartesianXMarker

Edits an XMarker value.

UI Access	N/A		
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	EditCartesianXMarker (<ReportName>, <MarkerName>, <XValue>)
Python Example	<code>oModule.EditCartesianXMarker ("XY Plot 1", "MX1", 0)</code>

VB Syntax	EditCartesianXMarker <ReportName>, <MarkerName>, <XValue>
VB Example	<code>oModule.EditCartesianXMarker "XY Plot1", "MX1", 0</code>

EditCartesianYMarker

Edits a YMarker value.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<MarkerName>	String	Marker name, including any trailing number.
	<AxisName>	String	Name of axis.
	<YValue>	Double	Y coordinate.
	<CurveName>	String	Name of curve.
Return Value	None		

Python Syntax	EditCartesianYMarker (<ReportName>, <MarkerName>, <AxisName>, <YValue>, <CurveName>)
Python Example	<code>oModule.EditCartesianYMarker("XY Plot 1", "MY1", "Y1", 0,</code>

	<code>"dB() : Setup1 : Sweep1")</code>

VB Syntax	<code>EditCartesianYMarker <ReportName>, <MarkerName>, <AxisName>, <YValue>, <CurveName></code>
VB Example	<code>oModule.EditCartesianYMarker "XY Plot 1", "MY1", "Y1", 0, _ "dB() : Setup1 : Sweep1"</code>

EditMarker

Edits a marker on a report.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><ReportName></code>	String	Name of report.
	<code><MarkerName></code>	String	Marker name, including any trailing number.
	<code><CurveName></code>	String	Full trace name.
	<code><PrimarySweepValue></code>	String	Primary sweep value, including unit.
Return Value	None.		

Python Syntax	<code>EditMarker(<ReportName>, <MarkerName>, <CurveName>, <PrimarySweepValue>)</code>
Python Example	<code>oModule.EditMarker("XY Plot 1", "m5", "GS1.VAL : TR4 : Cartesian", "3.61599999999997s")</code>

VB Syntax	EditMarker <ReportName>, <MarkerName>, <CurveName>, <PrimarySweepValue>
VB Example	<pre>Set oModule = oDesign.GetModule("ReportSetup") oModule.EditMarker "XY Plot1", "m1", "mag(S(Port1 Port1)) : Setup1: LastAdaptive : Cartesian", "0.3in"</pre>

ExportEyeMaskViolation

Exports eye mask violations to a file.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of specified report.
	<ExportFileName>	String	File name to export to.
Return Value	N/A		

Python Syntax	ExportEyeMaskViolation(<ReportName>, <ExportFileName>)
Python Example	oModule.ExportEyeMaskViolation("Variables Plot 1", "C:/eyemask1.csv")

VB Syntax	ExportEyeMaskViolation <ReportName>, <ExportFileName>
VB Example	oModule.ExportEyeMaskViolation "Variables Plot 1", "C:/eyemask1.csv"

ExportImageToFile [Reporter]

Exports a report image in a specified format. This command fully supports -ng (non-graphical) mode.

UI Access	N/A		
Parameters	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 if report window is closed.
Return Value	None.		

Python Syntax	ExportImageToFile(<ReportName>, <FileName>, <Width>, <Height>)
Python Example	<pre>##### Script Recorded for Saving 2D oDesktop.OpenProject("E:/Ansoft/Tee2.aedt") oProject = oDesktop.SetActiveProject("Tee2") oDesign = oProject.SetActiveDesign("TeeModel") oModule = oDesign.GetModule("ReportSetup") oModule.ExportImageToFile("S Parameter Plot 1", "E:/Ansoft/S Parameter Plot 1.png", 1920, 1080)</pre>

```
oModule.ExportImageToFile("S Parameter Plot 12", "E:/Ansoft/S Parameter Plot 12.png",
1920, 1080)

oModule.ExportImageToFile("S Parameter Plot 1", "E:/Ansoft/S Parameter Plot 1.png",
1920, 1080,
[
    "NAME:SaveImageParams",
    "ShowLegend:=" , "False",
    "ShowMarkerTable:=" , "False"
])

oModule.ExportImageToFile("S Parameter Plot 1", "E:/Ansoft/S Parameter Plot 1.png",
1920, 1080,
[
    "NAME:SaveImageParams",
    "ShowLegend:=" , "True",
    "ShowMarkerTable:=" , "True"
])

oModule.ExportImageToFile("S Parameter Plot 12", "E:/Ansoft/S Parameter Plot 12.png",
1920, 1080,
[
    "NAME:SaveImageParams",
    "ShowLegend:=" , "False",
    "ShowMarkerTable:=" , "Default"
])
```

```
oModule.ExportImageToFile("S Parameter Plot 12", "E:/Ansoft/S Parameter Plot 12.png",
1920, 1080,

[
    "NAME:SaveImageParams",
    "ShowLegend:=" , "True",
    "ShowMarkerTable:=" , "Default"
])

oDesktop.CloseProject("Tee2")
```

Script Recorded for saving as 3D:

```
oDesktop.OpenProject("E:/Ansoft/Tee2.aedt")
oProject = oDesktop.SetActiveProject("Tee2")
oDesign = oProject.SetActiveDesign("TeeModel")
oModule = oDesign.GetModule("ReportSetup")
oModule.ExportImageToFile("S Parameter Plot 6", "E:/Ansoft/S Parameter Plot 11.png",
1335, 466,

[
    "NAME:SaveImageParams",
    "AutoFit:=" , "False",
    "Orientation:=" , "",
```

```
        "ShowOrientationGadget:=", "Default"

    })

oModule.ExportImageToFile("S Parameter Plot 6", "E:/Ansoft/S Parameter Plot 11.png",
1335, 466,

    [

        "NAME:SaveImageParams",

        "AutoFit:=" , "False",

        "Orientation:=" , "",

        "ShowOrientationGadget:=", "Default"

        "ShowLegend:=" , "True"

    ])

oModule.ExportImageToFile("S Parameter Plot 6", "E:/Ansoft/S Parameter Plot 10.png",
1335, 466,

    [

        "NAME:SaveImageParams",

        "AutoFit:=" , "False",

        "Orientation:=" , "",

        "ShowOrientationGadget:=", "Default",

        "ShowLegend:=" , "False"

    ])

})
```

VB Syntax	ExportImageToFile <ReportName>, <FileName>, <Width>, <Height>
VB Example	<pre>oModule.ExportImageToFile _ "Rectangular Contour Plot 1", _ "D:/work/2015/UV-Export/pp1.gif", 0, 0</pre>

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 Ensign use *.avz. For export to Ensign in -ng mode, the corresponding version of Ensign 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 Ensign 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 boolean>, "ShowGrid:=" , <string containing boolean>,

		<pre> "ShowRuler:=" , <string containing boolean>, "ShowRegion:=" , <string>, "Selections:=" , <string>, "FieldPlotSelections:=", <string>' # Comma delimited string. #Use to set which field plot to show. "FitToSelections:=" , "", </pre> Note: "FitToSelections" specify geometry objects for the "Fit" operation. <pre> "FitToFieldPlotSelection:=" , "" </pre> Note: "FitToFieldPlotSelections" specifies field plots for the "Fit" operation. <pre> "AutoFit:=" , "True", </pre> Note: If FitToSlections or FitToFieldPlotSelections are used , then then AutoFit is true, it makes sure color key does not overlap field plot. It is true by default. If neither is used, then "AutoFit" will "Fit" to full model. <pre> "Orientation:=" , <string> "ShowOrientationGadget:=" , <False> </pre>

Return Value	None.

Python Syntax	ExportModelImageToFile(<path> <width> <height> <Parameters>)
Python Example	<pre>oEditor.ExportModelImageToFile ("C:/Users/jdoe/Desktop/export.bmp", 1920, 1080, ["NAME:SaveImageParams", "ShowAxis:=" , "True", "ShowGrid:=" , "True", "ShowRuler:=" , "True", "ShowRegion:=" , "Default", "Selections:=" , "", "FieldPlotSelections:=" , "", "FitToSelections:=" , "", "FitToFieldPlotSelections:=" , "", "AutoFit:=" , "True", "Orientation:=" , ""])</pre>

VB Syntax	ExportModelImageToFile <path> <width> <height> <Parameters>
VB Example	<pre>oEditor.ExportModelImageToFile "C:/Users/jdoe/Desktop/export.bmp", 1383, 512, Array("NAME:SaveImageParams", "ShowAxis:=", "True", "ShowGrid:=", "True", "ShowRuler:=", "True", "ShowRegion:=", "Default", "Selections:=", "", "FitToFieldPlotSelections:=", "", "AutoFit:=", "True", "Orientation:=", ""</pre>

ExportModelMeshToFile

Exports geometry model to a 3D model file (e.g. *.obj, *.wrl, etc.).

UI Access	N/A		
Parameters	Name	Type	Description
	<filePath>	String	Full file path, including extension *.obj, *.wrl, etc
	<selections>	Array	Selected parts.

Return Value	None.

Python Syntax	ExportModelMeshToFile <filePath>, <selections>)
Python Example	<pre>oEditor.ExportModelMeshToFile("E:/MyDir/scriptRun/2Selected-ng.obj", ["BotCover-", "AveragingVolumeAtPeakRMSEfieldLocation"])</pre>

VB Syntax	ExportModelMeshToFile <filePath>, <selections>
VB Example	<pre>oEditor.ExportModelMeshToFile("E:/MyDir/scriptRun/2Selected-ng.obj", ["BotCover-", "AveragingVolumeAtPeakRMSEfieldLocation"])</pre>

ExportPlot3DToFile [Reporter]

Use: Exports 3D polar, spherical and rectangular plot to a case file. It works in both graphical and NG mode..

Command: None.

Syntax: ExportPlot3DToFile(<plotName>, <path>)

Return Value: A 3D plot file.

Parameters: <plotName>

Type: <string>

Plot name.

<Path>

Type: <string>

Path to file.

Python Syntax	ExportPlot3DToFile(<name> <Path>)
Python Example	<pre> oModule = oDesign.GetModule("ReportSetup") oModule.UpdateReports(["rE Plot 1"]) oModule.UpdateReports(["Directivity Plot 1"]) oModule.UpdateReports(["Gain Plot 1"]) oModule.ExportPlot3DToFile("rE Plot 1", "D:/projects/test2-output/rEPlot1.case") oModule.ExportPlot3DToFile("Directivity Plot 1", "D:/projects/test2-output/DirectivityPlot1.case") oModule.ExportPlot3DToFile("Gain Plot 1", "D:/projects/test2-output/GainPlot1.case") </pre>

VB Syntax	ExportPlot3DToFile <plotName> <Path>
VB Example	<pre> oModule.ExportPlot3DToFile("rE Plot 1", "D:/projects/test2-output/rEPlot1.case") </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"

```

Python Syntax	ExportReport(<ReportName>, <FileName>)
Python Example	oDesign.ExportReport("Plot1", "c:\report1.dat")

ExportReportDataToFile

Exports report data to a file.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of specified report.
	<ExportFileName>	String	Name of export file. File extension "rdat" is expected.
Return Value	None.		

Python Syntax	ExportReportDataToFile(<ReportName>, <ExportFileName>)
Python Example	oModule.ExportReportDataToFile("Plot 1", "C:/Plot1data.rdat")

VB Syntax	ExportReportDataToFile <ReportName>, <ExportFileName>

VB Example	<code>oModule.ExportReportDataToFile "Plot 1", "C:/Plot1data.rdat"</code>

ExportTableToFile

Exports a marker table from a report to a file.

UI Access	Right-click on a plot, select Marker > Export Marker Table... or Export Delta Marker Table...		
Parameters	Name	Type	Description
	<ReportName>	String	Name of specified report.
	<ExportFileName>	String	Name of export file.
	<TableType>	String	Type of marker table to export. "Marker" or "DeltaMarker".
Return Value	None.		

Python Syntax	<code>ExportTableToFile(<ReportName>, <ExportFileName>, <TableType>)</code>
Python Example	<code>oModule.ExportTableToFile("Plot 1", "C:/Marker.csv", "Marker")</code>

VB Syntax	<code>ExportTableToFile <ReportName>, <ExportFileName>, <TableType></code>
VB Example	<code>oModule.ExportTableToFile "Plot 1", "C:/Marker.csv", "Marker"</code>

ExportToFile [Reporter]

From a data table or plot, generates text format, comma delimited, tab delimited, .dat, or .rdat type output files.

UI Access	Right-click on report name in the Project tree and select Export .			
Parameters	Name	Type	Description	
	<ReportName>	String	Name of the report	
	<FileName>	String	Path and File Name	
			Supported formats	
			.txt	Post processor format file
			.csv	Comma-delimited data file
			.tab	Tab-separated file
			.dat	Ansys plot data file
			.rdat	Ansys report data file
Return Value	None			

Python Syntax	ExportToFile (<ReportName>, <FileName>)
Python Example	<pre>oModule.ExportToFile("Plot1", "c:\report1.dat")</pre>

VB Syntax	ExportToFile <ReportName>, <FileName>
VB Example	<pre>oModule.ExportToFile "Plot1", "c:\report1.dat"</pre>

ExportUniformPointsToFile

Exports uniform points from a data table or plot report that includes the Export Uniform Points to File option enabled to text format, comma delimited, tab delimited, or .dat type output files.

File name:

erEPhi, Phi=0 plane For Phase Center Calc.csv

Save as type:

Comma delimited data files(*.csv)

Save

Cancel

Options

☒ Use Separate Columns for Curves

☒ Export Uniform Points

☒ Export with Full Sweep Range

Start:

-180

degStop:

180

degStep:

36

deg

UI Access	Right-click on report name in the project tree and select Export Data .		
Parameters	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

	<SweepRange> String Start, stop, and step range with units, for sweep. <UnitSpec> String For example, "kV, Mhz, yd" <UseTraceNumberFormat> Boolean "True", "False"
Return Value	None.

Python Syntax	<code>ExportUniformPointsToFile(<ReportName>, <FileName>, <SweepRange>)</code>
Python Example	<code>oModule.ExportUniformPointsToFile("S Parameter Table 2", "D:/MyFiles/cftt.csv", "4GHz", "5GHz", "1GHz", False, " kV, MHz, yd ", False)</code>

VB Syntax	<code>ExportUniformPointsToFile <ReportName>, <FileName>, <SweepRange></code>
VB Example	<code>oModule.ExportUniformPointsToFile "rETheta and rEPhi [dB]", "C:/MyFiles/rETheta and rEPhi [dB].csv", "-180deg", "180deg", "36deg"</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.

UI Access	NA		
Parameters	Name	Type	Description
	<ReportType>	String	Report type name.
	<DisplayType>	String	Name of display type.
	<SolutionName>	String	Name of solution.
	<SimValueCtxt>	Array	A context name, or array of strings that encode the contexts.

Return Value	Array of text strings

Python Syntax	<code>GetAllCategories(<ReportType>, <DisplayType>, <SolutionName>, <SimValueCtxt>)</code>
Python Example	<pre>oModule.GetAllCategories("Far Fields", "Rectangular Plot", "Setup1 : LastAdaptive", "Infinite Sphere1")</pre>

VB Syntax	<code>GetAllCategories <ReportType>, <DisplayType>, <SolutionName>, <SimValueCtxt></code>
VB Example	<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.

UI Access	NA		
Parameters	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"

Return Value	Array of text strings

Python Syntax	<code>GetAllQuantities(<ReportType>,<DisplayType>, <SolutionName>, <SimValueCtxt>, <CategoryName>)</code>
Python Example	<pre>oModule.GetAllQuantities("Far Fields", "Rectangular Plot", "Setup1 : LastAdaptive", "Infinite Sphere1", "Gain")</pre>

VB Syntax	<code>GetAllQuantities <ReportType>,<DisplayType>, <SolutionName>, <SimValueCtxt>, <CategoryName></code>
VB Example	<pre>oModule.GetAllQuantities _ "Far Fields", "Rectangular Plot", _ "Setup1 : LastAdaptive",) "Infinite Sphere1", "Gain"</pre>

GetAllReportNames

Gets the names of existing reports in a design

UI Access	N/A
Parameters	None.
Return Value	Array of report names

Python Syntax	GetAllReportNames()
Python Example	<code>oModule.GetAllReportNames()</code>

VB Syntax	GetAllReportNames
VB Example	<code>oModule.GetAllReportNames</code>

GetAvailableDisplayTypes

Retrieves all supported display types in report type as an array of text strings.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportType>	String	Report type name.
Return Value	Array of text strings		

Python Syntax	GetAvailableDisplayTypes(<ReportType>)
Python Example	<code>oModule.GetAvailableDisplayTypes("Far Fields")</code>

VB Syntax	GetAvailableDisplayTypes <ReportType>

VB Example	<code>oModule.GetAvailableDisplayTypes "Far Fields"</code>

GetAvailableReportTypes

Retrieves all available report types in the current Design as an array of text string.

UI Access	N/A
Parameters	None.
Return Value	Array of text strings

Python Syntax	<code>GetAvailableReportTypes()</code>
Python Example	<code>oModule.GetAvailableReportTypes()</code>

VB Syntax	<code>GetAvailableReportTypes</code>
VB Example	<code>oModule.GetAvailableReportTypes</code>

GetAvailableSolutions

Gets all available solutions in report type as an array of text strings.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportType>	String	Report type name.
Return Value	Array of text strings		

Python Syntax	GetAvailableSolutions(<ReportType>)
Python Example	<code>oModule.GetAvailableSolutions("Far Fields")</code>

VB Syntax	GetAvailableSolutions <ReportType>
VB Example	<code>oModule.GetAvailableSolutions "Far Fields"</code>

GetChildNames [Report Setup]

Gets a list of report names.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing all report names.

Python Syntax	GetChildNames()
Python Example	<code>oModule.GetChildNames()</code>

VB Syntax	GetChildNames
VB Example	<code>oModule.GetChildNames</code>

GetChildObject [Report Setup]

Gets a report object or report child object; the module's 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.

UI Access	NA		
Parameters	Name	Type	Description
	<ObjectPath>	String	Report Name or an object path beginning with a report name.
Return Value	A ReportSetup(Results) Module Child Object, [ReportSetup(Results) Module Child Objects]		

Python Syntax	GetChildObject(<ObjectPath>)
Python Example	<pre> oRpt = oRptModule.GetChildObject("S Parameter Plot 1") oTrace = oRptModule.GetChildObject("S Parameter Plot 1/dB(S(Port1,Port1))") oAxisX = oRptModule.GetChildObject("S Parameter Plot 1/AxisX") </pre>

VB Syntax	GetChildObject <ObjectPath>
VB Example	<pre> set oRpt = oRptModule.GetChildObject "S Parameter Plot 1" set oTrace = oRptModule.GetChildObject "S Parameter Plot 1/dB(S(Port1,Port1))" set oAxisX = oRptModule.GetChildObject "S Parameter Plot 1/AxisX" </pre>

GetChildTypes [ReportSetup]

Gets child types of queried Report module object.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing child object types.

Python Syntax	GetChildTypes ()
Python Example	<code>oModule.GetChildTypes ()</code>

VB Syntax	GetChildTypes
VB Example	<code>oModule.GetChildTypes</code>

GetCurvePropServerName

Gets the PropServer (the owner of the properties, or the list containing them) name of a curve.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of specified report.
	<TraceName>	String	Name of specified trace.

Return Value	Array of string containing the PropServer name.

Python Syntax	<code>GetCurvePropServerName(<ReportName>, <TraceName>)</code>
Python Example	<code>oModule.GetCurvePropServerName("Plot 1", "Phase")</code>

VB Syntax	<code>GetCurvePropServerName <ReportName>, <TraceName></code>
VB Example	<code>oModule.GetCurvePropServerName "Plot 1", "Phase"</code>

GetDisplayType

Gets the display type of a specified report.

UI Access	NA		
Parameters	Name	Type	Description
	<ReportName>	String	Report name
Return Value	String containing display type.		

Python Syntax	<code>GetDisplayType(<ReportName>)</code>
Python Example	<code>oModule.GetDisplayType("Design Plot 1")</code>

VB Syntax	GetDisplayType <ReportName>
VB Example	oModule.GetDisplayType "Design Plot 1"

GetDynLinkIntrinsicVariables

Gets variable names from a trace included in dynamic link outputs.

UI Access	N/A		
Parameters	Name	Type	Description
	<TraceName>	String	Trace name with its report name.
Return Value	Array of variable names		

Python Syntax	GetDynLinkIntrinsicVariables(<TraceName>)
Python Example	oModule.GetDynLinkIntrinsicVariables("Plot 1:Trace")

VB Syntax	GetDynLinkIntrinsicVariables <TraceName>
VB Example	oModule.GetDynLinkIntrinsicVariables "Plot 1:Trace")

GetDynLinkQtyValueState

Gets the state of the quantity values from a source dynamic linked trace.

UI Access	N/A

Parameters	Name	Type	Description
	<TraceName>	String	Name of specified source trace.
	<QtyName>	String	Name of specified quantity value.
Return Value	Array of strings containing states.		

Python Syntax	GetDynLinkQtyValueState(<TraceName>, <QtyName>)
Python Example	<code>oModule.GetDynLinkQtyValueState("Plot 1:Trace1", "")</code>

VB Syntax	GetDynLinkQtyValueState <TraceName>, <QtyName>
VB Example	<code>oModule.GetDynLinkQtyValueState "Plot 1:Trace1", ""</code>

GetDynLinkTraces

Gets the names of the dynamic linked traces.

UI Access	N/A		
Parameters	Name	Type	Description
	<SoluName>	String	Name of the source solution. If empty, refer to current solution.
Return Value	Array of strings containing trace names.		

Python Syntax	GetDynLinkTraces(<SoluName>)
Python Example	<code>oModule.GetDynLinkTraces("")</code>

VB Syntax	GetDynLinkTraces <SoluName>
VB Example	oModule.GetDynLinkTraces ""

GetDynLinkVariableValues

Gets the values of a variable from a dynamic linked trace.

UI Access	N/A		
Parameters	Name	Type	Description
	<TraceName>	String	Name of specified dynamic linked trace, with its report name.
	<VarName>	String	Name of specified variable.
Return Value	Array of strings containing variable values.		

Python Syntax	GetDynLinkVariableValues(<TraceName>, <VarName>)
Python Example	oModule.GetDynLinkVariableValues("Plot 1:Trace", "Var1")

VB Syntax	GetDynLinkVariableValues <TraceName>, <VarName>
VB Example	oModule.GetDynLinkVariableValues "Plot 1:Trace", "Var1"

GetName

Returns the design ID and design name of the active design, in that order separated by a semicolon. If the active design is a sub-circuit, it also returns the parent ID and parent design name, listed before the active design information.

To get the *only* the design's name without the parent design name or design IDs, see `oDesign.GetDesignName()`.

UI Access	N/A
Parameters	None.
Return Value	String indicating the name and ID of the active design and parent design (if the active design is a sub-circuit).

Python Syntax	<code>GetName()</code>
Python Example	<pre>oDesign.GetName() for a subcircuit <i>EMDesign2</i> in a parent design <i>EMDesign1</i> design_name = oDesign.GetName() 1;EMDesign1/2;EMDesign2</pre> In the returned string: 1 is the parent design ID; <i>EMDesign1</i> is the parent design name. 2 is the active design ID; <i>EMDesign2</i> is the active design name.

VB Syntax	<code>GetName</code>
VB Example	<code>design_name = oDesign.GetName</code>

GetObjPath [Design]

Obtains the path to the design.

UI Access	N/A
Parameters	None.
Return Value	String containing the path to the design.

Python Syntax	GetObjPath()
Python Example	<code>oDesign.GetObjPath()</code>

VB Syntax	GetObjPath
VB Example	<code>oDesign.GetObjPath</code>

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.

UI Access	N/A		
Parameters	Name	Type	Description
	<PropTab>	String	One of the following, where tab titles are shown in parentheses: - PassedParameterTab ("Parameter Values") - DefinitionParameterTab (Parameter Defaults") - LocalVariableTab ("Variables" or "Local Variables") - ProjectVariableTab ("Project variables") - ConstantsTab ("Constants") - BaseElementTab ("Symbol" or "Footprint") - ComponentTab ("General") - Component("Component") - CustomTab ("Intrinsic Variables") - Quantities ("Quantities") - Signals ("Signals")
	<PropServer>	String	An object identifier, generally returned from another script method, such as <code>CompInst@R;2;3</code>
	<PropName>	String	Name of the property.
Return Value	String value of the property.		

Python Syntax	GetPropertyValue (<PropTab>, <PropServer>, <PropName>)
Python Example	<pre> selectionArray = oEditor.GetSelections() for k in selectionArray: val = oEditor.GetPropertyValue("PassedParameterTab", k, "R") ... </pre>

VB Syntax	GetPropertyValue (<PropTab>, <PropServer>, <PropName>)
VB Example	<pre>selectionArray = oEditor.GetSelections for k in selectionArray: val = oEditor.GetPropertyValue("PassedParameterTab", k, "R") ...</pre>

GetPropNames [Reporter]

Report setup module does not have its own property, this function always returns empty array.

UI Access	N/A
Parameters	None.
Return Value	Empty array.

Python Syntax	GetPropNames()
Python Example	<code>oModule.GetPropNames()</code>

VB Syntax	GetPropNames

VB Example	<code>oModule.GetPropNames</code>

GetPropValue [Report Setup]

Gets the property value for a Report, or reports' child object.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><PropPath></code>	String	A child object's property path. See property path discussion here .
Return Value	String containing the property value.		

Python Syntax	<code>GetPropValue(<PropPath>)</code>
Python Example	<code>oModule.GetPropValue("S Parameter Plot 1/Display Type")</code>

VB Syntax	<code>GetPropValue <PropPath></code>
VB Example	<code>oModule.GetPropValue "S Parameter Plot 1/Display Type"</code>

GetQtyExpressionsForSourceTrace

Gets the quantity expressions from a specified source trace.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><SourceTraceName></code>	String	Name of specified source trace.

Return Value	Array of strings containing quantity expressions.

Python Syntax	<code>GetQtyExpressionsForSourceTrace(<SourceTraceName>)</code>
Python Example	<code>oModule.GetQtyExpressionsForSourceTrace("Plot 1:Trace1")</code>

VB Syntax	<code>GetQtyExpressionsForSourceTrace <SourceTraceName></code>
VB Example	<code>oModule.GetQtyExpressionsForSourceTrace "Plot 1:Trace1"</code>

GetReportTraceNames

Gets the names of existing trace names in a plot.

UI Access	N/A		
Parameters	Name	Type	Description
	<PlotName>	String	Name of specified plot.
Return Value	Array of strings containing trace names.		

Python Syntax	<code>GetReportTraceNames(<PlotName>)</code>
Python Example	<code>oModule.GetReportTraceNames("SParameter Plot 1")</code>

VB Syntax	<code>GetReportTraceNames <PlotName></code>
VB Example	<code>oModule.GetReportTraceNames "SParameter Plot 1"</code>

GetReportSummaryForRegressionTesting

Gets report summary from a dumped report file.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of a specified dumped report.
Return Value	String containing report summary.		

Python Syntax	<code>GetReportSummaryForRegressionTesting(<ReportName>)</code>
Python Example	<code>oModule.GetReportSummaryForRegressionTesting("C:/report.rdat")</code>

VB Syntax	<code>GetReportSummaryForRegressionTesting <ReportName></code>
VB Example	<code>oModule.GetReportSummaryForRegressionTesting "C:/report.rdat"</code>

GetSolutionContexts

Gets all available solution context names in a solution as an array of text strings.

UI Access	N/A		
Parameters	Name	Type	Description

	<i><ReportType></i>	String	Report type name.
	<i><DisplayType></i>	String	Display type name.
	<i><SolutionName></i>	String	Name of solution.
Return Value	Array of text strings		

Python Syntax	<code>GetSolutionContexts(<ReportType>, <DisplayType>, <SolutionName>)</code>
Python Example	<pre>oModule.GetSolutionContexts("Far Fields", "Rectangular Plot", "Setup1:LastAdaptive")</pre>

VB Syntax	<code>GetSolutionContexts <ReportType>, <DisplayType>, <SolutionName></code>
VB Example	<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.

UI Access	N/A		
Parameters	Name	Type	Description
	<i><reportTypeArg></i>	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	Same as FamiliesArray values created in the relevant CreateReport script.
	<expressionArg>	String or Array of Strings	Text string or array of text strings; valid expression, may validate it as the data-table Y-component.
Return Value	ARRAY of ISolutionDataResultComInterface objects, containing: • GetSweepNames() • GetSweepUnits() • GetSweepValues() • IsPerQuantityPrimarySweep() • GetDataExpressions() – should be the same as <expressionArg> • GetPerQuantityPrimarySweepValues() • IsDataComplex() • GetDataUnits() • GetRealDataValues() • GetImagDataValues() • ReleaseData() • GetDesignVariableNames() • GetDesignVariableUnits() • GetDesignVariableValue() • GetDesignVariationKey()		

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].

Python Syntax	GetSolutionDataPerVariation(reportTypeArg, solutionNameArg, simValueCtxtArg, familiesArg, expressionArg)
Python Example	<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>

VB Syntax	GetSolutionDataPerVariation(reportTypeArg, solutionNameArg, simValueCtxtArg, familiesArg, expressionArg)
VB Example	<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))")
)
```

GetDataExpressions

Returns data expressions.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A
Parameters	N/A
Return Value	Array of text strings

Python Syntax	GetDataExpressions()
Python Example	<pre>expressions = oModule.GetDataExpressions()</pre>

VB Syntax	GetDataExpressions()
VB Example	<pre>dim expressions expressions = oModule.GetDataExpressions()</pre>

GetDataUnits

Returns text string containing units.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<expressionString>	String	Can be returned from GetDataExpressions()
Return Value	Text string of units; empty if no units		

Python Syntax	GetDataUnits(<expressionString>)
Python Example	<code>oModule.GetDataUnits(expressions)</code>

VB Syntax	GetDataUnits(<expressionString>)
VB Example	<code>oModule.GetDataUnits(expressions)</code>

GetDesignVariableNames

Returns array of strings containing design variable names.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	NA
Parameters	NA
Return Value	Array of strings

Python Syntax	GetDesignVariableNames()
Python Example	<pre>names = oModule.GetDesignVariableNames()</pre>

VB Syntax	GetDesignVariableNames()
VB Example	<pre>dim names names = oModule.GetDesignVariableNames()</pre>

GetDesignVariableUnits

Returns array of strings containing design variable units.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<varName>	String	Can be returned from GetDesignVariableNames()
Return Value	Text string of units; empty if no units.		

Python Syntax	GetDesignVariableUnits(<varName>)
Python Example	<pre>units = oModule.GetDesignVariableUnits('Variable Name')</pre>

VB Syntax	GetDesignVariableUnits(<varName>)
VB Example	<pre>dim units units = oModule.GetDesignVariableUnits("Variable Name")</pre>

GetDesignVariableValue

Returns a design variable's value.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<varName>	String	Can be returned from GetDesignVariableNames()
	<siValue>	Boolean	True to return SI-value; False to return by GetDesignVariableUnits()
Return Value	Double value		

Python Syntax	GetDesignVariableValue(<varName>, <siValue>)
Python Example	<pre>value = oModule.GetDesignVariableValue('varName',1)</pre>

VB Syntax	GetDesignVariableValue(<varName>, <siValue>)
VB Example	<pre>dim value value = oModule.GetDesignVariableValue("varName", True)</pre>

GetDesignVariationKey

Returns a design's Variation Key.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A
Parameters	N/A
Return Value	String containing variation key.

Python Syntax	GetDesignVariationKey()
Python Example	<code>oModule.GetDesignVariationKey()</code>

VB Syntax	GetDesignVariationKey()
VB Example	<pre>dim key set key = oModule.GetDesignVariationKey()</pre>

GetImagDataValues

Returns array of imaginary data values.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<expressionString>	String	Can be returned from GetDataExpressions()
	<siValue>	Boolean	True to return SI-value; False to return with units returned in GetSweepUnits() .
Return Value	Array of doubles		

Python Syntax	GetImagDataValues(<expressionString>,<siValue>)
Python Example	<pre>imaginaryvalues = oModule.GetImagDataValues('expression',1)</pre>

VB Syntax	GetImagDataValues(<expressionString>,<siValue>)
VB Example	<pre>dim imaginaryvalues imaginaryvalues = oModule.GetImagDataValues("expression",True)</pre>

GetPerQuantityPrimarySweepValues

Returns per quantity primary sweep values.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<expressionString>	String	Can be returned from GetDataExpressions() .
	<siValue>	Boolean	True to return SI-value; False to return by GetSweepUnits() .
Return Value	Array of doubles if IsPerQuantityPrimarySweep() returned True; error if returned False		

Python Syntax	GetPerQuantitySweepValues(<expressionString>, <siValue>)
Python Example	<pre>sweepvalues = oModule.GetPerQuantitySweepValues('0.111,0.201,0.345,0.231', 1)</pre>

VB Syntax	GetPerQuantitySweepValues(<expressionString>, <siValue>)
VB Example	<pre>dim sweepvalues sweepvalues = oModule.GetPerQuantitySweepValues("0.111,0.201,0.345,0.231", True)</pre>

GetRealDataValues

Returns array of real data values.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<expressionString>	String	Can be returned from GetDataExpressions()
	<siValue>	Boolean	True to return SI-value; False to return with units returned in GetSweepUnits() .
Return Value	Array of doubles		

Python Syntax	GetRealDataValues(<expressionString>,<siValue>)
Python Example	<pre>realvalues = oModule.GetRealDataValues('expression',1)</pre>

VB Syntax	GetRealDataValues(<expressionString>,<siValue>)
VB Example	<pre>dim realvalues realvalues = oModule.GetRealDataValues("expression",True)</pre>

GetSweepNames

Returns array of text strings containing primary sweep name(s).

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A
Parameters	N/A
Return Value	Array of text strings

Python Syntax	GetSweepNames()
Python Example	<pre>sweepnames = arr[0].GetSweepNames()</pre>

VB Syntax	GetSweepNames()
VB Example	<pre>dim sweepnames sweepnames = arr[0].GetSweepNames()</pre>

GetSweepUnits

Returns text string containing units.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<sweepName>	String	Primary sweep name
Return Value	Text string containing units		

Python Syntax	GetSweepUnits(<sweepName>)
Python Example	<pre>sweepunits = oModule.GetSweepUnits('Sweep 1')</pre>

VB Syntax	GetSweepUnits(<sweepName>)
VB Example	<pre>dim sweepunits sweepunits = oModule.GetSweepUnits("Sweep 1")</pre>

GetSweepValues

Returns sweep values.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A		
Parameters	Name	Type	Description
	<sweepName>	String	Primary sweep name
	<siValue>	Boolean	True to return SI-value; False to return by GetSweepUnits() .
Return Value	Array of doubles		

Python Syntax	GetSweepValues(<sweepName>, <siValue>)
Python Example	<pre>sweepvalues = oModule.GetSweepValues('Sweep 1', True)</pre>

VB Syntax	GetSweepValues(<sweepName>, <siValue>)
VB Example	<pre>dim sweepvalues sweepvalues = oModule.GetSweepValues("Sweep 1", True)</pre>

IsDataComplex

Returns whether an expression is complex.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	NA		
Parameters	Name	Type	Description
	<expressionString>	String	Can be returned from GetDataExpressions() .
Return Value	Boolean (True if expression is Complex data; False if not)		

Python Syntax	IsDataComplex(<expressionString>)
Python Example	<code>oModule.IsDataComplex('.001,.234,.455,.434')</code>

VB Syntax	IsDataComplex(<expressionString>)
VB Example	<code>oModule.IsDataComplex('.001,.234,.455,.434')</code>

IsPerQuantityPrimarySweep

Returns whether data expressions have different primary sweep values.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	N/A
Parameters	N/A
Return Value	Boolean (True if data expressions have different primary sweep values)

Python Syntax	IsPerQuantityPrimarySweep()
Python Example	<pre>var = oModule.IsPerQuantityPrimarySweep()</pre>

VB Syntax	IsPerQuantityPrimarySweep()
VB Example	<pre>dim var var = oModule.IsPerQuantityPrimarySweep()</pre>

Release Data

Releases all cached data. After this function is called, all subsequent function calls to the object will fail.

Important:

This is a member function of ISolutionDataResultComInterface object, which is the element of the returned array from function [GetSolutionDataPerVariation](#).

UI Access	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>

GroupPlotCurvesByGroupingStrategy

Groups curves in a Stacked Plot automatically based on a curve grouping strategy.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<GroupStrategy>	String	Strategy for grouping, "Single", "By Trace" or "By Units".

Return Value	None.

Python Syntax	GroupPlotCurvesByGroupingStrategy(<ReportName>, <GroupStrategy>)
Python Example	oModule.GroupPlotCurvesByGroupingStrategy("Transient Plot 1", "By Trace")

VB Syntax	GroupPlotCurvesByGroupingStrategy <ReportName>, <GroupStrategy>
VB Example	oModule.GroupPlotCurvesByGroupingStrategy "Transient Plot 1", "By Trace"

ImportIntoReport

Imports .tab, .csv, and .dat format files into a report.

UI Access	Right-click on report name in the Project tree and select Import....			
Parameters	Name	Type	Description	
	<ReportName>	String	Name of the Report	
	<FileName>	String	Path and File Name	
			.csv	Comma-delimited data file
			.tab	Tab-separated file
			.dat	Ansys plot data file
Return Value	None			

Python Syntax	ImportIntoReport (<ReportName>, <FileName>)
Python Example	oDesign.ImportIntoReport ("Plot1", "c:\report1.dat")

VB Syntax	ImportIntoReport <ReportName>, <FileName>
VB Example	oDesign.ImportIntoReport "Plot1", "c:\report1.dat"

ImportReportDataIntoReport

Imports report data from a file into a specified report.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of specified report.
	<FileName>	String	Name of specified data file. File extension "rdat" is expected.
Return Value	None.		

Python Syntax	ImportReportDataIntoReport(<ReportName>, <FileName>)
Python Example	oModule.ImportReportDataIntoReport("Plot 1", "C:/Plot1data.rdat")

VB Syntax	ImportReportDataIntoReport <ReportName>, <FileName>
VB Example	oModule.ImportReportDataIntoReport "Plot 1", "C:/Plot1data.rdat"

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.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<CurveArray>	Array	Array of curve names to move.
	<StackName>	String	Name of stack to move to.
Return Value	None.		

Python Syntax	MovePlotCurvesToGroup(<ReportName>, <CurveArray>, <StackName>)
Python Example	<pre>oModule.MovePlotCurvesToGroup("XY Stacked Plot 1", ["R2.V : TR", "R2.I : TR"], "Stack 2")</pre>

VB Syntax	MovePlotCurvesToGroup <ReportName>, <CurveArray>, <StackName>
VB Example	<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.

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<CurveArray>	Array	Array of curve names to move.
Return Value	None.		

Python Syntax	MovePlotCurvesToNewGroup (<ReportName>, <CurveArray>)		
Python Example	<pre>oModule.MovePlotCurvesToNewGroup ("XY Stacked Plot 1", ["R2.V : TR", "R2.I : TR"])</pre>		

VB Syntax	MovePlotCurvesToNewGroup <ReportName>, <CurveArray>		
VB Example	<pre>oModule.MovePlotCurvesToNewGroup _ "XY Stacked Plot 1", _ Array("R2.V : TR", "R2.I : TR")</pre>		

OpenWindowForAllReports

Opens windows for all reports belong to current design.

UI Access	N/A
Parameters	None.
Return Value	None.

Python Syntax	OpenWindowForAllReports()
Python Example	<code>oModule.OpenWindowForAllReports()</code>

VB Syntax	OpenWindowForAllReports
VB Example	<code>oModule.OpenWindowForAllReports</code>

OpenWindowForReports

Opens windows for specified reports.

UI Access	N/A		
Parameters	Name	Type	Description
	<i><ReportNames></i>	Array	Array of strings containing report names.
Return Value	None.		

Python Syntax	OpenWindowForReports(<i><ReportNames></i>)

Python Example	<code>oModule.OpenWindowForReports(["Report1", "Report2"])</code>

VB Syntax	<code>OpenWindowForReports <ReportNames></code>
VB Example	<code>oModule.OpenWindowForReports Array("Report1", "Report2")</code>

PastePlotSettings

Paste plot settings to a specified report.

UI Access	Right-click a report, select Paste		
Parameters	Name	Type	Description
	<ReportName>	String	Name of specified report.
	<PropTypeToApply>	String	Property type to paste. "Graphical", "Data", or "All".
Return Value	None.		

Python Syntax	<code>PastePlotSettings(<ReportName>, <PropTypeToApply>)</code>
Python Example	<code>oModule.PastePlotSettings("Plot 1", "Graphical")</code>

VB Syntax	<code>PastePlotSettings <ReportName>, <PropTypeToApply></code>
VB Example	<code>oModule.PastePlotSettings "Plot 1", "Graphical"</code>

PasteReports

Paste copied reports to results in the current project.

UI Access	Edit > Paste
Parameters	None.
Return Value	None.

Python Syntax	PasteReports ()
Python Example	<code>oModule.PasteReports ()</code>

VB Syntax	PasteReports
VB Example	<code>oModule.PasteReports</code>

PasteReportsWithLegacyNames

Pastes copied reports to results in the current project with legacy name definitions.

UI Access	N/A
Parameters	None.
Return Value	None.

Python Syntax	PasteReportsWithLegacyNames ()
Python Example	<code>oModule.PasteReportsWithLegacyNames ()</code>

VB Syntax	PasteReportsWithLegacyNames
VB Example	<code>oModule.PasteReportsWithLegacyNames</code>

PasteTraces

Pastes copied traces to a named plot.

UI Access	Paste		
Parameters	Name	Type	Description
	<ReportName>	String	Name of plot
Return Value	None		

Python Syntax	PasteTraces (<ReportName>)
Python Example	<code>oModule.PasteTraces ("XY Plot1")</code>

VB Syntax	PasteTraces <ReportName>

VB Example	<code>oModule.PasteTraces "XY Plot1"</code>

PasteTracesWithLegacyNames

Pastes copied traces to a named plot using legacy name definitions.

UI Access	N/A		
Parameters	Name	Type	Description
	<i><ReportName></i>	String	Name of plot
Return Value	None		

Python Syntax	<code>PasteTracesWithLegacyNames (<ReportName>)</code>
Python Example	<code>oModule.PasteTracesWithLegacyNames ("XY Plot1")</code>

VB Syntax	<code>PasteTracesWithLegacyNames <ReportName></code>
VB Example	<code>oModule.PasteTracesWithLegacyNames "XY Plot1"</code>

RenameReport

Renames an existing report.

UI Access	Select a report on the Project tree, right-click and select Rename

Parameters	Name	Type	Description
	<OldReportName>	String	Old Report Name
	<NewReportName>	String	New Report Name
Return Value	None.		

Python Syntax	RenameReport (<OldReportName>, <NewReportName>)
Python Example	<code>oModule.RenameReport("XY Plot1", "Reflection")</code>

VB Syntax	RenameReport <OldReportName>, <NewReportName>
VB Example	<code>oModule.RenameReport "XY Plot1", "Reflection"</code>

RenameTrace

To rename a trace in a plot

UI Access	N/A		
Parameters	Name	Type	Description
	<ReportName>	String	Name of report.
	<TraceName>	String	Name of Trace
	<NewName>	String	New trace name.
Return Value	None.		

Python Syntax	<code>RenameTrace(<ReportName>, <TraceName>, <NewName>)</code>
Python Example	<pre>oModule.RenameTrace ("XY Plot1", "dB(S(WavePort1,WavePort1))1", "Port1dbS")</pre>

VB Syntax	<code>RenameTrace <ReportName>, <TraceName>, <NewName></code>
VB Example	<pre>oModule.RenameTrace "XY Plot1", _ "dB(S(WavePort1,WavePort1))1", _ "Port1dbS"</pre>

ResetPlotSettings

Resets plot settings to defaults.

UI Access	Right-click on a plot, select Edit > Reset Plot Settings		
Parameters	Name	Type	Description
	<PlotName>	String	Name of specified plot.
Return Value	None.		

Python Syntax	<code>ResetPlotSettings(<PlotName>)</code>
Python Example	<pre>oModule.ResetPlotSettings("Differential S-parameters")</pre>

VB Syntax	ResetPlotSettings <PlotName>
VB Example	<code>oModule.ResetPlotSettings("Differential S-parameters")</code>

SavePlotSettingsAsDefault

Saves report plot settings as default.

UI Access	Report Templates > Save Settings as Default		
Parameters	Name	Type	Description
	<PlotName>	String	Name of plot to use for plot defaults.
Return Value	None.		

Python Syntax	SavePlotSettingsAsDefault("<PlotName>")
Python Example	<code>oModule.SavePlotSettingsAsDefault ("XY Plot1")</code>

VB Syntax	SavePlotSettingsAsDefault "<PlotName>"
VB Example	<code>oModule.SavePlotSettingsAsDefault "XY Plot 1"</code>

SetLinkOutputTraces

Specifies dynamic link output traces from the current design.

UI Access	Right-click the Results , select Link Output...		
Parameters	Name	Type	Description

	<i><TraceArray></i>	Array	Array of traces to set. Array("<ReportName>:=", <array of trace names>, "<ReportName>:=", <array of trace names>,...)
Return Value	None.		

Python Syntax	SetLinkOutputTraces(<i><TraceArray></i>)		
Python Example	<pre>oModule.SetLinkOutputTraces(["Plot 1:=", ["Trace1"], "Plot 2:=", ["Trace1"]])</pre>		

VB Syntax	SetLinkOutputTraces <i><TraceArray></i>		
VB Example	<pre>oModule.SetLinkOutputTraces _ Array("Plot 1:=", Array("Trace1"), _ "Plot 2:=", Array("Trace1"))</pre>		

SetPropValue [Report Setup]

Sets the property value for report module child object.

UI Access	Select Edit Properties on Report objects.		
Parameters	Name	Type	Description
	<PropPath>	String	A child object's property path. See property path discussion here .
	<NewValue>	String, Number, or Boolean	New value data type is depending on the property type,
Return Value	True if the property is found and the new value is valid. Otherwise return False.		

Python Syntax	SetPropValue(<PropPath>, <NewValue>)		
Python Example	oRptModule.SetPropValue("S Parameter Plot 1/Display Type", "DataTable")		
	oRptModule.SetPropValue("S Parameter Plot 1/db(S(port1,port2)/Primary sweep", "Freq")		

VB Syntax	SetPropValue <PropPath>, <NewValue>		
VB Example	oRptModule.SetPropValue "S Parameter Plot 1/Display Type", "DataTable"		
	oRptModule.SetPropValue "S Parameter Plot 1/db(S(port1,port2)/Primary sweep", "Freq"		

UnGroupPlotCurvesInGroup

From a Stacked Plot, ungroups curves in a stack.

UI Access	N/A		
Parameters	Name	Type	Description

	<ReportName>	String	Name of report.
	<GroupName>	String	Stack group name.
Return Value	None.		

Python Syntax	UnGroupPlotCurvesInGroup(<ReportName>, <GroupName>)		
Python Example	oModule.UngroupPlotCurvesInGroup("S Parameter Plot 3", "Stack 1")		

VB Syntax	UnGroupPlotCurvesInGroup <ReportName>, <GroupName>		
VB Example	oModule.UngroupPlotCurvesInGroup "S Parameter Plot 3", "Stack 1"		

UpdateAllReports

Updates all reports in the **Results** branch in the project tree.

UI Access	Right-click on Results in the project tree, select Update All Reports
Parameters	None
Return Value	None

Python Syntax	UpdateAllReports()
Python Example	oModule.UpdateAllReports()

VB Syntax	UpdateAllReports
VB Example	<code>oModule.UpdateAllReports</code>

UpdateReports

Updates specified reports.

UI Access	N/A		
Parameters	Name	Type	Description
	<i><ReportNames></i>	Array	Array of strings containing report names.
Return Value	None.		

Python Syntax	UpdateReports(<i><ReportNames></i>)
Python Example	<code>oModule.UpdateReports (["XY Plot 1", "XY Plot 4"])</code>

VB Syntax	UpdateReports <i><ReportNames></i>
VB Example	<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 Report dialog, click Apply Traces 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. <pre>Array("Domain:=", <DomainType>) <DomainType> ex. "Sweep" or "Time" Array("Context:=", <GeometryType>) <GeometryType> ex. "Infinite Spheren", "Spheren", "Polylinen"</pre>
	<FamiliesArray>	Array	Contains sweep definitions for the report. <pre>Array("<VariableName>:= ", <ValueArray>) <ValueArray> Array("All") or Array("Value1", "Value2", ... "Valuen") examples of <VariableName> "Freq", "Theta", "Distance"</pre>
	<ReportDataArray>	Array	This array contains the report quantity and X, Y, and (Z) axis definitions. <pre>Array("X Component:=", <VariableName>, "Y Component:=", <VariableName> <ReportQuantityArray>) <ReportQuantityArray> ex. Array("dB(S(Port1, Port1))")</pre>
	<ExtTraceInfo>	Array	Optional. Array defines extended trace information.

Return Value	None.

Python Syntax	<code>UpdateTraces(<ReportName>, <SolutionName>, <ContextArray>, <FamiliesArray>, <ReportDataArray>)</code>
Python Example	<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>

VB Syntax	UpdateTraces <ReportName>, <SolutionName>, <ContextArray>, <FamiliesArray>, <ReportDataArray>
VB Example	<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.

UI Access	Modify Report with multiple traces selected.		
Parameters	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 Modify Report dialog box. 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. ex. "Sweep" or "Time"

	<code><PointSet></code>	Array	Point set for the selected traces, for example, X and Y values for the plot.
Return Value	None		

Python Syntax	<code>UpdateTracesContextAndSweeps(<ReportName>, <TraceNames>, <SolutionName>, <ContextArray>, <PointSet>)</code>
Python Example	<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>

VB Syntax	<code>UpdateTracesContextAndSweeps <ReportName>, <TraceNames>, <SolutionName>, <ContextArray>, <PointSet></code>

VB Example	<pre>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"))</pre>

11 - 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.

The commands in this section include:

[CopySetup](#)

[DeleteSetups \[Optimetrics\]](#)

[DistributedAnalyzeSetup](#)

[EditSetup](#)

[EditSetup \[Parametric\]](#)

[EditSetup \[Optimization\]](#)

[EditSetup \[Sensitivity\]](#)

[EditSetup \[Statistical\]](#)

[EnableSetup](#)

[ExportDXConfigFile](#)

[ExportOptimetricsProfile](#)

[ExportOptimetricsResult](#)

[ExportParametricResults](#)

[ExportParametricSetupTable](#)

[ExportRespSurfaceMinMaxTable](#)

[ExportRespSurfaceRefinePoints](#)

[ExportRespSurfaceResponsePoints](#)

[ExportRespSurfaceVerificationPoints](#)

[GenerateVariationData \[Parametric\]](#)

[GetChildNames \[Optimetrics\]](#)

[GetChildObject \[Optimetrics\]](#)

[GetChildTypes \[Optimetrics\]](#)

[GetName](#)

[GetObjPath \[Editor\]](#)

[GetOptimetricResult](#)

[GetPropNames \[Optimetrics\]](#)

[GetPropValue \[Optimetrics\]](#)

[GetSetupNames \[Optimetrics\]](#)

[GetSetupNamesByType \[Optimetrics\]](#)

[ImportSetup](#)

[InsertSetup](#)

[InsertSetup \[Parametric\]](#)

[InsertSetup \[Optimization\]](#)

[InsertSetup \[Sensitivity\]](#)[InsertSetup \[Statistical\]](#)[PasteSetup \[Optimetrics\]](#)[RenameSetup \[Optimetrics\]](#)[SetPropValue \[Optimetrics\]](#)[SolveAllSetup](#)[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](#)

CopySetup

Copy the specified Optimetrics setup.

UI Access	NA		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup.
Return Value	None.		

Python Syntax	<code>CopySetup (<SetupName>)</code>
Python Example	<code>oModule.CopySetup ("OptimizationSetup1")</code>

VB Syntax	<code>CopySetup <SetupName></code>
VB Example	<code>oModule.CopySetup "OptimizationSetup1"</code>

DeleteSetups [Optimetrics]

Deletes the specified Optimetrics setups.

UI Access	Right-click the setup in the project tree, and then click Delete on the shortcut menu		
Parameters	Name	Type	Description
	<NameArray>	Array of Strings	An Array of Setup Names
Return Value	None		

Python Syntax	<code>DeleteSetups (<NameArray>)</code>
Python Example	<code>oModule.DeleteSetups (["OptimizationSetup1"])</code>

VB Syntax	<code>DeleteSetups <NameArray></code>
VB Example	<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

UI Access	Right-click the parametric setup name in the project tree and select Distribute Analysis.		
Parameters	Name	Type	Description
	<ParametricSetupName>	String	Name of the Setup
	isBlocking	Boolean	An optional arg that defaults to <code>true</code> . If it's <code>false</code> , the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.
Return Value	None		

Python Syntax	DistributedAnalyzeSetup (<ParametricSetupName>)
Python Example	<code>oModule.DistributedAnalyzeSetup ("ParametricSetup1")</code>

VB Syntax	DistributedAnalyzeSetup <ParametricSetupName>
VB Example	<code>oModule.DistributedAnalyzeSetup "ParametricSetup1"</code>

EnableSetup

Enables and disables a defined optimetrics analysis setup.

UI Access	Right-click on a setup in the project tree, select Enable Setup or Disable Setup		
Parameters	Name	Type	Description
	<SetupName>	String	Name of specified setup.
	<Enable>	Boolean	Determines whether enable or disable a setup. • True - enable setup. • False - disable setup.
Return Value	None.		

Python Syntax	EnableSetup(<SetupName>, <Enable>)
Python Example	<code>oModule.EnableSetup("OptimizationSetup1", True)</code>

VB Syntax	EnableSetup <SetupName>, <Enable>
VB Example	<code>oModule.EnableSetup "OptimizationSetup1", true</code>

ExportDXConfigFile

Create an xml file with the setup information for Design Xplorer

UI Access	Right click on the Design Xplorer setup in the project tree and choose Export External Connector Addin Configuration...		
Parameters	Name	Type	Description
	<SetupName>	String	Must be one of existing DesignExplorer setup names

	<FileName>	String	Must be a valid file path and name
Return Value	None		

Python Syntax	ExportDXConfigFile (<SetupName>, <FileName>)		
Python Example	<pre>oModule.ExportDXConfigFile ("DesignXplorerSetup1", "c:/exportdir/DXSetup1.xml")</pre>		

VB Syntax	ExportDXConfigFile <SetupName>, <FileName>		
VB Example	<pre>oModule.ExportDXConfigFile ("DesignXplorerSetup1", "c:/exportdir/DXSetup1.xml")</pre>		

ExportOptimetricsProfile

Export Optimetrics profile data

UI Access	Right click on the Optimetrics setup in the project tree and choose View Analysis Result... On the Post Analysis Display dialog box, click the Profile tab and click on the Export button.		
Parameters	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.

Return Value	None

Python Syntax	ExportOptimetricsProfile (<SetupName>, <FileName>, [profileNum]
Python Example	<pre>oModule.ExportOptimetricsProfile ("StatisticalSetup1", "c:/exportdir/test.csv")</pre>

VB Syntax	ExportOptimetricsProfile <SetupName>, <FileName>, [profileNum]
VB Example	<pre>oModule.ExportOptimetricsProfile "StatisticalSetup1", "c:/exportdir/test.csv"</pre>

ExportOptimetricsResult

Export an existing Optimization, Sensitivity, Statistical or DesignXplorer result. (Does not export Parametric results.)

UI Access	Right click on the desired Optimetrics setup in the project tree and choose View Analysis Result... On the Post Analysis Display dialog box, click the Result tab, then select Table view, and click on the Export button		
Parameters	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.

Return Value	None

Python Syntax	ExportOptimetricsResult (<SetupName>, <FileName>, [useFullOutputName])
Python Example	<pre>oModule.ExportOptimetricsResult ("StatisticalSetup1", "c:/exportdir/test.csv", false)</pre>

VB Syntax	ExportOptimetricsResult <SetupName>, <FileName>, [useFullOutputName]
VB Example	<pre>oModule.ExportOptimetricsResult "StatisticalSetup1", "c:/exportdir/test.csv", false</pre>

ExportParametricResults

Export existing Parametric results.

UI Access	Right click on the desired Parametric setup in the project tree and choose View Analysis Result... On the Post Analysis Display dialog box, click the Result tab, then select Table view, and click on the Export button.		
Parameters	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
Return Value	None		

Python Syntax	ExportParametricResults (<SetupName>, <FileName>, <bOutputUnits>)

Python Example	<pre>oModule.ExportParametricResults ("ParametricSetup1", "c:/exportdir/test.csv", False)</pre>

VB Syntax	<code>ExportParametricResults <SetupName>, <FileName>, <bOutputUnits></code>
VB Example	<pre>oModule.ExportParametricResults "ParametricSetup1", "c:/exportdir/test.csv", false</pre>

ExportParametricSetupTable

Exports the parametric setup table as a CSV file.

UI Access	Double-click parametric setup. Select Table tab. Click Export .		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup.
	<filePath>	String	Full path for file export.
Return Value	None		

Python Syntax	<code>ExportParametricSetupTable (<SetupName>, <filePath>)</code>
Python Example	<pre>oModule.ExportParametricSetupTable('ParametricSetup1', 'E:/Files/ParametricSetup1_ Table.csv')</pre>

VB Syntax	ExportParametricSetupTable <SetupName>, <filePath>
VB Example	obj.ExportParametricSetupTable "ParametricSetup1", "E:/Files/ParametricSetup1_Table.csv"

ExportRespSurfaceMinMaxTable

Exports min-max table from a response surface to a file

UI Access	Click on Export... From the Response Surface tab of the Design of Experiments Post Analysis Display dialog.		
Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.
Return Value	None.		

Python Syntax	ExportRespSurfaceMinMaxTable(<DOEName>, <FileName>)
Python Example	oModule.ExportRespSurfaceMinMaxTable("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Min-Max_Search.csv")

VB Syntax	ExportRespSurfaceMinMaxTable <DOEName>, <FileName>
VB Example	oModule.ExportRespSurfaceMinMaxTable _

	"DesignOfExperimentsSetup1", _ "C:/temp/DesignOfExperimentsSetup1_Min-Max_Search.csv"

ExportRespSurfaceRefinePoints

Exports refinement points table to a file

UI Access	From the Response Surface tab of the Design of Experiments Post Analysis Display dialog box, select Refinement Points option under View , Click on Export...		
Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.
Return Value	None.		

Python Syntax	ExportRespSurfaceRefinePoints(<DOEName>, <FileName>)
Python Example	oModule.ExportRespSurfaceRefinePoints("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Refine_Points.csv")

VB Syntax	ExportRespSurfaceRefinePoints <DOEName>, <FileName>
VB Example	oModule.ExportRespSurfaceRefinePoints_ "DesignOfExperimentsSetup1", _ "C:/temp/DesignOfExperimentsSetup1_Refine_Points.csv"

ExportRespSurfaceResponsePoints

Exports response points table to a file

UI Access	From the Response Surface tab of the Design of Experiments Post Analysis Display dialog box, select Response Points option under View , Click on Export....		
Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.
Return Value	None.		

Python Syntax	ExportRespSurfaceResponsePoints (<DOEName>, <FileName>)		
Python Example	<pre>oModule.ExportRespSurfaceResponsePoints("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Response_Points.csv")</pre>		

VB Syntax	ExportRespSurfaceResponsePoints <DOEName>, <FileName>		
VB Example	<pre>oModule.ExportRespSurfaceResponsePoints_ "DesignOfExperimentsSetup1", _ "C:/temp/DesignOfExperimentsSetup1_Response_Points.csv"</pre>		

ExportRespSurfaceVerificationPoints

Exports verification points table to a file

UI Access	From the Response Surface tab of the Design of Experiments Post Analysis Display dialog box, select Verification Points option under View , Click on Export...		
Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.
Return Value	None.		

Python Syntax	ExportRespSurfaceVerificationPoints (<DOEName>, <FileName>)
Python Example	<pre>oModule.ExportRespSurfaceVerificationPoints ("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Veri_Points.csv")</pre>

VB Syntax	ExportRespSurfaceVerificationPoints <DOEName>, <FileName>
VB Example	<pre>oModule.ExportRespSurfaceVerificationPoints_ "DesignOfExperimentsSetup1", _ "C:/temp/DesignOfExperimentsSetup1_Veri_Points.csv"</pre>

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"
```

GetChildNames [Optimetrics]

If used without a specific optimization setup name, gets a list of all setups for all types. If a with a specific setup name, returns names for that optimization setup.

UI Access	NA		
Parameters	Name	Type	Description
	typeName	text string	Optional, default to get all types of setup names. Or one of type name return in GetChildTypes(). Also, the type name can be used without the prefix "Opti".
Return Value	Array of setup names.		

Python Syntax	GetChildNames()
Python Example	<pre>oOptimModule = oDesign.GetChildObject("Optimetrics") arrAllSetup = oOptimModule.GetChildNames() arrParmSetup = oOptimModule.GetChildNames("'OptiParametric'") arrOptimizeSetup = oOptimModule.GetChildNames("'Optimization'")</pre>

GetChildObject [Optimetrics]

Gets a Setup Object of the Optimetrics module

UI Access	NA		
Parameters	Name	Type	Description
	Setup Name	text string	A optimetrics setup name, names returned by the GetChildNames().
Return Value	A script object for the setup See discussion of Optimetrics Setup Objects in Object Script Property Function Summary .		

Python Syntax	GetChildObject()
Python Example	<pre>oParamSetup = oOptModule.GetChildObject('ParametricSetup1') oOptSetup = oOptModule.GetChildObject('OptimizationSetup1')</pre>

GetChildTypes [Optimetrics]

Use: Gets child types of queried Optimetrics module.

Syntax: GetChildTypes()

Return Value: Array of text string, it can be an empty array if there is no setup is defined. There are six types of setup, they are ['OptiParametric', 'OptiOptimization', 'OptiSensitivity', 'OptiStatistical', 'OptiDesignExplorer', 'OptiDXDOE'].

Python Syntax	GetChildTypes ()
Python Example	<pre>oOptimModule = oDesign.GetChildObject("Optimetrics") arrSetupTypes = oOptimModule.GetChildTypes()</pre>

GetName

Returns the design ID and design name of the active design, in that order separated by a semicolon. If the active design is a sub-circuit, it also returns the parent ID and parent design name, listed before the active design information.

To get the *only* the design's name without the parent design name or design IDs, see oDesign.GetDesignName().

UI Access	N/A
Parameters	None.
Return Value	String indicating the name and ID of the active design and parent design (if the active design is a sub-circuit).

Python Syntax	GetName()
Python Example	<pre>oDesign.GetName() for a subcircuit <i>EMDesign2</i> in a parent design <i>EMDesign1</i> design_name = oDesign.GetName() 1;EMDesign1/2;EMDesign2</pre> In the returned string: 1 is the parent design ID; <i>EMDesign1</i> is the parent design name. 2 is the active design ID; <i>EMDesign2</i> is the active design name.

VB Syntax	GetName
VB Example	<code>design_name = oDesign.GetName</code>

GetObjPath [Design]

Obtains the path to the design.

UI Access	N/A
Parameters	None.
Return Value	String containing the path to the design.

Python Syntax	GetObjPath()
Python Example	<code>oDesign.GetObjPath()</code>

VB Syntax	GetObjPath
VB Example	<code>oDesign.GetObjPath</code>

GetOptimetricResult

Returns an Optimetric calculation. The specific calculation is determined by the setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<SetupName>	String	Optimetrics setup name.
	<vars>	Array	Array containing string variable names. Use the Sweep Definitions tab in the UI or the <SweepDefs> parameter in the InsertSetup script to determine appropriate inputs.
	<values>	Array	<i>Optional.</i> Array containing string values. When multiple variables and values are provided, the order must be the same in both the <vars> and <values> arrays. The first variable is paired with the first value, the second variable is paired with the second value, and so on.
Return Value	Calculation result. If the setup contains more than one calculation, the output will be an array of values.		

Python Syntax	GetOptimetricResult(<SetupName>, <vars>, <values>)
Python Example	oModule.GetOptimetricResult('ParametricSetup1', ['AR', 'Re'], ['4.64', '6e+04'])

VB Syntax	GetOptimetricResult <SetupName>, <vars>, <values>
VB Example	<pre> dim vars(1) vars(0) = "AR" vars(1) = "Re" dim values(1) values(0) = "4.64" values(1) = "6e+04" </pre>


```
oModule.GetOptimetricResult "ParametricSetup1", vars, values
```

GetOptimetricsResult

Get an existing Optimization, Sensitivity, Statistical, Parametric or DesignXplorer result.

UI Access	Right click on the desired Optimetrics setup in the project tree and choose View Analysis Result... On the Post Analysis Display dialog box, click the Result tab, then select Table view, and click on the Export button		
Parameters	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.
Return Value	None		

Python Syntax	GetOptimetricsResult (<SetupName>, <FileName>, [useFullOutputName])
Python Example	<pre>oModule.GetOptimetricsResult ("StatisticalSetup1", "c:/exportdir/test.csv", false)</pre>

VB Syntax	GetOptimetricsResult <SetupName>, <FileName>, [useFullOutputName]
VB Example	<pre>oModule.ExportOptimetricsResult "StatisticalSetup1", "c:/exportdir/test.csv", false</pre>

GetPropNames [Optimetrics]

Use: Always returns the empty set for Optimetrics objects since they do not have properties.

Syntax: `GetPropNames(bIncludeReadOnly)`

Return Value: Returns empty set.

Parameters: `bIncludeReadOnly`—optional, default to `True`.

Python Syntax	<code>GetPropNames ()</code>
Python Example	<pre>oOptModule.GetPropNames () oOptModule.GetPropNames (True) oOptModule.GetPropNames (False)</pre>

GetPropValue [Optimetrics]

Returns the property value for a setup property.

UI Access	NA		
Parameters	Name	Type	Description
	property-path		a child object's property path. See property path discussion here .
Return Value	Returns the value of an setup property.		

Python Syntax	<code>GetPropValue(propPath)</code>
Python Example	<pre>oOptModule.GetPropValue("OptimizationSetup1\Optimizer") //get the optimizer name for OptimizationSetup1</pre>

	<pre>oOptModule.GetPropValue("OptimizationSetup1\Optimizer\Choices") //Get the menu property's menu items. In this case all Optimizer names.</pre>

VB Syntax	<code>GetPropValue()</code>
VB Example	<pre>GetPropValue("Optimetrics/OptimizationSetup1/Enabled") Returns True if Enabled, or False if disabled.</pre>

GetSetupNames [Optimetrics]

Gets a list of Optimetrics setup names

UI Access	NA		
Parameters	Name	Type	Description
	None		
Return Value	IAnsoftCollectionObj – a collection of Optimetrics setup names		

Python Syntax	<code>GetSetupNames()</code>
Python Example	<pre>oModule = oDesign.GetModule("Optimetrics") setupNames = oModule.GetSetupNames()</pre>

VB Syntax	GetSetupNames()
VB Example	<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.

UI Access	NA		
Parameters	Name	Type	Description
	<Optimetrics type>	String	Examples: parametric, optimization, statistical, sensitivity
Return Value	Array of Optimetrics setup names of the given type.		

Python Syntax	GetSetupNamesByType (<Optimetrics type>)
Python Example	<pre>for name in oModule.GetSetupNamesByType("optimization") AddInfoMessage(str(name))</pre>

VB Syntax	GetSetupNamesByType <Optimetrics type>
VB Example	<pre>For each name in oModule.GetSetupNamesByType("optimization") MsgBox name</pre>

	Next

ImportSetup

Import an Optimetric setup from a file.

UI Access	NA		
Parameters	Name	Type	Description
	<SetupTypeName>	String	Must be one of "OptiParametric" , "OptiOptimization" , "OptiSensitivity" , "OptiStatistical" , or "OptiDesignExplorer".
	<SetupInfo>	Array	Array("NAME:<SetupName>", "FilePath") <SetupName> Type: <string> Name of the setup. <FilePath> Type : <string: file path> Must be a valid file path and name.
Return Value	None		

Python Syntax	ImportSetup (<SetupTypeName>, <SetupInfo>)
Python Example	<pre>oModule.ImportSetup ("OptiStatistical", ["NAME:StatisticalSetup1",</pre>

	<code>"c:/importdir/mySetupInfoFile"])</code>

VB Syntax	<code>ImportSetup <SetupTypeName>, <SetupInfo></code>
VB Example	<pre>oModule.ImportSetup "OptiStatistical", Array("NAME:StatisticalSetup1", "c:/importdir/mySetupInfoFile")</pre>

PasteSetup [Optimetrics]

Pastes the specified Optimetrics setup.

UI Access	NA		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the Setup
Return Value	None		

Python Syntax	<code>PasteSetup (<SetupName>)</code>
Python Example	<code>oModule.PasteSetup ("OptimizationSetup1")</code>

VB Syntax	<code>PasteSetup <SetupName></code>

VB Example	<code>oModule.PasteSetup "OptimizationSetup1"</code>

RenameSetup [Optimetrics]

Renames the specified Optimetrics setup.

UI Access	Right-click the setup in the project tree, and then click Rename on the shortcut menu.		
Parameters	Name	Type	Description
	<OldName>	String	The name that needs to be replaced
	<NewName>	String	Replacement name
Return Value	None		

Python Syntax	<code>RenameSetup (<OldName> <NewName>)</code>
Python Example	<code>oModule.RenameSetup ("OptimizationSetup1" "MyOptimization")</code>

VB Syntax	<code>RenameSetup <OldName> <NewName></code>
VB Example	<code>oModule.RenameSetup "OptimizationSetup1" "MyOptimization"</code>

SetPropValue [Optimetrics]

Sets the property value for the active Optimetrics setup.

UI Access	Set Property value on Optimetrics objects		
Parameters	Name	Type	Description
	Property path	text string	Setup property path. See discussion of Property Path
	new Value	Text String, Number, or Boolean	New value data type is depending on the property type,
Return Value	True if the property is found and the new value is valid. Otherwise return False.		

Python Syntax	SetPropValue(propPath, newValue)		
Python Example	<pre>oOptModule.SetPropValue("ParametricSetup1\Enabled", False) //disable ParametricSetup1</pre>		
	<pre>oOptModule.SetPropValue("OptimizationSetup1/Optimizer","Quasi Newton")</pre>		

VB Syntax	SetPropValue(propPath, newValue)		
VB Example	SetPropValue("ParametricSetup1\Enabled", False)		

SolveAllSetup

Solves all Optimetrics setups

UI Access	Right-click on Optimetrics in Project Manager and select Analyze>All from context menu		
Parameters	Name	Type	Description
	isBlocking	Boolean	An optional arg that defaults to <code>true</code> . If it's <code>false</code> , the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> com-

	<table><tr><td></td><td></td><td>mand.</td></tr></table>			mand.
		mand.		
Return Value	None			

Python Syntax	SolveAllSetup()
Python Example	<code>oModule.SolveAllSetup()</code>

VB Syntax	SolveAllSetup
VB Example	<code>oModule.SolveAllSetup</code>

SolveSetup [Optimetrics]

Solves the specified Optimetrics setup.

UI Access	Right-click the setup in the project tree, and then click Analyze on the shortcut menu.		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup to be solved
	isBlocking	Boolean	An optional arg that defaults to <code>true</code> . If it's <code>false</code> , the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.
Return Value	None		

Python Syntax	SolveSetup (<SetupName>)
Python Example	<code>oModule.SolveSetup ("OptimizationSetup1")</code>

VB Syntax	SolveSetup <SetupName>
VB Example	<code>oModule.SolveSetup "OptimizationSetup1"</code>

ValidateSetup [Parametric]

Performs a validation of all design variations in a parametric setup.

For information on reviewing the output produced by this script command, see: [Validating a Parametric Setup](#)

UI Access	Under <i>Optimetrics</i> in the Project Manager, right-click the <Parametric Setup Name> and then click Validate Parametric Setup on the shortcut menu.		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the parametric setup.
Return Value	None		

Python Syntax	ValidateSetup (<SetupName>)
Python Example	<code>oModule.ValidateSetup ('ParametricSetup1')</code>

VB Syntax	ValidateSetup <SetupName>

VB Example

```
oModule.ValidateSetup "ParametricSetup1"
```

General Commands Recognized by the Optimetrics Module

Following are general script commands recognized by the **Optimetrics** module:

[CopySetup](#)

[DeleteSetups \[Optimetrics\]](#)

[DistributedAnalyzeSetup](#)

[EditSetup](#)

[EnableSetup](#)

[ExportDXConfigFile](#)

[ExportOptimetricsProfile](#)

[ExportOptimetricsResult](#)

[ExportParametricResults](#)

[ExportRespSurfaceMinMaxTable](#)

[ExportRespSurfaceRefinePoints](#)

[ExportRespSurfaceResponsePoints](#)

[ExportRespSurfaceVerificationPoints](#)

[ExportDOEResponseCurve](#)

[ExportDOEResponseCurveSlices](#)

[ExportDOEResponseSurface](#)

[ExportDOELocalSensitivity](#)

[ExportDOELocalSensitivityCurve](#)

[GetChildNames \[Optimetrics\]](#)

[GetChildObject \[Optimetrics\]](#)

[GetChildTypes \[Optimetrics\]](#)

[GetName](#)

[GetObjPath \[Editor\]](#)

[GetOptimetricResult](#)

[GetPropNames \[Optimetrics\]](#)

[GetPropValue \[Optimetrics\]](#)

[GetSetupNames \[Optimetrics\]](#)

[GetSetupNamesByType \[Optimetrics\]](#)

[ImportSetup](#)

[InsertSetup](#)

[PasteSetup \[Optimetrics\]](#)

[RenameSetup \[Optimetrics\]](#)

[SetPropValue \[Optimetrics\]](#)

[SolveSetup \[Optimetrics\]](#)

[SolveAllSetup](#)

CopySetup

Copy the specified Optimetrics setup.

UI Access	NA		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup.
Return Value	None.		

Python Syntax	CopySetup (<SetupName>)		
Python Example	oModule.CopySetup ("OptimizationSetup1")		

VB Syntax	CopySetup <SetupName>		
VB Example	oModule.CopySetup "OptimizationSetup1"		

DeleteSetups [Optimetrics]

Deletes the specified Optimetrics setups.

UI Access	Right-click the setup in the project tree, and then click Delete on the shortcut menu		
Parameters	Name	Type	Description
	<NameArray>	Array of Strings	An Array of Setup Names
Return Value	None		

Python Syntax	DeleteSetups (<NameArray>)
Python Example	<code>oModule.DeleteSetups (["OptimizationSetup1"])</code>

VB Syntax	DeleteSetups <NameArray>
VB Example	<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

UI Access	Right-click the parametric setup name in the project tree and select Distribute Analysis.		
Parameters	Name	Type	Description
	<ParametricSetupName>	String	Name of the Setup
	isBlocking	Boolean	An optional arg that defaults to <code>true</code> . If it's <code>false</code> , the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.
Return Value	None		

Python Syntax	DistributedAnalyzeSetup (<ParametricSetupName>)
Python Example	<code>oModule.DistributedAnalyzeSetup ("ParametricSetup1")</code>

VB Syntax	DistributedAnalyzeSetup <ParametricSetupName>
VB Example	oModule.DistributedAnalyzeSetup "ParametricSetup1"

EnableSetup

Enables and disables a defined optimetrics analysis setup.

UI Access	Right-click on a setup in the project tree, select Enable Setup or Disable Setup		
Parameters	Name	Type	Description
	<SetupName>	String	Name of specified setup.
	<Enable>	Boolean	Determines whether enable or disable a setup. • True - enable setup. • False - disable setup.
Return Value	None.		

Python Syntax	EnableSetup(<SetupName>, <Enable>)
Python Example	oModule.EnableSetup("OptimizationSetup1", True)

VB Syntax	EnableSetup <SetupName>, <Enable>
VB Example	oModule.EnableSetup "OptimizationSetup1", true

ExportDXConfigFile

Create an xml file with the setup information for Design Explorer

UI Access	Right click on the Design Xplorer setup in the project tree and choose Export External Connector Addin Configuration...		
Parameters	Name	Type	Description
	<SetupName>	String	Must be one of existing DesignExplorer setup names
	<FileName>	String	Must be a valid file path and name
Return Value	None		

Python Syntax	ExportDXConfigFile (<SetupName>, <FileName>)		
Python Example	<pre>oModule.ExportDXConfigFile ("DesignXplorerSetup1", "c:/exportdir/DXSetup1.xml")</pre>		

VB Syntax	ExportDXConfigFile <SetupName>, <FileName>		
VB Example	<pre>oModule.ExportDXConfigFile ("DesignXplorerSetup1", "c:/exportdir/DXSetup1.xml")</pre>		

ExportOptimetricsProfile

Export Optimetrics profile data

UI Access	Right click on the Optimetrics setup in the project tree and choose View Analysis Result... On the Post Analysis Display dialog box, click the Profile tab and click on the Export button.		
Parameters	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.
Return Value	None		

Python Syntax	ExportOptimetricsProfile (<SetupName>, <FileName>, [profileNum])
Python Example	<pre>oModule.ExportOptimetricsProfile ("StatisticalSetup1", "c:/exportdir/test.csv")</pre>

VB Syntax	ExportOptimetricsProfile <SetupName>, <FileName>, [profileNum]
VB Example	<pre>oModule.ExportOptimetricsProfile "StatisticalSetup1", "c:/exportdir/test.csv"</pre>

ExportOptimetricsResult

Export an existing Optimization, Sensitivity, Statistical or DesignXplorer result. (Does not export Parametric results.)

UI Access	Right click on the desired Optimetrics setup in the project tree and choose View Analysis Result... On the Post Analysis Display dialog box, click the Result tab, then select Table view, and click on the Export button		
Parameters	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.
Return Value	None		

Python Syntax	ExportOptimetricsResult (<SetupName>, <FileName>, [useFullOutputName])		
Python Example	<pre>oModule.ExportOptimetricsResult ("StatisticalSetup1", "c:/exportdir/test.csv", false)</pre>		

VB Syntax	ExportOptimetricsResult <SetupName>, <FileName>, [useFullOutputName]		
VB Example	<pre>oModule.ExportOptimetricsResult "StatisticalSetup1", "c:/exportdir/test.csv", false</pre>		

ExportParametricResults

Export existing Parametric results.

UI Access	Right click on the desired Parametric setup in the project tree and choose View Analysis Result... On the Post Analysis Display dialog box, click the Result tab, then select Table view, and click on the Export button.		
Parameters	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
Return Value	None		

Python Syntax	ExportParametricResults (<SetupName>, <FileName>, <bOutputUnits>)
Python Example	<pre>oModule.ExportParametricResults ("ParametricSetup1", "c:/exportdir/test.csv", False)</pre>

VB Syntax	ExportParametricResults <SetupName>, <FileName>, <bOutputUnits>
VB Example	<pre>oModule.ExportParametricResults "ParametricSetup1", "c:/exportdir/test.csv", false</pre>

ExportParametricSetupTable

Exports the parametric setup table as a CSV file.

UI Access	Double-click parametric setup. Select Table tab. Click Export .		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup.
	<filePath>	String	Full path for file export.
Return Value	None		

Python Syntax	ExportParametricSetupTable (<SetupName>, <filePath>)

Python Example	<code>oModule.ExportParametricSetupTable('ParametricSetup1', 'E:/Files/ParametricSetup1_Table.csv')</code>

VB Syntax	<code>ExportParametricSetupTable <SetupName>, <filePath></code>
VB Example	<code>obj.ExportParametricSetupTable "ParametricSetup1", "E:/Files/ParametricSetup1_Table.csv"</code>

ExportRespSurfaceMinMaxTable

Exports min-max table from a response surface to a file

UI Access	Click on Export... From the Response Surface tab of the Design of Experiments Post Analysis Display dialog.		
Parameters	Name	Type	Description
	<code><DOEName></code>	String	Name of the Design of Experiments (DOE) setup.
	<code><FileName></code>	String	Output file name with path.
Return Value	None.		

Python Syntax	<code>ExportRespSurfaceMinMaxTable(<DOEName>, <FileName>)</code>
Python Example	<code>oModule.ExportRespSurfaceMinMaxTable("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Min-Max_Search.csv")</code>

VB Syntax	<code>ExportRespSurfaceMinMaxTable <DOEName>, <FileName></code>
VB Example	<pre>oModule.ExportRespSurfaceMinMaxTable _ "DesignOfExperimentsSetup1", _ "C:/temp/DesignOfExperimentsSetup1_Min-Max_Search.csv"</pre>

ExportRespSurfaceRefinePoints

Exports refinement points table to a file

UI Access	From the Response Surface tab of the Design of Experiments Post Analysis Display dialog box, select Refinement Points option under View , Click on Export....		
Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.
Return Value	None.		

Python Syntax	<code>ExportRespSurfaceRefinePoints(<DOEName>, <FileName>)</code>
Python Example	<pre>oModule.ExportRespSurfaceRefinePoints("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Refine_Points.csv")</pre>

VB Syntax	<code>ExportRespSurfaceRefinePoints <DOEName>, <FileName></code>
VB Example	<code>oModule.ExportRespSurfaceRefinePoints_</code>

	"DesignOfExperimentsSetup1", _ "C:/temp/DesignOfExperimentsSetup1_Refine_Points.csv"

ExportRespSurfaceResponsePoints

Exports response points table to a file

UI Access	From the Response Surface tab of the Design of Experiments Post Analysis Display dialog box, select Response Points option under View , Click on Export...		
Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.
Return Value	None.		

Python Syntax	ExportRespSurfaceResponsePoints (<DOEName>, <FileName>)
Python Example	oModule.ExportRespSurfaceResponsePoints ("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Response_Points.csv")

VB Syntax	ExportRespSurfaceResponsePoints <DOEName>, <FileName>
VB Example	oModule.ExportRespSurfaceResponsePoints_ "DesignOfExperimentsSetup1", _ "C:/temp/DesignOfExperimentsSetup1_Response_Points.csv"

ExportRespSurfaceVerificationPoints

Exports verification points table to a file

UI Access	From the Response Surface tab of the Design of Experiments Post Analysis Display dialog box, select Verification Points option under View , Click on Export...		
Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.
Return Value	None.		

Python Syntax	ExportRespSurfaceVerificationPoints (<DOEName>, <FileName>)
Python Example	<pre>oModule.ExportRespSurfaceVerificationPoints("DesignOfExperimentsSetup1", "C:/temp/DesignOfExperimentsSetup1_Veri_Points.csv")</pre>

VB Syntax	ExportRespSurfaceVerificationPoints <DOEName>, <FileName>
VB Example	<pre>oModule.ExportRespSurfaceVerificationPoints_ "DesignOfExperimentsSetup1", _ "C:/temp/DesignOfExperimentsSetup1_Veri_Points.csv"</pre>

ExportDOEResponseCurve

Exports response curve from a response surface to a file.

UI Access	Click on Export... From the Response Surface tab of the Design of Experiments Post Analysis Display dialog.		
Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.
Return Value	None		

Python Syntax	<i>DOEName (<FileName>)</i>		
Python Example	<pre>oModule.ExportDOEResponseCurve("DesignOfExperimentsSetup1", "C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_Response_Points.csv")</pre>		

VB Syntax	<i>DOEName (<FileName>)</i>		
VB Example	<pre>oModule.ExportDOEResponseCurve "DesignOfExperimentsSetup1", Array(_ "\$CPU_Power=25W", "\$Fan_X=2m_per_sec", "\$Memory_Power=5W"), "\$CPU_Power", 50_ "CPU_Monitor1.Temperature", _C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_responsecurve.csv"</pre>		

ExportDOEResponseCurveSlices

Exports response curve slices from a response surface to a file.

UI Access	Click on Export... From the Response Surface tab of the Design of Experiments Post Analysis Display dialog.

Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.
Return Value	None		

Python Syntax	<i>DOEName (<FileName>)</i>
Python Example	<pre>oModule.ExportDOEResponseCurveSlices("DesignOfExperimentsSetup1", "C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_Response_Points.csv")</pre>

VB Syntax	<i>DOEName (<FileName>)</i>
VB Example	<pre>oModule.ExportDOEResponseCurveSlices "DesignOfExperimentsSetup1", Array("\$CPU_Power=25W", "\$Fan_X=2m_per_sec", "\$Memory_Power=5W"), "\$CPU_Power", 50, _"\$Fan_X", 50, "CPU_Monitor1.Temperature", _C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1.csv"</pre>

ExportDOEResponseSurface

Exports response surface to a file.

UI Access	Click on Export... From the Response Surface tab of the Design of Experiments Post Analysis Display dialog.

Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.
Return Value	None		

Python Syntax	<i>DOEName (<FileName>)</i>
Python Example	<pre>oModule.ExportDOEResponseSurface("DesignOfExperimentsSetup1", "C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_Response_Points.csv")</pre>

VB Syntax	<i>DOEName (<FileName>)</i>
VB Example	<pre>oModule.ExportDOEResponseCurveSurface "DesignOfExperimentsSetup1", Array("\$CPU_Power=25W", "\$Fan_X=2m_per_sec", "\$Memory_Power=5W"), "\$CPU_Power", 50, _"\$Fan_X", 50, "CPU_Monitor1.Temperature", _C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1.csv"</pre>

ExportDOELocalSensitivity

Exports local sensitivity to a file.

UI Access	Click on Export... From the Response Surface tab of the Design of Experiments Post Analysis Display dialog.		
Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.

Return Value	None

Python Syntax	<i>DOEName</i> (<FileName>)
Python Example	<pre>oModule.ExportDOELocalSensitivity("DesignOfExperimentsSetup1", "C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_Response_Points.csv")</pre>

VB Syntax	<i>DOEName</i> (<FileName>)
VB Example	<pre>oModule.ExportDOELocalSensitivity "DesignOfExperimentsSetup1", Array(_ "\$CPU_Power=25W", "\$Fan_X=2m_per_sec", "\$Memory_Power=5W"), Array("\$CPU_Power", _"\$Fan_X", "\$Memory_Power"), "CPU_Monitor1.Temperature", true, _"C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_localsensitivity.csv"</pre>

ExportDOELocalSensitivityCurve

Exports local sensitivity curves to a file.

UI Access	Click on Export... From the Response Surface tab of the Design of Experiments Post Analysis Display dialog.		
Parameters	Name	Type	Description
	<DOEName>	String	Name of the Design of Experiments (DOE) setup.
	<FileName>	String	Output file name with path.
Return Value	None		

Python Syntax	<i>DOEName</i> (<FileName>)
Python Example	<pre>oModule.ExportDOELocalSensitivityCurve("DesignOfExperimentsSetup1", "C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_Response_Points.csv")</pre>

VB Syntax	<i>DOEName</i> (<FileName>)
VB Example	<pre>oModule.ExportDOELocalSensitivityCurve "DesignOfExperimentsSetup1", Array(_ "\$CPU_Power=25W", "\$Fan_X=2m_per_sec", "\$Memory_Power=5W"), Array("\$CPU_Power", _ "\$Fan_X", "\$Memory_Power"), "Input Parameters", 50, "CPU_Monitor1.Temperature", _ "C:/Users/AEDT/R24.1/D/DesignOfExperimentsSetup1_localsensitivitycurves.csv"</pre>

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"
```

GetChildNames [Optimetrics]

If used without a specific optimization setup name, gets a list of all setups for all types. If a with a specific setup name, returns names for that optimization setup.

UI Access	NA		
Parameters	Name	Type	Description
	typeName	text string	Optional, default to get all types of setup names. Or one of type name return in GetChildTypes(). Also, the type name can be used without the prefix "Opti".
Return Value	Array of setup names.		

Python Syntax	GetChildNames()		
Python Example	<pre>oOptimModule = oDesign.GetChildObject("Optimetrics") arrAllSetup = oOptimModule.GetChildNames() arrParmSetup = oOptimModule.GetChildNames("'OptiParametric'") arrOptimizeSetup = oOptimModule.GetChildNames("'Optimization'")</pre>		

GetChildObject [Optimetrics]

Gets a Setup Object of the Optimetrics module

UI Access	NA		
Parameters	Name	Type	Description

	Setup Name	text string	A optimetrics setup name, names returned by the GetChildNames().
Return Value	A script object for the setup See discussion of Optimetrics Setup Objects in Object Script Property Function Summary .		

Python Syntax	GetChildObject()
Python Example	<pre>oParamSetup = oOptModule.GetChildObject('ParametricSetup1') oOptSetup = oOptModule.GetChildObject('OptimizationSetup1')</pre>

GetChildTypes [Optimetrics]

Use: Gets child types of queried Optimetrics module.

Syntax: GetChildTypes()

Return Value: Array of text string, it can be an empty array if there is no setup is defined. There are six types of setup, they are ['OptiParametric', 'OptiOptimization', 'OptiSensitivity', 'OptiStatistical', 'OptiDesignExplorer', 'OptiDXDOE'].

Python Syntax	GetChildTypes ()
Python Example	<pre>oOptimModule = oDesign.GetChildObject("Optimetrics") arrSetupTypes = oOptimModule.GetChildTypes()</pre>

GetName

Returns the design ID and design name of the active design, in that order separated by a semicolon. If the active design is a sub-circuit, it also returns the parent ID and parent design name, listed before the active design information.

To get the *only* the design's name without the parent design name or design IDs, see oDesign.GetDesignName().

UI Access	N/A
Parameters	None.
Return Value	String indicating the name and ID of the active design and parent design (if the active design is a sub-circuit).

Python Syntax	GetName()
Python Example	<pre>oDesign.GetName() for a subcircuit <i>EMDesign2</i> in a parent design <i>EMDesign1</i> design_name = oDesign.GetName() 1;EMDesign1/2;EMDesign2</pre> In the returned string: 1 is the parent design ID; <i>EMDesign1</i> is the parent design name. 2 is the active design ID; <i>EMDesign2</i> is the active design name.

VB Syntax	GetName
VB Example	<pre>design_name = oDesign.GetName</pre>

GetObjPath [Design]

Obtains the path to the design.

UI Access	N/A

Parameters	None.
Return Value	String containing the path to the design.

Python Syntax	GetObjPath()
Python Example	<code>oDesign.GetObjPath()</code>

VB Syntax	GetObjPath
VB Example	<code>oDesign.GetObjPath</code>

GetOptimetricResult

Returns an Optimetric calculation. The specific calculation is determined by the setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<SetupName>	String	Optimetrics setup name.
	<vars>	Array	Array containing string variable names. Use the Sweep Definitions tab in the UI or the <SweepDefs> parameter in the InsertSetup script to determine appropriate inputs.
	<values>	Array	<i>Optional.</i> Array containing string values. When multiple variables and values are provided, the order must be the same in both the <vars> and <values> arrays. The first variable is paired with the first value, the second variable is paired with the second value, and so on.
Return Value	Calculation result. If the setup contains more than one calculation, the output will be an array of values.		

Python Syntax	GetOptimetricResult(<SetupName>, <vars>, <values>)
Python Example	<code>oModule.GetOptimetricResult('ParametricSetup1', ['AR', 'Re'], ['4.64', '6e+04'])</code>

VB Syntax	GetOptimetricResult <SetupName>, <vars>, <values>
VB Example	<pre>dim vars(1) vars(0) = "AR" vars(1) = "Re" dim values(1) values(0) = "4.64" values(1) = "6e+04" oModule.GetOptimetricResult "ParametricSetup1", vars, values</pre>

GetPropNames [Optimetrics]

Use: Always returns the empty set for Optimetrics objects since they do not have properties.

Syntax: GetPropNames(bIncludeReadOnly)

Return Value: Returns empty set.

Parameters: bIncludeReadOnly—optional, default to True.

Python Syntax	GetPropNames ()

Python Example	<pre>oOptModule.GetPropNames() oOptModule.GetPropNames(True) oOptModule.GetPropNames(False)</pre>

GetPropValue [Optimetrics]

Returns the property value for a setup property.

UI Access	NA		
Parameters	Name	Type	Description
	property-path		a child object's property path. See property path discussion here .
Return Value	Returns the value of an setup property.		

Python Syntax	GetPropValue(propPath)
Python Example	<pre>oOptModule.GetPropValue("OptimizationSetup1\Optimizer") //get the optimizer name for OptimizationSetup1 oOptModule.GetPropValue("OptimizationSetup1\Optimizer\Choices") //Get the menu property's menu items. In this case all Optimizer names.</pre>

VB Syntax	GetPropValue()
VB Example	<pre>GetPropValue("Optimetrics/OptimizationSetup1/Enabled") Returns True if Enabled, or False if disabled.</pre>

GetSetupNames [Optimetrics]

Gets a list of Optimetrics setup names

UI Access	NA		
Parameters	Name	Type	Description
	None		
Return Value	IAnsoftCollectionObj – a collection of Optimetrics setup names		

Python Syntax	GetSetupNames()
Python Example	<pre>oModule = oDesign.GetModule("Optimetrics") setupNames = oModule.GetSetupNames()</pre>

VB Syntax	GetSetupNames()
VB Example	<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.

UI Access	NA		
Parameters	Name	Type	Description
	<Optimetrics type>	String	Examples: parametric, optimization, statistical, sensitivity
Return Value	Array of Optimetrics setup names of the given type.		

Python Syntax	GetSetupNamesByType (<Optimetrics type>)		
Python Example	<pre>for name in oModule.GetSetupNamesByType("optimization") AddInfoMessage(str(name))</pre>		

VB Syntax	GetSetupNamesByType <Optimetrics type>		
VB Example	<pre>For each name in oModule.GetSetupNamesByType("optimization") MsgBox name Next</pre>		

ImportSetup

Import an Optimetric setup from a file.

UI Access	NA		
Parameters	Name	Type	Description
	<SetupTypeName>	String	Must be one of "OptiParametric" , "OptiOptimization" , "OptiSensitivity" , "OptiStatistical" , or "OptiDesignExplorer".
	<SetupInfo>	Array	Array("NAME:<SetupName>" , "FilePath")

	<table><tr><td></td><td><SetupName> Type: <string> Name of the setup. <FilePath> Type : <string: file path> Must be a valid file path and name.</td></tr></table>		<SetupName> Type: <string> Name of the setup. <FilePath> Type : <string: file path> Must be a valid file path and name.
	<SetupName> Type: <string> Name of the setup. <FilePath> Type : <string: file path> Must be a valid file path and name.		
Return Value	None		

Python Syntax	ImportSetup (<SetupTypeName>, <SetupInfo>)
Python Example	<pre>oModule.ImportSetup ("OptiStatistical", ["NAME:StatisticalSetup1", "c:/importdir/mySetupInfoFile"])</pre>

VB Syntax	ImportSetup <SetupTypeName>, <SetupInfo>
VB Example	<pre>oModule.ImportSetup "OptiStatistical", Array("NAME:StatisticalSetup1", "c:/importdir/mySetupInfoFile")</pre>

PasteSetup [Optimetrics]

Pastes the specified Optimetrics setup.

UI Access	NA		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the Setup
Return Value	None		

Python Syntax	PasteSetup (<SetupName>)
Python Example	<code>oModule.PasteSetup ("OptimizationSetup1")</code>

VB Syntax	PasteSetup <SetupName>
VB Example	<code>oModule.PasteSetup "OptimizationSetup1"</code>

RenameSetup [Optimetrics]

Renames the specified Optimetrics setup.

UI Access	Right-click the setup in the project tree, and then click Rename on the shortcut menu.		
Parameters	Name	Type	Description
	<OldName>	String	The name that needs to be replaced
	<NewName>	String	Replacement name
Return Value	None		

Python Syntax	RenameSetup (<OldName> <NewName>)
Python Example	<code>oModule.RenameSetup ("OptimizationSetup1" "MyOptimization")</code>

VB Syntax	RenameSetup <OldName> <NewName>
VB Example	<code>oModule.RenameSetup "OptimizationSetup1" "MyOptimization"</code>

SetPropValue [Optimetrics]

Sets the property value for the active Optimetrics setup.

UI Access	Set Property value on Optimetrics objects		
Parameters	Name	Type	Description
	Property path	text string	Setup property path. See discussion of Property Path
	new Value	Text String, Number, or Boolean	New value data type is depending on the property type,
Return Value	True if the property is found and the new value is valid. Otherwise return False.		

Python Syntax	SetPropValue(propPath, newValue)
Python Example	<code>oOptModule.SetPropValue("ParametricSetup1\Enabled", False) //disable ParametricSetup1</code>

	<code>oOptModule.SetPropValue("OptimizationSetup1/Optimizer", "Quasi Newton")</code>

VB Syntax	<code>SetPropValue(propPath, newValue)</code>
VB Example	<code>SetPropValue("ParametricSetup1\Enabled", False)</code>

SolveAllSetup

Solves all Optimetrics setups

UI Access	Right-click on Optimetrics in Project Manager and select Analyze>All from context menu		
Parameters	Name	Type	Description
	isBlocking	Boolean	An optional arg that defaults to <code>true</code> . If it's <code>false</code> , the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.
Return Value	None		

Python Syntax	<code>SolveAllSetup()</code>
Python Example	<code>oModule.SolveAllSetup()</code>

VB Syntax	<code>SolveAllSetup</code>
VB Example	<code>oModule.SolveAllSetup</code>

SolveSetup [Optimetrics]

Solves the specified Optimetrics setup.

UI Access	Right-click the setup in the project tree, and then click Analyze on the shortcut menu.		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup to be solved
	isBlocking	Boolean	An optional arg that defaults to <code>true</code> . If it's <code>false</code> , the command immediately returns while the analysis or analyses run in the background. The status of the analyses can be checked with the <code>AreThereSimulationsRunning</code> command.
Return Value	None		

Python Syntax	<code>SolveSetup (<SetupName>)</code>
Python Example	<code>oModule.SolveSetup ("OptimizationSetup1")</code>

VB Syntax	<code>SolveSetup <SetupName></code>
VB Example	<code>oModule.SolveSetup "OptimizationSetup1"</code>

ValidateSetup [Parametric]

Performs a validation of all design variations in a parametric setup.

For information on reviewing the output produced by this script command, see: [Validating a Parametric Setup](#)

UI Access	Under <i>Optimetrics</i> in the Project Manager, right-click the <Parametric Setup Name> and then click Validate Parametric Setup on the shortcut menu.		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the parametric setup.
Return Value	None		

Python Syntax	ValidateSetup (<SetupName>)
Python Example	<code>oModule.ValidateSetup('ParametricSetup1')</code>

VB Syntax	ValidateSetup <SetupName>
VB Example	<code>oModule.ValidateSetup "ParametricSetup1"</code>

Parametric Script Commands

[EditSetup \[Parametric\]](#)

[ExportParametricSetupTable](#)

[GenerateVariationData \[Parametric\]](#)

[InsertSetup \[Parametric\]](#)

[ValidateSetup \[Parametric\]](#)

EditSetup [Parametric]

Modifies an existing parametric setup

UI Access	Right-click the setup in the project tree, and then click Properties on the shortcut menu.		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the Setup
	<ParametricParams>	List	List that defines the parameters of the parametric setup; examples are listed below.
Return Value	None		

Python Syntax	EditSetup (<SetupName>, <ParametricParams>)
Python Example	See EditSetup [Optimization]

VB Syntax	EditSetup <SetupName>, <ParametricParams>
VB Example	See EditSetup [Optimization]

ExportParametricSetupTable

Exports the parametric setup table as a CSV file.

UI Access	Double-click parametric setup. Select Table tab. Click Export .		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup.
	<filePath>	String	Full path for file export.
Return Value	None		

Python Syntax	ExportParametricSetupTable (<SetupName>, <filePath>)
Python Example	oModule.ExportParametricSetupTable('ParametricSetup1', 'E:/Files/ParametricSetup1_Table.csv')

VB Syntax	ExportParametricSetupTable <SetupName>, <filePath>
VB Example	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 Optimetrics folder in the project tree, and then click Add> Parametric 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	<i>SweepData</i> s 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.
Return Value	None		

Python Syntax	InsertSetup ("OptiParametric", <ParametricParams>)
Python Example	<pre> oModule.InsertSetup("OptiParametric", ["NAME:ParametricSetup1", "SaveFields:=", true, ["NAME:StartingPoint"], "Sim. Setups:=", ["Setup1"], ["NAME:Sweeps", ["NAME:SweepDefinition", "Variable:=", "\$width", "Data:=", "LIN 12mm 17mm 2.5mm", "OffsetFl:=", false, </pre>

```
        "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",
```


	<pre> "Stop:=", "8GHz"]]]]]]]]</pre>

VB Syntax	InsertSetup "OptiParametric", <ParametricParams>
VB Example	<pre>oModule.InsertSetup "OptiParametric", Array("NAME:ParametricSetup1", _ "SaveFields:=", true, _ Array("NAME:StartingPoint"), _ "Sim. Setups:=", Array("Setup1"), _ Array("NAME:Sweeps", _ Array("NAME:SweepDefinition", _ "Variable:=", "\$width", _ "Data:=", "LIN 12mm 17mm 2.5mm", _</pre>

```

"OffsetFl:=", false, _
"Synchronize:=", 0),
Array("NAME:SweepDefinition", _
"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"))))
```

ValidateSetup [Parametric]

Performs a validation of all design variations in a parametric setup.

For information on reviewing the output produced by this script command, see: [Validating a Parametric Setup](#)

UI Access	Under <i>Optimetrics</i> in the Project Manager, right-click the <Parametric Setup Name> and then click Validate Parametric Setup on the shortcut menu.		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the parametric setup.
Return Value	None		

Python Syntax	ValidateSetup (<SetupName>)
Python Example	oModule.ValidateSetup('ParametricSetup1')

VB Syntax	ValidateSetup <SetupName>
VB Example	oModule.ValidateSetup "ParametricSetup1"

Optimization Script Commands

[EditSetup \[Optimization\]](#)

[InsertSetup \[Optimization\]](#)

EditSetup [Optimization]

Modifies an existing optimization setup.

UI Access	Right-click the setup in the Project Manager tree, and then click Properties from the shortcut menu		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the Setup
	<OptimizationParams>	List	Parameters that define the setup; examples are listed below.
Return Value	None		

Python Syntax	EditSetup (<SetupName>, <OptimizationParams>)
Python Example	<pre>oModule.EditSetup("OptimizationSetup1", ["NAME:OptimizationSetup1", "UseFastCalculationUpdateAlgo:=", False, "FastCalcOptCtrledByUser:=", 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:Variables"
```

```
],  
[  
    "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 <SetupName>, <OptimizationParams>
VB Example	oModule.EditSetup "OptimizationSetup1",

```

Array("NAME:OptimizationSetup1", "UseFastCalculationUpdateAlgo:=", _
    false, "FastCalcOptCtrledByUser:=", false,
    "IsEnabled:=", true, "SaveSolutions:=", _
    false, Array("NAME:StartingPoint"), "Optimizer:=",
    "Quasi Newton", Array("NAME:AnalysisStopOptions", "StopForNumIteration:=", _
    true, "StopForElapsTime:=", false,
    "StopForSlowImprovement:=", false, "StopForGrdTolerance:=", _
    false, "MaxNumIteration:=", 1000, "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:=", "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:=", _

```


	<pre> "Real/Imag", "bG:=", Array("v:=", "[1;]")), "Weight:=", "[1;]")), "Acceptable_Cost:=", _ 0, "Noise:=", 0.0001, "UpdateDesign:=", false, "UpdateIteration:=", 5, "KeepReportAxis:=", _ true, "UpdateDesignWhenDone:=", true) </pre>

InsertSetup [Optimization]

Use: Inserts a new optimization setup.

UI Access	Right-click the Optimetrics folder in the project tree, and then click Add > Optimization on the shortcut menu.		
Parameters	Name	Type	Description
	<OptimizationParams>	Array	<pre> 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>)), </pre>

			"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.
Return Value	None		

<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.

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,</pre>

```
"PriorPSetup:=", "ParametricSetup1",
"PreSolvePSetup:=", true,
["NAME:Variables",
"$length:=", ["i:=", true, "Min:=", "6mm",
"Max:=", "18mm",
"MinStep:=", "0.001mm", "MaxStep:=",
"1.2mm"],
"$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", _ </pre>

```
"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)

```


Sensitivity Script Commands

[EditSetup \[Sensitivity\]](#)

[InsertSetup \[Sensitivity\]](#)

EditSetup [Sensitivity]

Modifies an existing sensitivity setup.

UI Access	Right-click the setup in the project tree, and then click Properties on the shortcut menu		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the Setup
Return Value	None		

Python Syntax	EditSetup (<SetupName>, <SensitivityParams>)
Python Example	<pre>oModule.EditSetup("OptimizationSetup1", ["NAME:OptimizationSetup1", "UseFastCalculationUpdateAlgo:=", False, "FastCalcOptCtrledByUser:=", False, "IsEnabled:=", True, "SaveSolutions:=", False,</pre>

```
[
"NAME:StartingPoint"
],
"Optimizer:=", "Quasi Newton",
[
"NAME:AnalysisStopOptions",
"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  
])
```

VB Syntax**EditSetup <SetupName>, <SensitivityParams>**

VB Example

```
oModule.EditSetup("OptimizationSetup1",  
Array(  
"NAME:OptimizationSetup1",  
"UseFastCalculationUpdateAlgo:=", False,  
"FastCalcOptCtrledByUser:=", False,  
"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
))

```

Statistical Script Commands

[EditSetup \[Statistical\]](#)

[InsertSetup Statistical](#)

EditSetup [Statistical]

Modifies an existing statistical setup.

UI Access	Right-click the setup in the project tree, and click Properties on the shortcut menu.		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the Setup
Return Value	None		

Python Syntax	EditSetup (<SetupName>, <StatisticalParams>)
Python Example	See EditSetup [Optimization]

VB Syntax	EditSetup <SetupName>, <StatisticalParams>
VB Example	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("optiguide")
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.

UI Access	Right-click Optimetrics in the project tree, and then click Add>Statistical on the shortcut menu.		
Parameters	Name	Type	Description
	<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	Distribution 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.
Return Value	None		

Python Syntax	InsertSetup ("OptiStatistical", <StatisticalParams>)
Python Example	<pre>oModule.InsertSetup("OptiStatistical", _ ["NAME:StatisticalSetup1", _ "SaveFields:=", true, _</pre>

```
["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>
VB Example	<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", _ "Type:=", "s", _ "Start:=", "8GHz", "Stop:=", "8GHz"))), _ </pre>

	<pre> Array("NAME:Goal",_ "Solution:=", "Setup1 : LastAdaptive",_ "Calculation:=", "reflect",_ "Context:=", "", _ Array("NAME:Ranges",_ "Range:=", Array("Var:=", "Freq", "Type:=",_ "s", "Start:=", "8GHz", "Stop:=", "8GHz")))) </pre>

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:=", "-1in",
"VarMax:=", "1in",
"Prob:=", "0.01",
"Mean:=", "0in")),
Array("NAME:Goals", Array("NAME:Goal",
```



```
"ReportType:=", "Modal Solution Data",  
"Solution:=", "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>

Distribution 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")))))
```

This page intentionally
left blank.

12 - Solutions Module Script Commands

Solutions commands should be executed by the "Solutions" module.

Set oModule = oDesign.GetModule("Solutions")

oModule.CommandName <args>

[DeleteSolutionVariation](#)

[EditSources \(Layout Editor\)](#)

[ExportEigenmodes](#)

[ExportNetworkData](#)

"FFTONReport " on page 12-18

[GetAvailableVariations](#)

[SetQ3DSATNetChanges \(Layout Editor\)](#)

DeleteImportData

Deletes imported solution or table data in HFSS. Not in HFSS-IE

UI Access	Right-click on Results > Import Solutions , click Delete Selections button.		
Parameters	Name	Type	Description
	<ImportSpecArray>	Array	Array of import solutions. Array(<ImportName:SolnName OR TableName>, ...)
Return Value	None.		

Python Syntax	<code>DeleteImportData(<ImportSpecArray>)</code>
Python Example	<pre>oModule.DeleteImportData(["Import1:Adaptive_1", "Import2:DataTable"])</pre>

VB Syntax	<code>DeleteImportData <ImportSpecArray></code>
VB Example	<pre>oModule.DeleteImportData _ Array("Import1:Adaptive_1", "Import2:DataTable")</pre>

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.

UI Access	Right-click on Results , select Browse Solutions... , click Delete button in the dialog.		
Parameters	Name	Type	Description
	<code><SoluParams></code>	Array	Structured array. <code>Array(<DataSpecifierArray>, ...)</code>
	<code><DataSpecifierArray></code>	Array	Structured array. <code>Array(<DesignVariationKey>, <SetupName>, <SolutionName>)</code>
Return Value	None.		

Python Syntax	DeleteSolutionVariation(< <i>SoluParams</i> >)
Python Example	<pre>oModule.DeleteSolutionVariation([["width='2in'", "Setup1", "Adaptive_1"], ["width='2in'", "Setup1", "Sweep1"]])</pre>

VB Syntax	DeleteSolutionVariation < <i>SoluParams</i> >
VB Example	<pre>oModule.DeleteSolutionVariation Array(_ Array("width='2in'", "Setup1", "Adaptive_1"), _ Array("width='2in'", "Setup1", "Sweep1"))</pre>

EditQ3DSources (Layout Editor)

Edits the parameters of the Q3D simulation's source excitations.

UI Access	From the Project Manager window, right-click a Q3D setup and select Q3D Edit Sources .		
Parameters	Name	Type	Description
	<Contents>	Array	Structured array containing these parameters in order: • <NAME:EditSourcesCmd> • <Q3D Setup Name> • <N Sources Value Type> • <Sources>
	< NAME:EditSourcesCmd>	String	Identifies the contents of the array (i.e., "EditSourcesCmd").

<Q3D Setup Name>	String	Name of the new solver setup (e.g., Q3D1).
<N Sources Value Type>	String	Type of the source value, where <i>N</i> is any of the following: A — current. V — voltage. N — not defined.
<DC Loss Type>	String	Identifies the DC volume loss calculation assumption. String "True DC Loss" or "DC Loss Low Frequency Sinusoidals"
<Sources>	Array	Structured array containing these parameters in order: • <NAME:Sources> • <IndividualSource>
<NAME:Sources>	String	Identifies the contents of the following arrays (i.e., "Sources").
<IndividualSource>	Array	Structured array containing these parameters in order: • <NAME:N Sources> • <Entry>
<NAME:N Sources>	String	Identifies the contents of the array (e.g., "AC Sources"), where <i>N</i> is AC , DC , or CG .
<Entry>	Array	Structured array containing some of these parameters in order: • <ID> • <Name> • <Value> • <Magnitude> • <Phase>
<ID>	Integer	Assigned ID number of the excitation (e.g., 372).
<Name>	String	Name of the excitation (e.g., Port1).
<Value>	String	Value of the excitation (e.g., 0A).
<Magnitude>	String	Magnitude value of the excitation (e.g., 0V).

	<table><tr><td><Phase></td><td>String</td><td>Phase value of the excitation (e.g., 0deg).</td></tr></table>			<Phase>	String	Phase value of the excitation (e.g., 0deg).
<Phase>	String	Phase value of the excitation (e.g., 0deg).				
Return Value	None.					

Python Syntax	EditQ3DSources([<Contents>])
Python Example	<pre>oModule.EditQ3DSources ([["NAME:EditSourcesCmd", "Q3D Setup Name:=", "Q3D1", "AC Sources Value Type:=", "A", "CG Sources Value Type:=", "N", "DC Sources Value Type:=", "A", "DC Loss Type:=", "DC Loss Low Frequency Sinusoidals", ["NAME:Sources", ["NAME:AC Sources", "Entry:=", ["ID:=", 372, "Name:=", "Port1", "Magnitude:=", "0A", "Phase:=", "0deg"], "Entry:=", ["ID:=", 366, "Name:=", "Port2", "Magnitude:=", "1A", "Phase:=", "30deg"],</pre>

```
"Entry:=", ["ID:=", 370, "Name:=", "Port2#ref", "Value:=",  
"0V", "Phase:=", "0deg"],  
  
"Entry:=", ["ID:=", 374, "Name:=", "Port3#ref", "Value:=",  
"0V", "Phase:=", "0deg"]  
  
],  
  
[  
  "NAME:CG Sources",  
  
  "Entry:=", ["ID:=", 0, "Name:=", "backplane", "Magnitude:=", "0V",  
  "Phase:=", "0deg"],  
  
  "Entry:=", ["ID:=", 2, "Name:=", "RXnet_pin1", "Magnitude:=", "1V",  
  "Phase:=", "90deg"],  
  
  "Entry:=", ["ID:=", 1, "Name:=", "RXnet_pin3", "Magnitude:=", "0V",  
  "Phase:=", "0deg"]  
  
],  
  
[  
  "NAME:DC Sources",  
  
  "Entry:=", ["ID:=", 372, "Name:=", "Port1", "Magnitude:=", "0A",  
  "Phase:=", "0deg"],  
  
  "Entry:=", ["ID:=", 366, "Name:=", "Port2", "Magnitude:=", "1A",  
  "Phase:=", "0deg"],  
  
  "Entry:=", ["ID:=", 370, "Name:=", "Port2#ref", "Magnitude:=", "0A",  
  "Phase:=", "0deg"],  
  
  "Entry:=", ["ID:=", 374, "Name:=", "Port3#ref", "Magnitude:=",  
  "0A", "Phase:=", "0deg"]
```

	<pre>]]]]) </pre>

VB Syntax	EditQ3DSources (<Contents>)
VB Example	<pre> oModule.EditQ3DSources Array(Array("NAME:EditSourcesCmd", "Q3D Setup Name:=" , "Q3D1", "AC Sources Value Type:=", "A", "CG Sources Value Type:=", "N", "DC Sources Value Type:=", "A", "DC Loss Type:=", "DC Loss Including Low Frequency Sinusoidals", Array("NAME:Sources", Array("NAME:AC Sources", "Entry:=", Array ("ID:=", 372, "Name:=", "Port1", "Magnitude:=", "0A", "Phase:=", "30deg"), "Entry:=", Array("ID:=", 366, "Name:=", "Port2", "Magnitude:=", "1A", "Phase:=", "0deg"], "Entry:=", Array("ID:=", 370, "Name:=", "Port2#ref", "Magnitude:=", "0A", "Phase:=", "0deg"), "Entry:=", Array("ID:=", 374, "Name:=", "Port3#ref", "Magnitude:=", "0A", "Phase:=", "0deg"))))) </pre>

EditSources (Layout Editor)

Indicates which source excitations should be used for fields post-processing.

UI Access	From the Project Manager window, right-click a Q3D setup and select Q3D Net Settings/Q3D Edit Sources .		
Parameters	Name	Type	Description
	<acData>	Array	Structured array that corresponds to the AC RL tab in the UI. Within this array, each sub-array corresponds to columns in the UI. The Name column is represented by <sourceNames>, and <sourceValues> assigns the Value and Unit to those sources, in the same order.

		<pre> Array("NAME:AC", "Value Type:=" , <string "V" for Voltage or "A" for Current>, <sourceNames array>, <magnitudeArray>) </pre>
<capData>	Array	Structured array that corresponds to the CG tab in the UI. Within this array, each sub-array corresponds to columns in the UI. The Name column is represented by <sourceNames>. The first <sourceValues> array assigns the Magnitude and its Unit, while the second <sourceValues> array assigns the Phase and its Unit. <pre> Array("NAME:Cap", "Value Type:=" , <string "N" for null>, <sourceNames array>, <magnitudeArray>, <phaseArray>) </pre>
<dcData>	Array	Structured array that corresponds to the DC RL tab in the UI. Within this array, each sub-array corresponds to columns in the UI. The Name column is represented by <sourceNames>, and <sourceValues> assigns the Value and Unit to those sources, in the same order. <pre> Array("NAME:DC", "Value Type:=" , <string "V" for Voltage or "A" for Current>, <sourceNames array>, <magnitudeArray>) </pre>
<sourceNames>	Array	Structured array.

			<pre>Array("NAME:Source Names", <string source name>, <string source name>, ...)</pre>
	<magnitudeArray>	Array	Structured array containing magnitude values in the same order as the sources in <sourceNames>. <pre>Array("NAME:Source Magnitudes", <string containing value and unit>, <string containing value and unit>, ...)</pre>
	<phaseArray>	Array	Structured array containing phase values in the same order as the sources in <sourceNames>. <pre>Array("NAME:Source Phases", <string containing value and unit>, <string containing value and unit>, ...)</pre>
Return Value	None.		

Python Syntax	EditSources(<acData>, <capData>, <dcData>)
Python Example	<pre>oModule.EditSources ([</pre>

```
"NAME:AC",
"Value Type:=" , "A",
[
  "NAME:Source Names",
  "Port2",
  "Port2#ref
],
[
  "NAME:Source Magnitudes",
  "0A",
  "1A"
]
],
[
"NAME:Cap",
"Value Type:=" , "N",
[
  "NAME:Source Names",
  "Port1_SINK",
  "Port1#ref_SINK"
],
[
  "NAME:Source Magnitudes",
  "0V",
  "1V"
],
[
  "NAME:Source Phases",
  "0deg",
  "1deg"
]
```

	<pre> ue Type:=" , "A", ["NAME:Source Names", "Port2", "Port2#ref"], ["NAME:Source Magnitudes", "0A", "1A"]]) </pre>

VB Syntax	EditSources <acData>, <capData>, <dcData>
VB Example	<pre> oModule.EditSources Array("NAME:AC", "Value Type:=", "A", Array("NAME:Source Names", "Port2", "Port2#ref"), Array("NAME:Source Magnitudes", "0A", "1A")), Array("NAME:Cap", "Value Type:=", "N", Array("NAME:Source Names", _ "Port1_SINK", "Port1#ref_SINK"), Array("NAME:Source Magnitudes", "0V", "1V"), Array ("NAME:Source Phases", _ "0deg", "1deg")), Array("NAME:DC", "Value Type:=", "A", Array("NAME:Source Names", _ "Port2", "Port2#ref"), Array("NAME:Source Magnitudes", "0A", "1A")) </pre>

ExportEigenmodes

Exports a tab delimited table of Eigenmodes in HFSS. Not in HFSS-IE.

UI Access	N/A

Parameters	Name	Type	Description
	<SoluName>	String	Name of specified solution.
	<DesignVariation>	String	Design variation key.
	<FileName>	String	Output file name with path.
Return Value	None.		

Python Syntax	ExportEigenmodes(<SoluName>, <DesignVariation>, <FileName>)
Python Example	<pre>oModule.ExportEigenmodes("Setup1 : LastAdaptive", "", "C:\mydir\myeigenmode.eig")</pre>

VB Syntax	ExportEigenmodes <SoluName>, <DesignVariation>, <FileName>
VB Example	<pre>oModule.ExportEigenmodes _ "Setup1 : LastAdaptive", "", _ "C:\mydir\myeigenmode.eig"</pre>

ExportForHSpice

Exports matrix solution data to a file in a format suitable for HSpice analysis. Available only for Driven Terminal solution types with ports. Output in an appropriate format will be generated for each of the non-empty file names provided.

UI Access	N/A

Parameters	Name	Type	Description
	<DesignVariation>	String	Design variation key.
	<SoluSelections>	Array	Array of selected solutions.
	<SpiceType>	Integer	1 - HSpice.
	<BandWidth>	Integer	0 - Low (narrow) band width.
	<FWSFile>	String	Full wave spice file name.
	<LumpedElementFile>	String	Lumped element file name.
	<PoleZeroSpiceFile>	String	Pole zero spice file name.
	<PoleZeroMatlabFile>	String	Pole zero Matlab file name.
	<PartialFractionFile>	String	Partial fraction file name.
	<FittingError>	Double	The accuracy to use in fitting the pole zero model.
	<MinimumOrder>	Integer	Minimum number of poles in rational function expansion.
	<MaximumOrder>	Integer	Maximum number of poles in rational function expansion.
	<UseCommonGround>	Integer	Optional. 1 to use common ground, 0 otherwise.
	<EnforcePassivity>	Integer	Optional. 1 to enforce passivity, 0 otherwise.
Return Value	None.		
Python Syntax	ExportForHSpice(<DesignVariation>, <SoluSelections>, <SpiceType>, <BandWidth>, <FWSFile>, <LumpedElementFile>, <PoleZeroSpiceFile>, <PoleZeroMatlabFile>, <PartialFractionFile>, <FittingError>, <MinimumOrder>, <MaximumOrder>, [Optional <UseCommonGround>], [Optional <EnforcePassivity>])		
Python Example	<pre>oModule.ExportForHSpice("width='2in'", ["Setup1:Sweep1"], 1, 0, "c:\mydir\Sweep1.fws", "", "", "", "", .005, 20, 200)</pre>		
VB Syntax	ExportForHSpice <DesignVariation>, <SoluSelections>, <SpiceType>, <BandWidth>, <FWSFile>, <LumpedElementFile>, <PoleZeroSpiceFile>, <PoleZeroMatlabFile>, <PartialFractionFile>, <FittingError>, <MinimumOrder>, <MaximumOrder>, [Optional <UseCommonGround>], [Optional <EnforcePassivity>]		

VB Example	<pre>oModule.ExportForHSpice "width='2in'", _ Array("Setup1:Sweep1"), 1, 0, _ "c:\mydir\Sweep1.fws", "", "", "", "", _ .005, 20, 200</pre>

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. <pre>Array(<SolnSelector>, <SolnSelector>, ...)</pre> 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 syntax, but ignored if DoRenorm is false.
	<DataType>	String	Optional. Type: "S", "Y", or "Z". The matrix to export.
	<pass>	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.
	<ComplexFormat>	Integer	Optional. Type: "0", "1", or "2" The format to use for the exported data. 0 = Magnitude/Phase. 1 = Real/Imaginary. 2 = db/Phase.
	<Precision>	Integer	Optional. Touchstone number of digits precision. Default if not specified is 15.
	<UseExportFreqs>	Boolean	Specifies whether to use export frequencies.
	<IncludeGammaComments>	Boolean	Specifies whether to include Gamma and Impedance comments.
	<SupportNonStdExport>	Boolean	Specifies whether to support non-standard Touchstone extensions for mixed reference impedance.
Return Value	None.		

Python Syntax	ExportNetworkData(<DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>, <FreqsArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <Sup-

	<code>portNonStdExport>))</code>
Python Example	<pre>oModule.ExportNetworkData("", ["Setup1:Sweep1"], 3, "C://Documents/package_ HFSSDesign1.s4p", ['all'], True, 50, "S", -1, 0, 15, True, True, True</pre>

VB Syntax	<code>ExportNetworkData <DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>, <FreqsArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <SupportNonStdExport>]</code>
VB Example	<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

Exports matrix solution data to a file in neutral model format. Available only for Driven solution types with ports. Variables can be held constant by setting their values in the variation field. For example: "length='50mm' width='30mm'". All other independent variables will be treated as NMF parameters.

UI Access	N/A		
Parameters	Name	Type	Description
	<sourceName>	Array	Array of selected solutions.
	<OutFile>	String	Full path to the file to write out, including the .nmf extension
	<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.
	<DesignVariationKey>	String	Design variation.
	<DoRenorm>	Boolean	Specifies whether to renormalize the data before export.
	<RenormImped>	Double	Real impedance value in ohms, for renormalization. Required in syntax, but ignored if DoRenorm is false.
	<Pass>	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.
	<ComplexFormat>	Integer	Optional. The format to use for the exported data. Options are 0 = Magnitude/Phase 1 = Real/Imaginary. 2 = db/Phase.
	<Precision>	Integer	Optional. Number of digits precision. If not specified, the default is 6.
Return Value	None		

Python Syntax	ExportNMFData(<sourceName>, <OutFile>, <FreqsArray>, <DesignVariationKey>, <DoRenorm>, <RenormImped>, <Pass>, <ComplexFormat>, <Precision>)
Python Example	oModule.ExportNMFData(["Setup1:Sweep1"], "c:\mydir\out.nmf", ["all"], "", False, 0)

VB Syntax	ExportNMFData <sourceName>, <OutFile>, <FreqsArray>, <DesignVariationKey>, <DoRenorm>, <RenormImped>, <Pass>, <ComplexFormat>, <Precision>
VB Example	oModule.ExportNMFData Array("Setup1:Sweep1"), _ "c:\mydir\out.nmf", Array("all"), "", false, 0

FFTONReport

Performs an FFT on a selected report.

UI Access	Right-click on Results in the project tree, select Perform FFT On Report...		
Parameters	Name	Type	Description
	<ReportName>	String	Name of specified report.
	<WindowName>	String	Name of window to apply for FFT. Possible values are "Rectangular", "Tri", "Van Hann", "Hamming", "Blackman", "Lanczos", "Weber", "Welch".
	<Function>	String	Function to apply on transformation values. Possible values are "none", "ang_deg", "ang_rad", "arg", "cang_deg", "cang_rad", "dB", "dB1 normalize", "dB20normalize", "dBc", "im", "mag", "normalize", "re".
Return Value	None.		

Python Syntax	FFTONReport <ReportName>, <WindowName>, <Function>
Python Example	oModule.FFTONReport("XY Plot 1", "Rectangular", "dB")

VB Syntax	FFTONReport(<ReportName>, <WindowName>, <Function>)

VB Example	<code>oModule.FFTOnReport "XY Plot 1", "Rectangular", "dB"</code>

GetAdaptiveFreq

Obtains an adaptive frequency for a specified setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<SetupName>	String	Name of specified setup.
Return Value	Double frequency value.		

Python Syntax	<code>GetAdaptiveFreq(<SetupName>)</code>
Python Example	<code>oModule.GetAdaptiveFreq("Setup1")</code>

VB Syntax	<code>GetAdaptiveFreq <SetupName></code>
VB Example	<code>oModule.GetAdaptiveFreq "Setup1"</code>

GetAvailableVariations

Returns the available variation for a solution.

UI Access	N/A		
Parameters	Name	Type	Description
	<SoluName>	String	Name of specified solution.

Return Value	Array of strings containing available variations.

Python Syntax	GetAvailableVariations(<SoluName>)
Python Example	<code>oModule.GetAvailableVariations("Setup1 : LastAdaptive")</code>

VB Syntax	GetAvailableVariations <SoluName>
VB Example	<code>oModule.GetAvailableVariations "Setup1 : LastAdaptive"</code>

GetExcitationScaling

Gets source scaling parameters.

UI Access	N/A		
Parameters	Name	Type	Description
	<portName>	String	Port ID.
	<Index>	Integer	Optional. Port/mode/terminal/eigenmode index, if present.
Return Value	Array of strings containing source scaling parameters.		

Python Syntax	GetExcitationScaling(<portName>, <portIndex>)
Python Example	<code>oModule.GetExcitationScaling("Port1", 2)</code>

VB Syntax	GetExcitationScaling(<portName>, <portIndex>)
VB Example	<code>oModule.GetExcitationScaling("Port1", 2)</code>

GetISolutionVersionID

Use: To obtain the solution ID to help track solution validity.

Syntax: GetISolutionVersionID(BSTR fullSolutionName)

Return Value: Returns a solution ID.

Parameters: None

VB Example:

```
versionID = oModule.GetISolutionVersionID(BSTR fullSolutionName)
```

GetSolveRangeInfo

Determines the frequency range of a particular simulation setup. For fast sweeps and interpolating sweeps this command returns the start and stop frequencies. For discrete sweeps, it returns a list of frequencies. For an adaptive solution, it returns the adaptive frequency.

UI Access	N/A		
Parameters	Name	Type	Description
	<SoluName>	String	Name of specified solution.
	<IncludeRangeFieldSaved>	Boolean	Optional. Specify whether to only include the range with field saved.
Return Value	Array of strings containing frequency values.		

Python Syntax	<code>GetSolveRangeInfo(<SoluName>, [Optional <IncludeRangeFieldSaved>])</code>
Python Example	<code>oModule.GetSolveRangeInfo("Setup1:Sweep1")</code>

VB Syntax	<code>GetSolveRangeInfo <SoluName>, [Optional <IncludeRangeFieldSaved>]</code>
VB Example	<code>oModule.GetSolveRangeInfo "Setup1:Sweep1"</code>

GetSourceContexts

Obtains sources currently enabled as context in the Edit Sources dialog Source Context tab.

UI Access	N/A
Parameters	None.
Return Value	Array of enabled source names

Python Syntax	<code>GetSourceContexts()</code>
Python Example	<code>oModule.GetSourceContexts()</code>

VB Syntax	<code>GetSourceContexts</code>
VB Example	<code>oModule.GetSourceContexts</code>

GetSourceData (Layout Editor)

Takes a source name as input and returns a VARIANT array containing a data block of the source data.

Command: None

Syntax: GetSourceData

Return Value: Type: Array of i/o block data

Parameters: <SourceName>

Type: string

VB Example: Dim sourceDataArr

```
sourceDataArr = oDesign.GetSourceData ("SourceName")
```

GetValidISolutionList

Gets all available solution names that exist in a design.

UI Access	N/A		
Parameters	Name	Type	Description
	<IncludeImportedSolutions>	Boolean	Optional. Specifies whether to include imported solutions. Default is false.
Return Value	Array of strings containing solution names.		

Python Syntax	GetValidISolutionList([Optional <IncludeImportedSolutions>])
Python Example	oModule.GetValidISolutionList()

VB Syntax	GetValidISolutionList [Optional <IncludeImportedSolutions>]
VB Example	<code>oModule.GetValidISolutionList</code>

HasFields

Determines if fields exist for a particular solution.

UI Access	N/A		
Parameters	Name	Type	Description
	<SoluName>	String	Name of specified solution.
	<DesignVar>	String	Design variation.
Return Value	Integer (1 - True, 0 - False).		

Python Syntax	HasFields(<SoluName>, <DesignVar>)
Python Example	<code>oModule.HasFields("Setup1:Sweep1", "x_size=2mm")</code>

VB Syntax	HasFields <SoluName>, <DesignVar>
VB Example	<code>oModule.HasFields "Setup1:Sweep1", "x_size=2mm"</code>

HasMatrixData

Determines if matrix data exists for a particular solution.

UI Access	N/A		
Parameters	Name	Type	Description
	< <i>SoluName</i> >	String	Name of specified solution.
	< <i>DesignVar</i> >	String	Design variation.
Return Value	Integer (1 - True, 0 - False).		

Python Syntax	HasMatrixData(< <i>SoluName</i> >, < <i>DesignVar</i> >)		
Python Example	oModule.HasMatrixData("Setup1:Sweep1", "")		

VB Syntax	HasMatrixData < <i>SoluName</i> >, < <i>DesignVar</i> >		
VB Example	oModule.HasMatrixData "Setup1:Sweep1", ""		

HasMesh

Determines if a current mesh exists for a particular simulation setup, not including the initial mesh.

UI Access	N/A		
Parameters	Name	Type	Description
	< <i>DesignName</i> >	String	Name of specified design.
	< <i>DesignVar</i> >	String	Design variation.
Return Value	Integer (1 - True, 0 - False).		

Python Syntax	HasMesh(<DesignName>, <DesignVar>)
Python Example	oModule.HasMesh("Setup1", "")

VB Syntax	HasMesh <DesignName>, <DesignVar>
VB Example	oModule.HasMesh "Setup1", ""

ImportSolution

Imports a matrix solution in HFSS, which can then be used in creating reports or in the display of matrix data. The imported solution need not have the same characteristics as the current design. Imported terminal data that meets the required criteria can be used for full-wave Spice export. Not used in HFSS-IE.

UI Access	Right-click on Results > Import Solutions... , click on Import Solution... in the opened dialog window.		
Parameters	Name	Type	Description
	<FileName>	String	Location of the source data. The type of the data file will be determined strictly by its file extension. Supported types are Touchstone (.sNp or .yNp or .zNp or .tou), and Ansoft Designer (.flp).
	<ImportName>	String	Identifying name to use for the import, analogous to solution setup name.
	<SoluArray>	Array	The names of the solutions selected for import from the file. The only import format supporting multiple solutions in one file is HFSS8.x format.
Return Value	None.		

Python Syntax	ImportSolution(<FileName>, <ImportName>, <SoluArray>)

Python Example	<pre>oModule.ImportSolution("c:\mydir\in.s2p", "MeasuredData", ["Sweep1"])</pre>

VB Syntax	ImportSolution <FileName>, <ImportName>, <SoluArray>
VB Example	<pre>oModule.ImportSolution _ "c:\mydir\in.s2p", _ "MeasuredData", Array("Sweep1")</pre>

ImportTable

Imports a data table for use in plotting reports in HFSS. Not used in HFSS-IE. The table can have multiple independent real-valued columns of data, and multiple dependent real- or complex-valued columns of data. The data supported imports are either tab delimited format (.tab) or comma delimited format (.csv). The first row may contain column names. Complex data columns are inferred from the column data format. In tab delimited format, "(double, double)" denotes a complex number. In comma delimited format, "(double, double)" denotes a complex number.

UI Access	Right-click on Results > Import Solutions... , click on Import Table... in the opened dialog window.		
Parameters	Name	Type	Description
	<FileName>	String	Location of the source data.
	<ImportName>	String	Identifying name to use for the import, analogous to solution setup name.
	<TableName>	String	Identifying name for the table, analogous to solution name.
	<IsReallmag>	Boolean	Whether to use real/imag to interpret data for any complex column. If false, then use mag/phase(degrees).
	<ContainsSoluData>	Boolean	Whether contains solution data.
	<ColNames>	Array	Non-empty array used only if intend to override the column names obtained from the table data file, in which case all column names are required.
	<ColIndepFlags>	Array	Indicates which columns are independent. If this is the empty array, the

			default is that only the first column is independent. If this is the non- empty array, a flag must be present for every column.
Return Value	None.		

Python Syntax	ImportTable(<FileName>, <ImportName>, <TableName>, <IsReallmag>, <ContainsSoluData>, <ColNames>, <ColIndepFlags>)		
Python Example	<pre>oModule.ImportTable("c:\mydir\mytable.tab", "ImportData", "Measurements", True, True, [], [True, True, False, False, False])</pre>		

VB Syntax	ImportTable <FileName>, <ImportName>, <TableName>, <IsReallmag>, <ContainsSoluData>, <ColNames>, <ColIndepFlags>		
VB Example	<pre>oModule.ImportTable "c:\mydir\mytable.tab", _ "ImportData", "Measurements", true, true, _ Array(), Array(true, true, false, false, false)</pre>		

IsFieldAvailableAt

Determines if a field solution exists for a particular frequency in a simulation.

UI Access	N/A

Parameters	Name	Type	Description
	<SoluName>	String	Name of specified solution.
	<DesignVar>	String	Design varitaiton.
	<Freq>	Double	Specified frequency.
Return Value	Integer (1 - True, 0 - False).		

Python Syntax	IsFieldAvailableAt(<SoluName>, <DesignVar>, <Freq>)
Python Example	<code>oModule.IsFieldAvailableAt("Setup1:Sweep1", "", 100.0)</code>

VB Syntax	IsFieldAvailableAt <SoluName>, <DesignVar>, <Freq>
VB Example	<code>oModule.IsFieldAvailableAt "Setup1:Sweep1", "", 100.0</code>

ListMatchingVariations

Gets a list of solved variations that include the specified variable values.

UI Access	N/A		
Parameters	Name	Type	Description
	<SoluName>	String	Name of specified solutions.
	<MatchVarNames>	Array	Array of matching variable names.
	<MatchValStrings>	Array	Array of matching variable value strings including units.
Return Value	Array of strings corresponding to solved variations. The match variables may be a partial set of design variables and the match values are one per variable in the same order as the variables.		

Python Syntax	ListMatchingVariations(<SoluName>, <MatchVarNames>, <MatchValStrings>)
Python Example	<pre>oModule.ListMatchingVariations("Setup1:LastAdaptive", ["x_size", "y_size"], ["2mm", "1mm"])</pre>

VB Syntax	ListMatchingVariations <SoluName>, <MatchVarNames>, <MatchValStrings>
VB Example	<pre>oModule.ListMatchingVariations _ "Setup1 :LastAdaptive", _ Array("x_size", "y_size"), Array("2mm", "1mm")</pre>

ListValuesOfVariable

Gets the values of a specified variable corresponding to the solved variations.

UI Access	N/A		
Parameters	Name	Type	Description
	<SoluName>	String	Name of specified solution.
	<VariableName>	String	Name of specified variable.
Return Value	Array of double precision values in SI units interpreted as the specified variable corresponding to the solved variations.		

Python Syntax	ListValuesOfVariable(<SoluName>, <VariableName>)
Python Example	oModule.ListValuesOfVariable("Setup1:LastAdaptive", "x_size")

VB Syntax	ListValuesOfVariable <SoluName>, <VariableName>
VB Example	oModule.ListValuesOfVariable "Setup1:LastAdaptive", "x_size"

SetQ3DSATNetChanges (Layout Editor)

Edits nets in a Q3D setup, including combining and changing net states.

UI Access	From the Project Manager window, right-click a Q3D setup and select Q3D Net Settings .		
Parameters	Name	Type	Description
	<SetupName>	String	The name of the setup.
	<Q3DSATNetChangesArray>	Array	An array that contains these parameters in order: • <SATFile> • <NetGroupsArray> This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.
	<SATFile>	String	Directory path to the SAT file.
	<NetGroupsArray>	Array	An array that contains these parameters in order: • <NAME:NetGroups> • <NAMEArray> The parameter <NAMEArray> may repeat, as necessary, for each user-selected change (e.g., changing net states, combining nets, et cetera). This array's placeholder does not appear in the syntax or examples

			below but instead has its contents listed.
	<NAME:NetGroups>	String	Identifies the contents of the array (i.e., NetGroups).
	<NAMEArray>	Array	An array that contains these parameters in order: • <NAME> • <Name> • <NetArray> • <NetType> This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.
	<NAME>	String	Identifies the net or net group being created or modified (e.g., NAME:Nets3).
	<Name>	String	Identifies the net or net group being created or modified (e.g., Nets3).
	<NetsArray>	Array	An array that contains the parameter <NAME:Nets> and then either a list of nets to combine, or a single restatement of the net name.
	<NAME:Nets>	String	Identifies the contents of the array (i.e., Nets).
	<Nets>	String	A list of nets to modify combine, or the name of the net or net group being modified.
	<NetType>	String	The state of the net being created or modified (i.e., REGULAR, FLOATING, or GROUND).
Return Value	None		

Python Syntax	<code>oProject.SetQ3DSATNetChanges(<SetupName>, [<SATFile>, [<NAME:NetGroups>, [<NAME>, <Name>], [<NetType>]]])</code>
Python Example	<code>oModule.SetQ3DSATNetChanges("Q3D1",</code>

```
[  
  "SATFile:=" , "C:/User-  
s/USERNAME/AppData/Local/Temp/ZC38E33379616974771197.aedtresults/Test_Design/-  
model.sat",  
  [  
    "NAME:NetGroups",  
    [  
      "NAME:Nets3",  
      "Name:=" , "Nets3",  
      [  
        "NAME:Nets",  
        "Nets1",  
        "Nets2"  
      ],  
      "NetType:=" , "SIGNAL"  
    ],  
    [  
      "NAME:Nets3",  
      "Name:=" , "Nets3",  
      [  
        "NAME:Nets",  
        "NETS3"
```


13 - 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>

[AddMarker](#)

[AddMarkerToPlot](#)

[AddNamedExpression](#)

[CalcOp](#)

[CalcStack](#)

[CalculatorRead](#)

[CalculatorWrite](#)

[ChangeGeomSettings](#)

[ChangeProperty \(Marker\)](#)

[ClcEval](#)

[ClcMaterial](#)

[ClcMaterialValue](#)

[ClearAllMarkers](#)

[ClearAllNamedExpr](#)

[CopyNamedExprToStack](#)

[DeleteFieldPlot](#)

[DeleteMarker](#)

[DeleteNamedExpr](#)

[DeleteUneditablePlot](#)

[DoesNamedExpressionExists](#)

[EditSurfaceMeshSummaryData](#)

[EnterComplex](#)

[EnterComplexVector](#)

[EnterCoord](#)

[EnterEdge](#)

[EnterLine](#)

[EnterOutputVar](#)

[EnterPoint](#)

[EnterQty](#)

[EnterScalar](#)

[EnterScalarFunc](#)

[EnterSurf](#)

[EnterVector](#)

[EnterVectorFunc](#)

[EnterVol](#)

[ExportFieldPlot](#)

[ExportOnGrid \[Field Overlays\]](#)

[ExportPlotImageToFile](#)

[ExportPlotImageWithViewToFile](#)

[ExportSurfaceMeshSummary](#)

[ExportToFile](#)

[GetFieldFolderNames](#)

[GetFieldPlotNames](#)

[GetFieldPlotQuantityName](#)

[GetMeshPlotNames](#)

[GetTopEntryValue](#)

[HideAntennaParametersOverlay](#)

[HideRadiatedPlotOverlay](#)

[HidePolarPlot](#)

[LoadNamedExpressions](#)

ModifyInceptionParameters

[ReassignFieldPlot](#)

[RenamePlotFolder](#)

[SaveFieldsPlots](#)

[SaveNamedExpressions](#)

[SetFieldPlotSettings](#)

SetPlotsViewSolutionContext

[ShowPolarPlot](#)

[SaveFieldsPlots](#)

[UpdateAllFieldsPlots](#)

[UpdateQuantityFieldsPlots](#)

AddMarkerToPlot

Adds a marker to a trace on a named field plot.

UI Access	N/A		
Parameters	Name	Type	Description
	<Location>	Array	Array of strings containing X.Y, and Z coordinates for the marker.
	<PlotName>	String	Name of the field plot.
Return Value	None.		

Python Syntax	AddMarkerToPlot(<Location>, <PlotName>)
Python Example	<pre>oModule.AddMarkerToPlot (["0.290455877780914in", "-0.616900205612183in", "1.77635683940025e-015in"], "Mag_H1") oModule.AddMarkerToPlot (["-0.317279517650604in",</pre>

	"1.22481322288513in", "0in"], "Mag_H1")

VB Syntax	AddMarkerToPlot <Location>, <PlotName>
VB Example	<pre>oModule.AddMarkerToPlot _ Array("0.290455877780914in", _ "-0.616900205612183in", "1.77635683940025e-015in"), _ "Mag_E1" oModule.AddMarkerToPlot _ Array("-0.317279517650604in", _ "1.22481322288513in", "0in"), _ "Mag_E1"</pre>

AddNamedExpression

Creates a named expression using the expression at the top of the stack.

UI Access	Click Add in the Fields Calculator window.		
Parameters	Name	Type	Description
	<ExprName>	String	Name for the new named expression.
	<FieldType>	String	Type of field.
Return Value	None		

Python Syntax	AddNamedExpression(<ExprName>, <FieldType>)
Python Example	<code>oModule.AddNamedExpression("Mag_JxE", "Fields")</code>

VB Syntax	AddNamedExpression <ExprName>, <FieldType>
VB Example	<code>oModule.AddNamedExpression "Mag_JxE", "Fields"</code>

```
oModule.AddNamedExpression("new2", "Fields",
```

CalcOp

Performs a calculator operation.

UI Access	Operation commands like Mag , + , etc.		
Parameters	Name	Type	Description
	<OpString>	String	The text on the corresponding calculator button.
Return Value	None.		

Python Syntax	CalcOp(<OpString>)
Python Example	<code>oModule.CalcOp("+")</code>

VB Syntax	CalcOp <OpString>

VB Example	<code>oModule.CalcOp "+"</code>

CalcStack

Performs an operation on the stack.

UI Access	Stack operation buttons such as Push and Pop .		
Parameters	Name	Type	Description
	<OpString>	String	The text on the corresponding calculator button.
Return Value	None.		

Python Syntax	<code>CalcStack(<OpString>)</code>
Python Example	<code>oModule.CalcStack("push")</code>

VB Syntax	<code>CalcStack <OpString></code>
VB Example	<code>oModule.CalcStack "push"</code>

CalculatorRead

Gets a register file and applies it to the calculator stack.

UI Access	Click Read... in the Fields Calculator dialog.		
Parameters	Name	Type	Description
	<FileName>	String	Path to and including name of input register file.
	<SoluName>	String	Specified solution to read in.

	<FieldType>	String	Type of specified field.
	<VarArray>	Array	Array of variable names, value pairs.
Return Value	None.		

Python Syntax	CalculatorRead(<FileName>, <SoluName>, <FieldType>, <VarArray>)
Python Example	<pre>oModule.CalculatorRead("c:\example.reg", "Setup1: LastAdaptive", "Fields", ["Freq:=", "10GHz", "Phase:=", "0deg"])</pre>

VB Syntax	CalculatorRead <FileName>, <SoluName>, <FieldType>, <VarArray>
VB Example	<pre>oModule.CalculatorRead _ "c:\example.reg", _ "Setup1: LastAdaptive", "Fields", _ Array("Freq:=", "10GHz", "Phase:=", "0deg")</pre>

CalculatorWrite

Writes contents of top register to file.

UI Access	Click Write... in the Fields Calculator window.

Parameters	Name	Type	Description
	<OutputFilePath>	String	Path to and including name of output register file.
	<SoluNameArray>	Array	Array of strings containing solution names.
	<VarArray>	Array	Array of variable names, value pairs.
Return Value	None		

Python Syntax	CalculatorWrite(<OutputFilePath>, <SoluNameArray>, <VarArray>)
Python Example	<pre>oModule.CalculatorWrite("C:\test.reg", ["Solution:=", "Setup1 : LastAdaptive"], ["Freq:=", "1GHz", "Phase:=", "0deg"])</pre>

VB Syntax	CalculatorWrite <OutputFilePath>, <SoluNameArray>, <VarArray>
VB Example	<pre>oModule.CalculatorWrite "C:\test.reg", _ Array("Solution:=", "Setup1 : LastAdaptive"), _ Array("Freq:=", "1GHz", "Phase:=", "0deg")</pre>

ChangeProperty

Changes the properties of a field plot marker.

UI Access	Select a field plot marker in the marker table.

Parameters	Name	Type	Description
	<propertyArgs>	Array	Structured array.
Return Value	None.		

Python Syntax	ChangeProperty(<propertyArgs>)
Python Example	<pre>oDesign.ChangeProperty(["NAME:AllTabs", ["NAME:FieldPlotMarkerDataTab", ["NAME:PropServers", "FieldsReporter:FieldsPlotMarker:m1"], ["NAME:ChangedProps", ["NAME:Name", "Value:=" , "Temp"]]]])</pre>

```

    ])
oDesign.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:FieldPlotMarkerDataTab",
        [
            "NAME:PropServers",
            "FieldsReporter:FieldsPlotMarker:Temp"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Display Value",
                "Value:=", , True
            ]
        ]
    ]
])
oDesign.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:FieldPlotMarkerDataTab",
        [
            "NAME:PropServers",
            "FieldsReporter:FieldsPlotMarker:Temp"
        ],
        [
            "NAME:ChangedProps",

```

```
ChangedProps",
    [
        "NAME:Display Name",
        "Value:=", True
    ]
]
])
oDesign.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:FieldPlotMarkerDataTab",
        [
            "NAME:PropServers",
            "FieldsReporter:FieldsPlotMarker:Temp"
        ],
        [
            "NAME:ChangedProps",
            [
                "NAME:Display Units",
                "Value:=", True
            ]
        ]
    ]
]
])
oDesign.ChangeProperty(
[
    "NAME:AllTabs",
    [
        "NAME:FieldPlotMarkerDataTab",
```


VB Syntax	ChangeProperty <propertyArgs>
VB Example	<pre> oDesign.ChangeProperty Array("NAME:AllTabs", Array("NAME:FieldPlotMarkerDataTab", Array("NAME:PropServers", _ "FieldsReporter:FieldsPlotMarker:m1"), Array("NAME:ChangedProps", Array("NAME:Display Value", "Value:=", _ true)))) oDesign.ChangeProperty Array("NAME:AllTabs", Array("NAME:FieldPlotMarkerDataTab", Array("NAME:PropServers", _ "FieldsReporter:FieldsPlotMarker:m1"), Array("NAME:ChangedProps", Array("NAME:Name", "Value:=", _ "Temp")))) oDesign.ChangeProperty Array("NAME:AllTabs", Array("NAME:FieldPlotMarkerDataTab", Array("NAME:PropServers", _ "FieldsReporter:FieldsPlotMarker:Temp"), Array("NAME:ChangedProps", Array("NAME:Display Name", "Value:=", _ false)))) oDesign.ChangeProperty Array("NAME:AllTabs", Array("NAME:FieldPlotMarkerDataTab", Array("NAME:PropServers", _ "FieldsReporter:FieldsPlotMarker:Temp"), Array("NAME:ChangedProps", Array("NAME:Display Name", "Value:=", _ true)))) oDesign.ChangeProperty Array("NAME:AllTabs", Array("NAME:FieldPlotMarkerDataTab", Array("NAME:PropServers", _ "FieldsReporter:FieldsPlotMarker:Temp"), Array("NAME:ChangedProps", Array("NAME:Color", "R:=", _ 255, "G:=", 255, "B:=", 255)))) oDesign.ChangeProperty Array("NAME:AllTabs", Array("NAME:FieldPlotMarkerDataTab", Array("NAME:PropServers", _ "FieldsReporter:FieldsPlotMarker:Temp"), Array("NAME:ChangedProps", Array("NAME:Symbol Size Scale", "Value:=", _ "1")))) </pre>

ChangeGeomSettings

Changes the line discretization setting.

UI Access	In the Fields Calculator dialog box, click on Geom Settings...		
Parameters	Name	Type	Description
	<LineDiscr>	Integer	Line discretization value.
Return Value	None.		

Python Syntax	ChangeGeomSettings(<LineDiscr>)
Python Example	oModule.ChangeGeomSettings(500)

VB Syntax	ChangeGeomSettings <LineDiscr>
VB Example	oModule.ChangeGeomSettings 500

ClcEval

Evaluates the expression at the top of the stack using the provided solution name and variable values.

UI Access	Click Eval in the Fields Calculator dialog.		
Parameters	Name	Type	Description
	<SoluName>	String	Name of specified solution.
	<VarArray>	Array	Array of variable name, value pairs.

	<i><FieldType></i>	String	Optional. Type of specified field.
Return Value	None		

Python Syntax	ClcEval(<SoluName>, <VarArray>, [Optional <FieldType>])		
Python Example	<pre>oModule.ClcEval("Setup1: LastAdaptive", ["Freq:=", "10GHz", "Phase:=", "0deg"])</pre>		

VB Syntax	ClcEval <SoluName>, <VarArray>, [Optional <FieldType>]		
VB Example	<pre>oModule.ClcEval "Setup1: LastAdaptive", _ Array ("Freq:=", "10GHz", _ "Phase:=", "0deg")</pre>		

ClcMaterial

Performs a material operation on the top stack element.

UI Access	Click Matl... in the Fields Calculator dialog.		
Parameters	Name	Type	Description
	<i><MatString></i>	String	The material property to apply.
	<i><OpString></i>	String	Name of operation. Possible values are "mult", or "div".
Return Value	None.		

Python Syntax	<code>ClcMaterial(<MatString>, <OpString>)</code>
Python Example	<code>oModule.ClcMaterial("Permeability (mu)" "mult")</code>

VB Syntax	<code>ClcMaterial <MatString>, <OpString></code>
VB Example	<code>oModule.ClcMaterial "Permeability (mu)" "mult"</code>

ClearAllNamedExpr

Clears all user-defined named expressions from the list.

UI Access	Click ClearAll in the Fields Calculator dialog.
Parameters	None.
Return Value	None.

Python Syntax	<code>ClearAllNamedExpr()</code>
Python Example	<code>oModule.ClearAllNamedExpr()</code>

VB Syntax	<code>ClearAllNamedExpr</code>
VB Example	<code>oModule.ClearAllNamedExpr</code>

CopyNamedExprToStack

Copies the named expression selected to the calculator stack.

UI Access	Select a named expression and then click Copy to stack .		
Parameters	Name	Type	Description
	<ExprName>	String	Name of the expression to be copied to the top of the stack.
Return Value	None.		

Python Syntax	CopyNamedExprToStack(<ExprName>)
Python Example	<code>oModule.CopyNamedExprToStack("Mag_JxE")</code>

VB Syntax	CopyNamedExprToStack <ExprName>
VB Example	<code>oModule.CopyNamedExprToStack "Mag_JxE"</code>

CreateFieldPlot

Note:

Use in conjunction with ["GetGeometryIdsForNetLayerCombinations" on page 7-90](#) and ["GetGeometryIdsForAllNetLayerCombinations" on page 7-91](#).

- 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.

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"

A space is required on either side of the colon (:) character. If omitted, the plot will not be created.

QuantityName:

Type of plot to create. Possible values vary according to the solver.

Mesh plots: "Mesh"

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 thisflag 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' "

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 two 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, two nets, net names are "net1" and "net2", i.e., [Top, net1] pair, and [Top, net2] pair. (2) "Ground" layer and one net, net name is "GND", i.e., [Ground, GND] pair.

HFSS example with "Plot on surface" option checked:

```
"PlotGeomInfo:=", [1, "Surface", "LayerNetsExtFace", 2,  
"Top", 2, "net1", "net2", "Ground", 1, "GND"]
```

The command creates field plot on two 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, two nets, net names are "net1" and "net2", i.e., [Top, net1] pair, and [Top, net2] pair. (2) "Ground" layer and one net, net name is "GND", i.e., [Ground, GND] pair.

Limitations:

- 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>)
```

Python Example – E Magnitude Field Plot:

```
oModule.CreateFieldPlot(  
[  
    "NAME:Mag_E1",  
    "SolutionName:=", "Setup1 : LastAdaptive",
```

```
"QuantityName:="      , "Mag_E",
"PlotFolder:="        , "E Field1",
"UserSpecifyName:="    , 0,
"UserSpecifyFolder:="  , 0,
"IntrinsicVar:="       , "Freq='1GHz' Phase='0deg'",
"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]
]
], "Field")
```

VB 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" (VB)

```

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"

```


VB 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 – E Magnitude 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))), "Field"

```

Python Example – Electron Density Field 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=\'0.200000000000000001GHz\'

```

```

                                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")

```

VB 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"

```

DeleteFieldPlot

Deletes one or more field plots.

UI Access	Right-click on one filed plot, select Delete .		
Parameters	Name	Type	Description
	<NameArray>	Array	Array of strings containing the names of the plots to delete.
Return Value	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")

DeleteMarker[Fields Reporter]

Deletes one or more marker in the current field plot.

UI Access	Right-click Field Overlays > Plot Fields > Marker > Delete Marker .		
Parameters	Name	Type	Description
	<MarkerNames>	Array	Array of strings containing marker names.
Return Value	None.		

Python Syntax	DeleteMarker(<MarkerNames>)
Python Example	oModule.DeleteMarker(["m1", "m2"])

VB Syntax	DeleteMarker <MarkerNames>
VB Example	oModule.DeleteMarker Array("m1", "m2")

DeleteNamedExpr

Deletes the selected named expression from the list.

UI Access	Select a named expression and then click Delete .		
Parameters	Name	Type	Description
	<ExprName>	String	Name of specified named expression.
Return Value	None.		

Python Syntax	DeleteNamedExpr(<ExprName>)
Python Example	oModule.DeleteNamedExpr("Mag_JxE")

VB Syntax	DeleteNamedExpr <ExprName>
VB Example	oModule.DeleteNamedExpr "Mag_JxE"

DeleteUneditablePlot

Deletes one or more uneditable plots.

UI Access	N/A		
Parameters	Name	Type	Description
	<NameArray>	Array	Array of the plot names to delete.
Return Value	None.		

Python Syntax	DeleteUneditablePlot(<NameArray>)
Python Example	<code>oModule.DeleteUneditablePlot(["Mag_E1", "Vector_E1"])</code>

VB Syntax	DeleteUneditablePlot <NameArray>
VB Example	<code>oModule.DeleteUneditablePlot Array("Mag_E1", "Vector_E1")</code>

DoesNamedExpressionExists

Determines whether specified named expression exists.

UI Access	N/A		
Parameters	Name	Type	Description
	<ExpName>	String	Name of the specified named expression.

Return Value	Boolean: • 1 - named expression exists. • 0 - named expression does not exist.

Python Syntax	DoesNamedExpressionExists(<ExpName>)
Python Example	oModule.DoesNamedExpressionExists("Mag_JxE")

VB Syntax	DoesNamedExpressionExists <ExpName>
VB Example	oModule.DoesNamedExpressionExists "Mag_JxE"

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:ItemPer-  
Row", "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", </pre>

```
[
  "NAME:ItemPerRow",
    "EntityPerRow:="      , "annular_rng",
  [
    "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
  ]
],
[
  "NAME:ItemPerRow",
    "EntityPerRow:="      , "gap",
  [
```

	<pre>"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]]]]])</pre>

UI Access	Click Number , and then click Scalar . Complex option is selected.		
Parameters	Name	Type	Description
	<ComplexNum>	String	String of complex value. Ex. "1 + 2j".

Return Value	None.

Python Syntax	EnterComplex(<ComplexNum>)
Python Example	<code>oModule.EnterComplex("1 + 2 j")</code>

VB Syntax	EnterComplex <ComplexNum>
VB Example	<code>oModule.EnterComplex "1 + 2 j"</code>

EnterComplexVector

Enters a complex vector onto the stack.

UI Access	Click Number , and then click Vector. Complex option is selected.		
Parameters	Name	Type	Description
	<ComplexVector>	Array	Array of strings containing X, Y and Z complex values.
Return Value	None.		

Python Syntax	EnterComplexVector(<ComplexVector>)
Python Example	<code>oModule.EnterComplexVector ([</code>

	<pre>"1 + 2 j", "1 + 2 j", "1 + 2 j" l)</pre>

VB Syntax	EnterComplexVector <ComplexVector>
VB Example	<pre>oModule.EnterComplexVector _ Array("1 + 2 j",_ "1 + 2 j",_ "1 + 2 j")</pre>

EnterCoord

Enters a coordinate system defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Coord .		
Parameters	Name	Type	Description
	<CoordName>	String	Name of a coordinate system defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterCoord(<CoordName>)
Python Example	oModule.EnterCoord("Global")

VB Syntax	EnterCoord <CoordName>
VB Example	<code>oModule.EnterCoord "Global"</code>

EnterEdge

Enters an edge defined in the 3D Modeler editor.

UI Access	N/A		
Parameters	Name	Type	Description
	<EdgeName>	String	Name of an edge defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterEdge(<EdgeName>)
Python Example	<code>oModule.EnterPoint("Edge_1")</code>

VB Syntax	EnterEdge <EdgeName>
VB Example	<code>oModule.EnterPoint "Edge_1"</code>

EnterLine

Enters a line defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Line		
Parameters	Name	Type	Description
	<LineName>	String	Name of a line defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterLine(<LineName>)
Python Example	<code>oModule.EnterLine("Line1")</code>

VB Syntax	EnterLine <LineName>
VB Example	<code>oModule.EnterLine "Line1"</code>

EnterOutputVar

Enters Output Vars, only valid for Eigenmode problems.

UI Access	Click Geometry and then select Output Vars .		
Parameters	Name	Type	Description
	<VarName>	String	Name of the output vars. Only 'freq' is supported.
	<VarType>	String	Type of the output vars. 'Complex' type.
Return Value	None.		

Python Syntax	EnterOutputVar(<VarName>, <VarType>)

Python Example	<code>oModule.EnterOutputVar("Freq", "Complex")</code>

VB Syntax	<code>EnterOutputVar <VarName>, <VarType></code>
VB Example	<code>oModule.EnterOutputVar "Freq", "Complex"</code>

EnterPoint

Enters a point defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Point .		
Parameters	Name	Type	Description
	<PointName>	String	Name of a point defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	<code>EnterPoint(<PointName>)</code>
Python Example	<code>oModule.EnterPoint("Point1")</code>

VB Syntax	<code>EnterPoint <PointName></code>
VB Example	<code>oModule.EnterPoint "Point1"</code>

EnterQty

Enters a field quantity.

UI Access	Click Quantity , and then select from the list.		
Parameters	Name	Type	Description
	<FieldQty>	String	The field quantity to be entered onto the stack.
Return Value	None.		

Python Syntax	EnterQty(<FieldQty>)
Python Example	<code>oModule.EnterQty("E")</code>

VB Syntax	EnterQty <FieldQty>
VB Example	<code>oModule.EnterQty "E"</code>

EnterScalar

Enters a scalar onto the stack.

UI Access	Click Number and then click Scalar . Complex option not selected.		
Parameters	Name	Type	Description
	<Scalar>	Double	The real number to enter onto the stack.
Return Value	None.		

Python Syntax	EnterScalar(<Scalar>)
Python Example	<code>oModule.EnterScalar(3.14159265358979)</code>

VB Syntax	EnterScalar <Scalar>
VB Example	<code>oModule.EnterScalar 3.14159265358979</code>

EnterScalarFunc

Enters a scalar function.

UI Access	Click Function and then select Scalar .		
Parameters	Name	Type	Description
	<VarName>	String	Name of a variable to enter as a scalar function onto the stack.
Return Value	None.		

Python Syntax	EnterScalarFunc(<VarName>)
Python Example	<code>oModule.EnterScalarFunc("Phase")</code>

VB Syntax	EnterScalarFunc <VarName>

VB Example	<code>oModule.EnterScalarFunc "Phase"</code>

EnterSurf

Enters a surface defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Surface .		
Parameters	Name	Type	Description
	<SurfName>	String	Name of a surface defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	<code>EnterSurf(<SurfName>)</code>
Python Example	<code>oModule.EnterSurf("Rectangle1")</code>

VB Syntax	<code>EnterSurf <SurfName></code>
VB Example	<code>oModule.EnterSurf "Rectangle1"</code>

EnterVector

Enters a vector onto the stack.

UI Access	Click Number , and then click Vector . Complex option not selected.		
Parameters	Name	Type	Description
	<VectorArray>	Array	Array of strings containing X, Y and Z components of the vector.

Return Value	None.

Python Syntax	EnterVector(<VectorArray>)
Python Example	oModule.EnterVector([1.0, 1.0, 1.0])

VB Syntax	EnterVector <VectorArray>
VB Example	oModule.EnterVector Array(1.0, 1.0, 1.0)

EnterVectorFunc

Enters a vector function.

UI Access	Click Function and then select Vector .		
Parameters	Name	Type	Description
	<VarNames>	Array	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.
Return Value	None.		

Python Syntax	EnterVectorFunc(<VarNames>)
Python Example	oModuleEnterVectorFunc(["X", "Y", "Z"])

VB Syntax	EnterVectorFunc <VarNames>
VB Example	oModuleEnterVectorFunc Array("X", "Y", "Z")

EnterVol

Enters a volume defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Volume .		
Parameters	Name	Type	Description
	<VolumeName>	String	Name of a volume defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterVol(<VolumeName>)
Python Example	oModule.EnterVol ("Box1")

VB Syntax	EnterVol <VolumeName>
VB Example	oModule.EnterVol "Box1"

ExportFieldPlot

Exports a field plot to a file.

UI Access	N/A		
Parameters	Name	Type	Description

	<PlotName>	String	Name of the field plot to export.
	<ShowHeader>	Boolean	True - export field plot header to a file. False - export field plot.
	<FileName>	String	Name of file to save as, including the file path.
Return Value	None.		

Python Syntax	ExportFieldPlot(<PlotName>, <ShowHeader>, <FileName>)		
Python Example	<pre>oModule.ExportFieldPlot("Mag_E1", True, "C:/field_report.dsp")</pre>		

VB Syntax	ExportFieldPlot <PlotName>, <ShowHeader>, <FileName>		
VB Example	<pre>oModule.ExportFieldPlot "Mag_E1", _ true, "C:/field_report.dsp"</pre>		

ExportMarkerTable

Exports the marker table to a .csv or .tab file.

UI Access	[product] > Fields > Plot Fields > Marker > Export Marker Table .		
Parameters	Name	Type	Description
	<FileName>	String	Name of export file include path.

Return Value	None.

Python Syntax	ExportMarkerTable(<FileName>)
Python Example	<code>oModule.ExportMarkerTable("C:/work/FieldMarkerTable.csv")</code>

VB Syntax	ExportMarkerTable <FileName>
VB Example	<code>oModule.ExportMarkerTable "C:/work/FieldMarkerTable.csv"</code>

ExportOnGrid [Field Overlays]

Evaluates the top stack element at a set of points specified by a grid and exports the data to a file.

UI Access	Click Export , and then click On Grid .		
Parameters	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.

Return Value	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”

Python Syntax	<code>ExportOnGrid(<OutputFile>, <Min>, <Max>, <Spacing>, <SolnName>, <SolnParameters>, [<ExportOption>, "IncludePointsInOutput:=", <boolean>], "RefCSName:=", <CSName>, "PtsInSI:=", <boolean>, "FieldRefCS:=", <boolean>], "<CSType>", [<Offsets>, <ByCount>])</code>
Python Example	<pre>oModule.ExportOnGrid("C:\offset_grid_model_unit_ref.fld", ["-1mm", "16mm", "0mm"], ["1mm", "18mm", "1mm"], ["2mm", "2mm", "1mm"], "4500MHz : LastAdaptive", ["Freq:=", "4.5GHz", "Phase:=" , "0deg"], ["NAME:ExportOption", "IncludePtInOutput:=" , True, "RefCSName:=" , "offset", "PtsInSI:=" , False, "FieldInRefCS:=" , True], "Cartesian", ["0mm", "0mm", "0mm"], False</pre>

)

VB Syntax	ExportOnGrid(<OutputFile>, <Min>, <Max>, <Spacing>, <SolnName>, <SolnParameters>, [<ExportOption>, "IncludePointsInOutput:=", <boolean>], "RefCSName:=", <CSName>, "PtsInSI:=", <boolean>, "FieldRefCS:=", <boolean>], "<CSType>", [<Offsets>, <ByCount>])
VB Example	<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"), _ 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"), [</pre>

	<pre>"NAME:ExportOption", "IncludePtInOut":=" , True, "RefCSName:=" , "offset", "PtInSI:=" , False, "FieldInRefCS:=" , True], "Cartesian", Array("0mm", "0mm", "0mm"), False)</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.

UI Access	N/A		
Parameters	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.
Return Value	None.		

Python Syntax	<code>ExportPlotImageToFile(<FileName>, <FolderName>, <ItemName>, <SetViewTopDownDirectionByRCS>)</code>
Python Example	<pre>oModule.ExportPlotImageToFile ("C:\TestEPITF2.jpg", "", "Mag_E2", "RelativeCS1")</pre>

VB Syntax	<code>ExportPlotImageToFile <FileName>, <FolderName>, <ItemName>, <SetViewTopDownDirectionByRCS></code>
VB Example	<pre>oModule.ExportPlotImageToFile _ "C:\TestEPITF2.jpg", "", _ "Mag_E2", "RelativeCS1"</pre>

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.

	<table><tr><td><ViewOrientation></td><td>String</td><td>Optional. Name of orientation to use for plot.</td></tr></table>	<ViewOrientation>	String	Optional. Name of orientation to use for plot.
<ViewOrientation>	String	Optional. Name of orientation to use for plot.		
Return Value	Image file is exported to the specified path.			

Python Syntax	ExportPlotImageWithViewToFile(<FileName>, <PlotQuantityName>, <PlotItemName>, <PixelSizeX>, <PixelSizeY>, <ViewOrientation>)
Python Example	<pre>oModule.ExportPlotImageWithViewToFile("E:/MyDir/OptimToutput/magE2.gif", "E Field", "Mag_E2", 1920, 1080, "newot2")</pre>

VB Syntax	ExportPlotImageWithViewToFile <FileName>, <PlotQuantityName>, <PlotItemName>, <PixelSizeX>, <PixelSizeY>, <ViewOrientation>
VB Example	<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", "DesignVari-  
ationKey:=", _  
"Nominal", "ExportFileName:=", "D:\documents\surfaceMeshSummaryReport.csv")
```

Python Syntax	ExportSurfaceMeshSummary (<SolutionName>, <DesignVariationKey>, <ExportFileName>)
Python Example	<pre>oModule.ExportSurfaceMeshSummary(["SolutionName:=" , "Setup1 : LastAdaptive", "DesignVariationKey:=" , "Nominal", "ExportFileName:=" , "D:\\Program Files\\Docu- ments\\surfaceMeshSummaryReport.csv"])</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.

UI Access	Right-click on report name in the project tree and select Export Data .		
Parameters	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
	<UnitSpec>	String	For example, "kV, Mhz, yd"
	<UseTraceNumberFormat>	Boolean	"True", "False"
Return Value	None.		

Python Syntax	ExportToFile(<ReportName>, <FileName>)
Python Example	<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>

VB Syntax	ExportToFile <ReportName>, <FileName>
VB Example	<pre>oModule.ExportToFile "rETotal", "C:/Documents/rETotal.csv"</pre>

GetFieldFolderNames

Gets the folder names of the field plots.

UI Access	N/A
Parameters	None.
Return Value	Array of folder names.

Python Syntax	GetFieldFolderNames()
Python Example	<pre>oModule.GetFieldFolderNames()</pre>

VB Syntax	GetFieldFolderNames
VB Example	<pre>oModule.GetFieldFolderNames</pre>

GetFieldPlotNames

Gets the names of field overlay plots defined in a design.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing field plots names.

Python Syntax	GetFieldPlotNames()
Python Example	<code>oModule.GetFieldPlotNames()</code>

VB Syntax	GetFieldPlotNames
VB Example	<code>oModule.GetFieldPlotNames</code>

GetFieldPlotQuantityName

Gets the quantity name of a specified field plot.

UI Access	N/A		
Parameters	Name	Type	Description
	<PlotName>	String	Name of specified plot.
Return Value	String plot quantity name.		

Python Syntax	GetFieldPlotQuantityName(< <i>PlotName</i> >)
Python Example	<code>oModule.GetFieldPlotQuantityName("Mag_H1")</code>

VB Syntax	GetFieldPlotQuantityName < <i>PlotName</i> >
VB Example	<code>oModule.GetFieldPlotQuantityName "Mag_H1"</code>

GetMeshPlotNames

Gets the names of mesh plots defined in a design.

UI Access	N/A
Parameters	None..
Return Value	Array of plot name.

Python Syntax	GetMeshPlotNames
Python Example	<code>oModule.GetMeshPlotNames()</code>

VB Syntax	GetMeshPlotNames
VB Example	<code>oModule.GetMeshPlotNames</code>

GetTopEntryValue

Gets the value of the top entry of the calculator stack.

UI Access	N/A		
Parameters	Name	Type	Description
	<SolnName>	String	Name of specified solution.
	<VarVals>	Array	Array of variable name, value pairs.
Return Value	Array of strings containing top entry values.		

Python Syntax	GetTopEntryValue(<SolnName>, <VarVals>)
Python Example	<pre>oModule.GetTopEntryValue("Setup1:LastAdaptive", ["Freq:=", "1GHz", "Phase:=", "0deg", "x_size:=", "2mm"])</pre>

VB Syntax	GetTopEntryValue <SolnName>, <VarVals>
VB Example	<pre>oModule.GetTopEntryValue "Setup1:LastAdaptive", _ Array("Freq:=", "1GHz", "Phase:=", "0deg", _ "x_size:=", "2mm")</pre>

HideAntennaParametersOverlay

Hides a currently visible antenna parameters overlay.

UI Access	HFSS or HFSS-IE > Fields > Plot Fields > Radiation Field		
Parameters	Name	Type	Description
	<Name>	String	The name of the antenna parameters overlay.
Return Value	None.		

Python Syntax	HideAntennaParametersOverlay(<Name>)
Python Example	<pre>oModule.HideAntennaParameterOverlay("Antenna Parameter Overlay1")</pre>

VB Syntax	HideAntennaParametersOverlay <Name>
VB Example	<pre>oModule.HideAntennaParameterOverlay _ "Antenna Parameter Overlay1"</pre>

HidePolarPlot

Hides a currently visible polar plot.

UI Access	N/A		
Parameters	Name	Type	Description
	<PlotName>	String	Name of the polar plot.

Return Value	None.

Python Syntax	HidePolarPlot(<PlotName>)
Python Example	<code>oModule.HidePolarPlot("rE Plot 1")</code>

VB Syntax	HidePolarPlot <PlotName>
VB Example	<code>oModule.HidePolarPlot "rE Plot 1"</code>

HideRadiatedPlotOverlay

Hides overlay radiation fields plot.

UI Access	N/A		
Parameters	Name	Type	Description
	<PlotName>	String	Name of specified plot.
Return Value	None.		

Python Syntax	HideRadiatedPlotOverlay(<PlotName>)
Python Example	<code>oModule.HideRadiatedPlotOverlay("Gain Plot 1")</code>

VB Syntax	HideRadiatedPlotOverlay <PlotName>
VB Example	oModule.HideRadiatedPlotOverlay "Gain Plot 1"

LoadNamedExpressions

Loads a named expression definition from a saved file.

UI Access	In the Fields Calculator, click Load From... in the Library area.		
Parameters	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	rarray of strings containing the names of expression definitions to load from the file.
Return Value	None.		

Python Syntax	LoadNamedExpressions(<FileName>, <FieldType>, <NamedExpr>)
Python Example	<pre>oModule.LoadNamedExpressions ("C:\Ansoft\PersonalLib\smth.clc", "Fields", ["smoothedtemp"])</pre>

VB Syntax	LoadNamedExpressions <FileName>, <FieldType>, <NamedExpr>
VB Example	<pre>oModule.LoadNamedExpressions _ "C:\Ansoft\PersonalLib\smth.clc", _</pre>

	"Fields", Array("smoothedtemp")

ModifyFieldPlot

Modifies a plot definition.

UI Access	[product] > Fields > Modify Plot		
Parameters	Name	Type	Description
	<OriginalName>	String	Name of the plot to be modified.
	<PlotParams>	Array	Array containing modified settings.
Return Value	None.		

Python Syntax	ModifyFieldPlot(<OriginalName>, <PlotParams>)
Python Example	<pre>oModule.ModifyFieldPlot("Vector_E1" , ["NAME:Vector_E2", "SolutionName:=", "Setup1 : LastAdaptive", "QuantityName:=", "Vector_E", "PlotFolder:=", "E Field1", "UserSpecifyName:=", 0, "UserSpecifyFolder:=", 0, "IntrinsicVar:=", "Freq='1GHz' Phase='30deg'",</pre>

	<pre> "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]]] </pre>

VB Syntax	ModifyFieldPlot <OriginalName>, <PlotParams>
VB Example	<pre> oModule.ModifyFieldPlot "Vector_E1" ,_ Array("NAME:Vector_E2", _ "SolutionName:=", "Setup1 : LastAdaptive", _ "QuantityName:=", "Vector_E", _ "PlotFolder:=", "E Field1", _ "UserSpecifyName:=", 0, _ </pre>

	<pre>"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)))</pre>

ReassignFieldPlot

Reassigns a field plot.

UI Access	Right-click on a field plot > Reassign .		
Parameters	Name	Type	Description
	<PlotName>	String	Name of specified plot to reassign.
	<PlotParameterArray>	Array	See CreateFieldPlot

Return Value	None.

Python Syntax	ReassignFieldPlot(<PlotName>, <PlotParameterArray>)
Python Example	<pre>oModule.ReassignFieldPlot "Mag_H1", Array("NAME:Mag_H1", _ "SolutionName:=", "Setup1 : LastAdaptive", _ "UserSpecifyName:=", 0, _ "UserSpecifyFolder:=", 0, _ "QuantityName:=", "Mag_H", _ "PlotFolder:=", "H Field", _ "StreamlinePlot:=", False, "AdjacentSidePlot:=", False, "FullModelPlot:=", False, "IntrinsicVar:=", "Freq='\20GHz\' Phase='\0deg\'", "PlotGeomInfo:=", [1,"Surface","FacesList",1,"117"], "FilterBoxes:=", [0], ["NAME:PlotOnSurfaceSettings", "Filled:=", False, "IsoValType:=", "Fringe", "AddGrid:=", False, "MapTransparency:=", True,</pre>

```
"Refinement:=", 0,  
"Transparency:=", 0,  
"SmoothingLevel:=", 0,  
"ShadingType:=", 0,  
["NAME:Arrow3DSpacingSettings",  
    "ArrowUniform:=", True,  
    "ArrowSpacing:=", 0,  
    "MinArrowSpacing:=", 0,  
    "MaxArrowSpacing:=", 0  
],  
"GridColor:=", [255,255,255]  
],  
"EnableGaussianSmoothing:=", False  
])
```

VB Syntax	ReassignFieldPlot <PlotName>, <PlotParameterArray>
VB Example	<pre>oModule.ReassignFieldPlot "Mag_E1", _ Array("NAME:Mag_E1", "SolutionName:=", _ "Setup1 : LastAdaptive", "UserSpecifyName:=", 0, _</pre>

```

"UserSpecifyFolder:=", 0, "QuantityName:=", _
"Mag_E", "PlotFolder:=", "E Field", _
"StreamlinePlot:=", false, "AdjacentSidePlot:=", _
false, "FullModelPlot:=", false, "IntrinsicVar:=", _
"Freq=" & Chr(39) & "20GHz" & Chr(39) & " _
Phase=" & Chr(39) & "0deg" & Chr(39) & "", "PlotGeomInfo:=", Array( _
1, "Surface", "FacesList", 1, "149"), _
"FilterBoxes:=", Array(0), _
Array("NAME:PlotOnSurfaceSettings", "Filled:=", _
false, "IsoValType:=", "Fringe", _
"AddGrid:=", false, "MapTransparency:=", true, "Refinement:=", _
0, "Transparency:=", 0, "SmoothingLevel:=", 0, _
"ShadingType:=", 0, Array("NAME:Arrow3DSpacingSettings", "ArrowUniform:=", _
true, "ArrowSpacing:=", 0, _
"MinArrowSpacing:=", 0, "MaxArrowSpacing:=", 0), "GridColor:=", Array( _
255, 255, 255)), "EnableGaussianSmoothing:=", false)

```

RenameFieldPlot

Renames a plot.

UI Access	Right-click the plot you want to rename in the project tree, and then click Rename on the shortcut menu.		
Parameters	Name	Type	Description
	<OldName>	String	Original name of the plot.

	<table><tr><td><NewName></td><td>String</td><td>New name of the plot.</td></tr></table>			<NewName>	String	New name of the plot.
<NewName>	String	New name of the plot.				
Return Value	None.					

Python Syntax	RenameFieldPlot(<OldName>, <NewName>)
Python Example	<pre>oModule.RenameFieldPlot("Vector_E1", "Vector_E2")</pre>

VB Syntax	RenameFieldPlot <OldName>, <NewName>
VB Example	<pre>oModule.RenameFieldPlot _ "Vector_E1", "Vector_E2"</pre>

RenamePlotFolder

Renames a plot folder.

UI Access	Right-click a plot folder in the project tree, and then click Rename on the shortcut menu.		
Parameters	Name	Type	Description
	<OldName>	String	Original name of the folder.
	<NewName>	String	New name of the folder.
Return Value	None.		

Python Syntax	<code>RenamePlotFolder(<OldName>, <NewName>)</code>
Python Example	<pre>oModule.RenamePlotFolder("E Field", "Surface Plots")</pre>

VB Syntax	<code>RenamePlotFolder <OldName>, <NewName></code>
VB Example	<pre>oModule.RenamePlotFolder _ "E Field", "Surface Plots"</pre>

SaveFieldsPlots

Saves field plot(s) to a file.

UI Access	HFSS > Fields > Save As...		
Parameters	Name	Type	Description
	<PlotNames>	Array	Array of plot names.
	<FileName>	String	Name of the file to save as, including the file path.
Return Value	None.		

Python Syntax	<code>SaveFieldsPlots(<PlotNames>, <FileName>)</code>
Python Example	<pre>oModule.SaveFieldsPlots(["Mag_E1", "Mag_E2"], "C:/field_report.dsp")</pre>

VB Syntax	SaveFieldsPlots <PlotNames>, <FileName>
VB Example	<pre>oModule.SaveFieldsPlots Array("Mag_E1", "Mag_E2"), _ "C:/field_report.dsp"</pre>

SaveNamedExpressions

Saves a named expression definition to a file.

UI Access	In the Fields Calculator, click Save To... in the Library area.		
Parameters	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.
Return Value	None.		

Python Syntax	SaveNamedExpressions(<FileName>, <NamedExprs>, <Overwrite>)
Python Example	<pre>oModule.SaveNamedExpressions("C:\Ansoft\PersonalLib\smth.clc", ["smoothedtemp"], True)</pre>

VB Syntax	SaveNamedExpressions <FileName>, <NamedExprs>, <Overwrite>

VB Example	<pre>oModule.SaveNamedExpressions _ "C:\Ansoft\PersonalLib\smth.clc", _ Array("smoothedtemp"), true</pre>

SetFieldPlotSettings

Sets plot attributes.

UI Access	[product] > Fields > Modify Plot Attributes		
Parameters	Name	Type	Description
	<PlotName>	String	Name of the plot to modify.
	<PlotItemAttributes>	Array	Structured array.
			<pre>Array("NAME:FieldsPlotItemSettings", <PlotOnPointsSettings>, <PlotOnLineSettings>, <PlotOnSurfaceSettings>, <PlotOnVolumeSettings>) See description of CreateFieldPlot command for details.</pre>
Return Value	None.		

Python Syntax	SetFieldPlotSettings(<PlotName> <PlotItemAttributes>)
Python Example	<pre>oModule.SetFieldPlotSettings("Mag_E2", ["NAME:FieldsPlotItemSettings",</pre>


```
[ "NAME:PlotOnLineSettings",  
  [ "NAME:LineSettingsID",  
    "Width:=", 4,  
    "Style:=", "Cylinder"),  
    "IsoValType:=", "Tone",  
    "ArrowUniform:=", True,  
    "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]]  
)
```

VB Syntax	SetFieldPlotSettings <PlotName> <PlotItemAttributes>
VB Example	<pre> oModule.SetFieldPlotSettings "Mag_E2", _ Array("NAME:FieldsPlotItemSettings", _ Array("NAME:PlotOnLineSettings", _ Array("NAME:LineSettingsID", _ "Width:=", 4, _ "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))) </pre>

SetPlotFolderSettings

Sets the attributes of all plots in the specified folder.

UI Access	[product] > Fields > Modify Plot Attributes		
Parameters	Name	Type	Description
	<PlotFolderName>	String	Name of the folder with the attributes to modify.
	<PlotFolderAttributes>	Array	Structured array. <pre>Array("NAME:FieldsPlotSettings", "Real time mode:=", <bool>, <ColorMapSettings>, <Scale3DSettings>, <Marker3DSettings>, <Arrow3DSettings>)</pre>
	<ColorMapSettings>	Array	Structured array. <pre>Array("NAME:ColorMapSettings", "ColorMapType:=", <string Possible values are "Uniform", "Ramp", "Spectrum">, "SpectrumType:=", <string Possible values are "Rainbow", "Temperature", "Magenta", "Gray">, "UniformColor:=", <array containing the R, G, B components of the color. Components should be in the range 0 to 255.>, "RampColor:=", <array containing the R, G, B com-</pre>

		ponents of the color. Components should be in the range 0 to 255.)
<Scale3DSettings>	Array	Structured array. <pre>Array("NAME:Scale3DSettings", "m_nLevels:=", <int>, "m_autoScale:=", <bool>, "minvalue:=", <double>, "maxvalue:=", <double>, "log:=", <bool>, "IntrinsicMin:=", <double>, "IntrinsicMax:=", <double>)</pre>
<Marker3DSettings>	Array	Structured array. <pre>Array("NAME:Marker3DSettings", "MarkerType:=", <integer 9: Sphere 10: Box 11: Tetrahedron 12: Octahedron default: Sphere>, "MarkerMapSize:=", <bool>, "MarkerMapColor:=", <bool>, "MarkerSize:=", <double>)</pre>
<Arrow3DSettings>	Array	Structured array.

			<pre>Array("NAME:Arrow3DSettings", "ArrowType:=", <integer 0: Line 1: Cylinder 2: Umbrella default: Line>, "ArrowMapSize:=", <bool>, "ArrowMapColor:=", <bool>, "ShowArrowTail:=", <bool>, "ArrowSize:=", <double>)</pre>
Return Value	None.		

Python Syntax	<code>SetPlotFolderSettings(<PlotFolderName>, <PlotFolderAttributes>)</code>
Python Example	<pre>oModule.SetPlotFolderSettings("E Field1", ["NAME:FieldsPlotSettings", "Real time mode:=", True, ["NAME:ColorMapSettings", "ColorMapType:=", "Spectrum", "SpectrumType:=", "Rainbow",</pre>

```
"UniformColor:=", [127, 255, 255],  
"RampColor:=", [255, 127, 127]],  
["NAME:Scale3DSettings",  
"m_nLevels:=", 27,  
"m_autoScale:=", True,  
"minvalue:=", 9.34379863739014,  
"maxvalue:=", 13683.755859375,  
"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]]
```

	)

VB Syntax	SetPlotFolderSettings <PlotFolderName>, <PlotFolderAttributes>
VB Example	<pre>oModule.SetPlotFolderSettings "E Field1", _ Array("NAME:FieldsPlotSettings", _ "Real time mode:=", true, _ Array("NAME:ColorMapSettings", _ "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), _</pre>

```

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))

```

ShowAntennaParameterOverlay

Displays an existing antenna parameters overlay as a table on a far field plot.

UI Access	HFSS or HFSS-IE > Fields > Plot > Radiation Fields		
Parameters	Name	Type	Description
	<Name>	String	The name of the antenna parameters overlay.
Return Value	None.		

Python Syntax	ShowAntennaParameterOverlay(<Name>)
Python Example	oModule.ShowAntennaParameterOverlay(

	"Antenna Parameter Overlay1")

VB Syntax	ShowAntennaParamedterOverlay <Name>
VB Example	<pre>oModule.ShowAntennaParameterOverlay _ "Antenna Parameter Overlay1"</pre>

UpdateAllFieldsPlots

Updates the All Fields Plots.

UI Access	N/A
Parameters	None.
Return Value	None.

Python Syntax	UpdateAllFieldsPlots()
Python Example	<pre>oModule.UpdateAllFieldsPlots()</pre>

VB Syntax	UpdateAllFieldsPlots
VB Example	<pre>oModule.UpdateAllFieldsPlots</pre>

UpdateQuantityFieldsPlots

Updates the Quantity Fields Plots.

UI Access	N/A		
Parameters	Name	Type	Description
	<FolderName>	String	Folder containing the plotted quantities.
Return Value	None.		

Python Syntax	UpdateQuantityFieldsPlots(<FolderName>)
Python Example	<code>oModule.UpdateQuantityFieldsPlots("fldplt")</code>

VB Syntax	UpdateQuantityFieldsPlots <FolderName>
VB Example	<code>oModule.UpdateQuantityFieldsPlots "fldplt"</code>

This page intentionally
left blank.

14 - Fields Calculator Script Commands

Fields Calculator commands should be executed by the Field Overlays module, which is called "FieldsReporter" in HFSS 3D Layout 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](#)

[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.

UI Access	Click Add in the Fields Calculator window.		
Parameters	Name	Type	Description
	<ExprName>	String	Name for the new named expression.
	<FieldType>	String	Type of field.
Return Value	None		

Python Syntax	AddNamedExpression(<ExprName>, <FieldType>)
Python Example	<code>oModule.AddNamedExpression("Mag_JxE", "Fields")</code>

VB Syntax	AddNamedExpression <ExprName>, <FieldType>
VB Example	<code>oModule.AddNamedExpression "Mag_JxE", "Fields"</code>

```
oModule.AddNamedExpression("new2", "Fields",
```

CalcOp

Performs a calculator operation.

UI Access	Operation commands like Mag , + , etc.		
Parameters	Name	Type	Description
	<OpString>	String	The text on the corresponding calculator button.
Return Value	None.		

Python Syntax	CalcOp(<OpString>)
Python Example	<code>oModule.CalcOp ("+")</code>

VB Syntax	CalcOp <OpString>
VB Example	<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.

UI Access	Click Read... in the Fields Calculator dialog.		
Parameters	Name	Type	Description
	<FileName>	String	Name of the file to be read.
	<SoluName>	String	Specified solution to read in.
	<VarArray>	Array	Array of variables to read.
Return Value	None.		

Python Syntax	CalcRead(<FileName>, <SoluName>, <VarArray>)		
Python Example	<pre>oModule.CalcRead("c:\example.reg", "Setup1: LastAdaptive", ["Freq:=", "10GHz", "Phase:=", "0deg"])</pre>		

VB Syntax	CalcRead <FileName>, <SoluName>, <VarArray>		
VB Example	<pre>oModule.CalcRead _ "c:\example.reg", _ "Setup1: LastAdaptive", _</pre>		

```
Array("Freq:=", "10GHz", "Phase:=", "0deg")
```

CalculatorRead

Gets a register file and applies it to the calculator stack.

UI Access	Click Read... in the Fields Calculator dialog.		
Parameters	Name	Type	Description
	<FileName>	String	Path to and including name of input register file.
	<SoluName>	String	Specified solution to read in.
	<FieldType>	String	Type of specified field.
	<VarArray>	Array	Array of variable names, value pairs.
Return Value	None.		

Python Syntax	CalculatorRead(<FileName>, <SoluName>, <FieldType>, <VarArray>)		
Python Example	<pre>oModule.CalculatorRead("c:\example.reg", "Setup1: LastAdaptive", "Fields", ["Freq:=", "10GHz", "Phase:=", "0deg"])</pre>		

VB Syntax	CalculatorRead <FileName>, <SoluName>, <FieldType>, <VarArray>		
VB Example	<pre>oModule.CalculatorRead _ "c:\example.reg", _</pre>		

	<pre>"Setup1: LastAdaptive", "Fields", _ Array("Freq=", "10GHz", "Phase=", "0deg")</pre>

CalcStack

Performs an operation on the stack.

UI Access	Stack operation buttons such as Push and Pop .		
Parameters	Name	Type	Description
	<OpString>	String	The text on the corresponding calculator button.
Return Value	None.		

Python Syntax	CalcStack(<OpString>)
Python Example	<code>oModule.CalcStack("push")</code>

VB Syntax	CalcStack <OpString>
VB Example	<code>oModule.CalcStack "push"</code>

CalculatorWrite

Writes contents of top register to file.

UI Access	Click Write... in the Fields Calculator window.

Parameters	Name	Type	Description
	<OutputFilePath>	String	Path to and including name of output register file.
	<SoluNameArray>	Array	Array of strings containing solution names.
	<VarArray>	Array	Array of variable names, value pairs.
Return Value	None		

Python Syntax	CalculatorWrite(<OutputFilePath>, <SoluNameArray>, <VarArray>)
Python Example	<pre>oModule.CalculatorWrite("C:\test.reg", ["Solution:=", "Setup1 : LastAdaptive"], ["Freq:=", "1GHz", "Phase:=", "0deg"])</pre>

VB Syntax	CalculatorWrite <OutputFilePath>, <SoluNameArray>, <VarArray>
VB Example	<pre>oModule.CalculatorWrite "C:\test.reg", _ Array("Solution:=", "Setup1 : LastAdaptive"), _ Array("Freq:=", "1GHz", "Phase:=", "0deg")</pre>

ChangeGeomSettings

Changes the line discretization setting.

UI Access	In the Fields Calculator dialog box, click on Geom Settings...

Parameters	Name	Type	Description
	<LineDiscr>	Integer	Line discretization value.
Return Value	None.		

Python Syntax	ChangeGeomSettings(<LineDiscr>)
Python Example	oModule.ChangeGeomSettings(500)

VB Syntax	ChangeGeomSettings <LineDiscr>
VB Example	oModule.ChangeGeomSettings 500

ClcEval

Evaluates the expression at the top of the stack using the provided solution name and variable values.

UI Access	Click Eval in the Fields Calculator dialog.		
Parameters	Name	Type	Description
	<SoluName>	String	Name of specified solution.
	<VarArray>	Array	Array of variable name, value pairs.
	<FieldType>	String	Optional. Type of specified field.
Return Value	None		

Python Syntax	<code>ClcEval(<SoluName>, <VarArray>, [Optional <FieldType>])</code>
Python Example	<pre>oModule.ClcEval("Setup1: LastAdaptive", ["Freq:=", "10GHz", "Phase:=", "0deg"])</pre>

VB Syntax	<code>ClcEval <SoluName>, <VarArray>, [Optional <FieldType>]</code>
VB Example	<pre>oModule.ClcEval "Setup1: LastAdaptive", _ Array ("Freq:=", "10GHz", _ "Phase:=", "0deg")</pre>

ClcMaterial

Performs a material operation on the top stack element.

UI Access	Click Matl... in the Fields Calculator dialog.		
Parameters	Name	Type	Description
	<MatString>	String	The material property to apply.
	<OpString>	String	Name of operation. Possible values are "mult", or "div".
Return Value	None.		

Python Syntax	<code>ClcMaterial(<MatString>, <OpString>)</code>
Python Example	<code>oModule.ClcMaterial("Permeability (mu)" "mult")</code>

VB Syntax	<code>ClcMaterial <MatString>, <OpString></code>
VB Example	<code>oModule.ClcMaterial "Permeability (mu)" "mult"</code>

ClcMaterialValue

Shows the value of the material property without performing any operation.

UI Access	Select None in the Material Operation dialog.		
Parameters	Name	Type	Description
	<MaterialName>	String	Name of specified material property.
Return Value	None.		

Python Syntax	<code>ClcMaterialValue(<MaterialName>)</code>
Python Example	<code>oModule.ClcMaterialValue("Mass Density")</code>

VB Syntax	<code>ClcMaterialValue <MaterialName></code>
VB Example	<code>oModule.ClcMaterialValue "Mass Density"</code>

ClearAllNamedExpr

Clears all user-defined named expressions from the list.

UI Access	Click ClearAll in the Fields Calculator dialog.
Parameters	None.
Return Value	None.

Python Syntax	<code>ClearAllNamedExpr()</code>
Python Example	<code>oModule.ClearAllNamedExpr()</code>

VB Syntax	<code>ClearAllNamedExpr</code>
VB Example	<code>oModule.ClearAllNamedExpr</code>

CopyNamedExprToStack

Copies the named expression selected to the calculator stack.

UI Access	Select a named expression and then click Copy to stack .		
Parameters	Name	Type	Description
	<code><ExprName></code>	String	Name of the expression to be copied to the top of the stack.
Return Value	None.		

Python Syntax	<code>CopyNamedExprToStack(<ExprName>)</code>
Python Example	<code>oModule.CopyNamedExprToStack("Mag_JxE")</code>

VB Syntax	CopyNamedExprToStack <ExprName>
VB Example	oModule.CopyNamedExprToStack "Mag_JxE"

DeleteNamedExpr

Deletes the selected named expression from the list.

UI Access	Select a named expression and then click Delete .		
Parameters	Name	Type	Description
	<ExprName>	String	Name of specified named expression.
Return Value	None.		

Python Syntax	DeleteNamedExpr(<ExprName>)
Python Example	oModule.DeleteNamedExpr ("Mag_JxE")

VB Syntax	DeleteNamedExpr <ExprName>
VB Example	oModule.DeleteNamedExpr "Mag_JxE"

DoesNamedExpressionExists

Determines whether specified named expression exists.

UI Access	N/A		
Parameters	Name	Type	Description
	<ExpName>	String	Name of the specified named expression.
Return Value	Boolean: • 1 - named expression exists. • 0 - named expression does not exist.		

Python Syntax	DoesNamedExpressionExists(<ExpName>)		
Python Example	oModule.DoesNamedExpressionExists("Mag_JxE")		

VB Syntax	DoesNamedExpressionExists <ExpName>		
VB Example	oModule.DoesNamedExpressionExists "Mag_JxE"		

EnterComplex

Enters a complex number onto the stack.

UI Access	Click Number , and then click Scalar. Complex option is selected.		
Parameters	Name	Type	Description
	<ComplexNum>	String	String of complex value. Ex. "1 + 2j".
Return Value	None.		

Python Syntax	EnterComplex(<ComplexNum>)
Python Example	oModule.EnterComplex("1 + 2 j")

VB Syntax	EnterComplex <ComplexNum>
VB Example	oModule.EnterComplex "1 + 2 j"

EnterComplexVector

Enters a complex vector onto the stack.

UI Access	Click Number , and then click Vector . Complex option is selected.		
Parameters	Name	Type	Description
	<ComplexVector>	Array	Array of strings containing X, Y and Z complex values.
Return Value	None.		

Python Syntax	EnterComplexVector(<ComplexVector>)
Python Example	<pre>oModule.EnterComplexVector (["1 + 2 j", "1 + 2 j", "1 + 2 j"</pre>

	l)

VB Syntax	EnterComplexVector <ComplexVector>
VB Example	<pre>oModule.EnterComplexVector _ Array("1 + 2 j",_ "1 + 2 j",_ "1 + 2 j")</pre>

EnterCoord

Enters a coordinate system defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Coord .		
Parameters	Name	Type	Description
	<CoordName>	String	Name of a coordinate system defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterCoord(<CoordName>)
Python Example	oModule.EnterCoord("Global")

VB Syntax	EnterCoord <CoordName>

VB Example	<code>oModule.EnterCoord "Global"</code>

EnterEdge

Enters an edge defined in the 3D Modeler editor.

UI Access	N/A		
Parameters	Name	Type	Description
	<EdgeName>	String	Name of an edge defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	<code>EnterEdge(<EdgeName>)</code>
Python Example	<code>oModule.EnterPoint("Edge_1")</code>

VB Syntax	<code>EnterEdge <EdgeName></code>
VB Example	<code>oModule.EnterPoint "Edge_1"</code>

EnterLine

Enters a line defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Line		
Parameters	Name	Type	Description

	<code><LineName></code>	String	Name of a line defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	<code>EnterLine(<LineName>)</code>		
Python Example	<code>oModule.EnterLine("Line1")</code>		

VB Syntax	<code>EnterLine <LineName></code>		
VB Example	<code>oModule.EnterLine "Line1"</code>		

EnterOutputVar

Enters Output Vars, only valid for Eigenmode problems.

UI Access	Click Geometry and then select Output Vars .		
Parameters	Name	Type	Description
	<code><VarName></code>	String	Name of the output vars. Only 'freq' is supported.
	<code><VarType></code>	String	Type of the output vars. 'Complex' type.
Return Value	None.		

Python Syntax	<code>EnterOutputVar(<VarName>, <VarType>)</code>		
Python Example	<code>oModule.EnterOutputVar("Freq", "Complex")</code>		

VB Syntax	EnterOutputVar <VarName>, <VarType>
VB Example	oModule.EnterOutputVar "Freq", "Complex"

EnterPoint

Enters a point defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Point .		
Parameters	Name	Type	Description
	<PointName>	String	Name of a point defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterPoint(<PointName>)
Python Example	oModule.EnterPoint("Point1")

VB Syntax	EnterPoint <PointName>
VB Example	oModule.EnterPoint "Point1"

EnterQty

Enters a field quantity.

UI Access	Click Quantity , and then select from the list.

Parameters	Name	Type	Description
	<FieldQty>	String	The field quantity to be entered onto the stack.
Return Value	None.		

Python Syntax	EnterQty(<FieldQty>)
Python Example	<code>oModule.EnterQty("E")</code>

VB Syntax	EnterQty <FieldQty>
VB Example	<code>oModule.EnterQty "E"</code>

EnterScalar

Enters a scalar onto the stack.

UI Access	Click Number and then click Scalar . Complex option not selected.		
Parameters	Name	Type	Description
	<Scalar>	Double	The real number to enter onto the stack.
Return Value	None.		

Python Syntax	EnterScalar(<Scalar>)
Python Example	<code>oModule.EnterScalar(3.14159265358979)</code>

VB Syntax	EnterScalar <Scalar>
VB Example	<code>oModule.EnterScalar 3.14159265358979</code>

EnterScalarFunc

Enters a scalar function.

UI Access	Click Function and then select Scalar .		
Parameters	Name	Type	Description
	<VarName>	String	Name of a variable to enter as a scalar function onto the stack.
Return Value	None.		

Python Syntax	EnterScalarFunc(<VarName>)
Python Example	<code>oModule.EnterScalarFunc("Phase")</code>

VB Syntax	EnterScalarFunc <VarName>
VB Example	<code>oModule.EnterScalarFunc "Phase"</code>

EnterSurf

Enters a surface defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Surface .

Parameters	Name	Type	Description
	<SurfName>	String	Name of a surface defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterSurf(<SurfName>)
Python Example	<code>oModule.EnterSurf("Rectangle1")</code>

VB Syntax	EnterSurf <SurfName>
VB Example	<code>oModule.EnterSurf "Rectangle1"</code>

EnterVector

Enters a vector onto the stack.

UI Access	Click Number , and then click Vector . Complex option not selected.		
Parameters	Name	Type	Description
	<VectorArray>	Array	Array of strings containing X, Y and Z components of the vector.
Return Value	None.		

Python Syntax	EnterVector(<VectorArray>)
Python Example	<code>oModule.EnterVector([1.0, 1.0, 1.0])</code>

VB Syntax	EnterVector <VectorArray>
VB Example	oModule.EnterVector Array(1.0, 1.0, 1.0)

EnterVectorFunc

Enters a vector function.

UI Access	Click Function and then select Vector .		
Parameters	Name	Type	Description
	<VarNames>	Array	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.
Return Value	None.		

Python Syntax	EnterVectorFunc(<VarNames>)
Python Example	oModuleEnterVectorFunc(["X", "Y", "Z"])

VB Syntax	EnterVectorFunc <VarNames>
VB Example	oModuleEnterVectorFunc Array("X", "Y", "Z")

EnterVol

Enters a volume defined in the 3D Modeler editor.

UI Access	Click Geometry and then select Volume .		
Parameters	Name	Type	Description
	<VolumeName>	String	Name of a volume defined in the 3D Modeler editor.
Return Value	None.		

Python Syntax	EnterVol(<VolumeName>)		
Python Example	oModule.EnterVol("Box1")		

VB Syntax	EnterVol <VolumeName>		
VB Example	oModule.EnterVol "Box1"		

ExportOnGrid [Fields Calculator]

Evaluates the top stack element at a set of points specified by a grid and exports the data to a file.

UI Access	Click Export , and then click On Grid .		
Parameters	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.
Return Value	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”

Python Syntax	ExportOnGrid(<OutputFile>, <Min>, <Max>, <Spacing>, <SolnName>, <SolnParameters>, [<ExportOption>, "IncludePointsInOutput:=", <boolean>], "RefCSName:=", <CSName>, "PtsInSI:=", <boolean>, "FieldRefCS:=", <boolean>], "<CSType>", [<Offsets>, <ByCount>])
Python Example	<pre>oModule.ExportOnGrid("C:\offset_grid_model_unit_ref.fld", ["-1mm", "16mm", "0mm"], ["1mm", "18mm", "1mm"], ["2mm", "2mm", "1mm"], "4500MHz : LastAdaptive", ["Freq:=", "4.5GHz", "Phase:=", "0deg"], ["NAME:ExportOption", "IncludePtInOutput:=", True, "RefCSName:=", "offset", "PtInSI:=", False,</pre>

	<pre> "FieldInRefCS:=" , True], "Cartesian", ["0mm", "0mm", "0mm"], False) </pre>

VB Syntax	ExportOnGrid(<OutputFile>, <Min>, <Max>, <Spacing>, <SolnName>, <SolnParameters>, [<ExportOption>, "IncludePointsInOutput:=", <boolean>], "RefCSName:=", <CSName>, "PtsInSl:=", <boolean>, "FieldRefCS:=", <boolean>], "<CSType>", [<Offsets>, <ByCount>])
VB Example	<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"), _ 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"), </pre>

	<pre>"4500MHz : LastAdaptive", Array("Freq:=", "4.5GHz","Phase:=" , "0deg"), ["NAME:ExportOption", "IncludePtInOutput:=" , True, "RefCSName:=" , "offset", "PtInSI:=" , False, "FieldInRefCS:=" , True], "Cartesian", Array("0mm", "0mm", "0mm"), False)</pre>

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”

UI Access	Click Export , and then click To File .		
Parameters	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>.
Return Value	None.

Python Syntax	ExportToFile(<ReportName>, <FileName>, <solution>, ["Freq:=", <freq>, "Phase:=", "<phase>"], <ExportOptions>)
Python Example	<pre>oModule.ExportToFile("C:\\offset_file_model_unit_ref.fld", "C:\\offset_SI.pts", "4500MHz : LastAdaptive", ["Freq:=" , "4.5GHz", "Phase:=" , "0deg"], ["NAME:ExportOption",</pre>

	<pre>"IncludePtInOutput:=" , True, "RefCSName:=" , "offset", "PtInSI:=" , False, "FieldInRefCS:=" , True])</pre>

VB Syntax	ExportToFile <ReportName>, <FileName>
VB Example	<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>

	])

GetTopEntryValue

Gets the value of the top entry of the calculator stack.

UI Access	N/A		
Parameters	Name	Type	Description
	<SolnName>	String	Name of specified solution.
	<VarVals>	Array	Array of variable name, value pairs.
Return Value	Array of strings containing top entry values.		

Python Syntax	GetTopEntryValue(<SolnName>, <VarVals>)
Python Example	<pre>oModule.GetTopEntryValue("Setup1:LastAdaptive", ["Freq:=", "1GHz", "Phase:=", "0deg", "x_size:=", "2mm"])</pre>

VB Syntax	GetTopEntryValue <SolnName>, <VarVals>
VB Example	<pre>oModule.GetTopEntryValue "Setup1:LastAdaptive", _ Array("Freq:=", "1GHz", "Phase:=", "0deg", _</pre>

	"x_size:=", "2mm")

LoadNamedExpressions

Loads a named expression definition from a saved file.

UI Access	In the Fields Calculator, click Load From... in the Library area.		
Parameters	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.
Return Value	None.		

Python Syntax	LoadNamedExpressions(<FileName>, <FieldType>, <NamedExpr>)
Python Example	<pre>oModule.LoadNamedExpressions ("C:\Ansoft\PersonalLib\smth.clc", "Fields", ["smoothedtemp"])</pre>

VB Syntax	LoadNamedExpressions <FileName>, <FieldType>, <NamedExpr>
VB Example	<pre>oModule.LoadNamedExpressions _ "C:\Ansoft\PersonalLib\smth.clc", _</pre>

	"Fields", Array("smoothedtemp")

SaveNamedExpressions

Saves a named expression definition to a file.

UI Access	In the Fields Calculator, click Save To... in the Library area.		
Parameters	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.
Return Value	None.		

Python Syntax	SaveNamedExpressions(<FileName>, <NamedExprs>, <Overwrite>)		
Python Example	<pre>oModule.SaveNamedExpressions ("C:\Ansoft\PersonalLib\smth.clc", ["smoothedtemp"], True)</pre>		

VB Syntax	SaveNamedExpressions <FileName>, <NamedExprs>, <Overwrite>		
VB Example	<pre>oModule.SaveNamedExpressions _ "C:\Ansoft\PersonalLib\smth.clc", _ Array("smoothedtemp"), true</pre>		

This page intentionally
left blank.

15 - Radiation Module Script Commands

Radiation field commands should be executed by the **RadField** module.

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

```
oModule.CommandName <args>
```

Conventions Used in this Chapter

<SetupName>

Type: <string>

Name of a radiation setup.

<FaceListName>

Type: <string>

Name of a qualifying face list. Used for specifying custom radiation surfaces. In order to be valid for use in a radiation surface, the face list should not contain any faces on PML objects and should contain only model faces.

<CSName>

Type: string

Name of a coordinate system.

The topics for this section include:

[General Commands Recognized by the Radiation Module](#)

[Script Commands for Creating and Modifying Radiation Setups](#)

[Script Commands for Modifying Antenna Array Setups](#)

[Script Commands for Exporting Antenna Parameters and Max Field Parameters](#)

General Commands Recognized by the Radiation Module

[DeleteFarFieldSetup](#)

[DeleteNearFieldSetup](#)

[GetSetupNames](#)

[RenameSetup](#)

[SetPropValue \[Radiation\]](#)

CopyRadFieldSetup

Copies a radiation field setup to clipboard.

UI Access	N/A		
Parameters	Name	Type	Description
	<SetupName>	String	Name of radiation field setup to be copied.
Return Value	None.		

Python Syntax	CopyRadFieldSetup(<SetupName>)
Python Example	<code>oModule.CopyRadFieldSetup("RadField1")</code>

VB Syntax	CopyRadFieldSetup <SetupName>
VB Example	<code>oModule.CopyRadFieldSetup "RadField1"</code>

DeleteFarFieldSetup

Deletes an existing far-field setup.

UI Access	Right-click on a far field setup under Radiation in the project tree, select Delete .		
Parameters	Name	Type	Description
	<SetupName>	Array	Name of setup(s) to be deleted.
Return Value	None.		

Python Syntax	DeleteFarFieldSetup(<SetupName>)
Python Example	<code>oModule.DeleteFarFieldSetup(["Infinite Sphere1"])</code>

VB Syntax	
VB Example	<code>oModule.DeleteFarFieldSetup _ Array("Infinite Sphere1")</code>

DeleteNearFieldSetup

Deletes an existing near-field line radiation setup.

UI Access	Project Manager > Radiation. Right-click a setup and click Delete .		
Parameters	Name	Type	Description
	<NameArray>	Array	List of fields for deletion.
Return Value	None.		

Python Syntax	DeleteNearFieldSetup(<i>NameArray</i>)
Python Example	<code>oModule.DeleteNearFieldSetup(['Line1', 'Sphere1', 'Box2', 'Rectangle1'])</code>

VB Syntax	DeleteNearFieldSetup(<i>NameArray</i>)
VB Example	<code>oModule.DeleteNearFieldSetup Array("Line1", "Sphere1", "Box2", "Rectangle1")</code>

DeleteSetup

Deletes specified radiation field setup(s).

UI Access	Right-click on a setup under Radiation in the project, select Delete .		
Parameters	Name	Type	Description
	<SetupName>	Array	Array of setup names to be deleted.
Return Value	None.		

Python Syntax	DeleteSetup(<SetupName>)
Python Example	<code>oModule.DeleteSetup(["Rad Field1"])</code>

VB Syntax	DeleteSetup <SetupName>

VB Example	<code>oModule.DeleteSetup Array("Rad Field1")</code>

EditRadiatedPowerCalculationMethod

Selects the method used to calculate radiated power.

UI Access	Right-click Radiation in the project tree, select Radiated Power Calculation Method .		
Parameters	Name	Type	Description
	<PowerCalculationMethod>	String	One of: "Auto", "Raidation Surface Integral", or "Far Field Integral".
Return Value	None.		

Python Syntax	<code>EditRadiatedPowerCalculationMethod(<PowerCalculationMethod>)</code>		
Python Example	<code>oModule.EditRadiatedPowerCalculationMethod("Auto")</code>		
	<code>oModule.EditRadiatedPowerCalculationMethod("Radiation Surface Integral")</code>		
	<code>oModule.EditRadiatedPowerCalculationMethod("Far Field Integral")</code>		

VB Syntax	<code>EditRadiatedPowerCalculationMethod <PowerCalculationMethod></code>		
VB Example	<code>oModule.EditRadiatedPowerCalculationMethod "Auto"</code>		
	<code>oModule.EditRadiatedPowerCalculationMethod "Radiation Surface Integral"</code>		
	<code>oModule.EditRadiatedPowerCalculationMethod "Far Field Integral"</code>		

GetChildNames [Radiation]

Use: Gets setup names .

Syntax: GetChildNames()

Return Value: All setup names.

Parameters: Optional, this parameter is ignored.

Python Syntax	GetChildNames ()
Python Example	<pre>GetChildNames() //return all setup names.</pre>

GetChildObject [Radiation]

Gets a Setup Object of the Radiation module.

Python Syntax	GetChildObject()
Python Example	<pre>oOverlay= oRadModule.GetChildObject('Antenna Parameter Overlay1') oSphere = oRadModule.GetChildObject('Infinite Sphere1')</pre>

GetChildTypes [Radiation]

Use: Gets child types of queried Radiation module object.

Syntax: GetChildTypes()

Return Value: Returns empty array.

Python Syntax	GetChildTypes ()

Python Example	<pre>oRad = oDesign.GetChildObject("RadField") oRadModule.GetChildTypes()</pre>

GetPropNames [Radiation]

Use: Radiation module does not have its own property, this function always returns empty array.

Syntax: GetPropNames(bIncludeReadOnly)

Return Value: Returns empty set.

Parameters: bIncludeReadOnly—optional, default to True.

Python Syntax	GetPropNames ()
Python Example	<pre>oRadModule.GetPropNames () oRadModule.GetPropNames (True) oRadModule.GetPropNames (False)</pre>

GetProperty [Radiation]

Get the property value for a setup object.

UI Access	NA		
Parameters	Name	Type	Description
	PropPath Value	text string	A Property path begin with a setup name. See property path discussion here .
Return Value	The property value.		

Python Syntax	GetPropValue(propPath)
Python Example	<pre>oRadModule.GetPropValue('Line1/Num Points') //Get the Line1 setups' "Num Points" property value.</pre>

GetSetupNames

Gets the names of far field and near field radiation setups in a design.

UI Access	N/A		
Parameters	Name	Type	Description
	<RadiationType>	String	Type of radiation.
Return Value	Array of strings containing setup names.		

Python Syntax	GetSetupNames(<RadiationType>)
Python Example	<pre>oModule.GetSetupNames("Infinite Sphere")</pre>

VB Syntax	GetSetupNames <RadiationType>
VB Example	<pre>oModule.GetSetupNames "Infinite Sphere"</pre>

PasteRadFieldSetup

Pastes a copied radiation field setup.

UI Access	N/A
Parameters	None.
Return Value	String name of pasted setup.

Python Syntax	PasteRadFieldSetup()
Python Example	<code>oModule.PasteRadFieldSetup()</code>

VB Syntax	PasteRadFieldSetup
VB Example	<code>oModule.PasteRadFieldSetup</code>

RenameSetup [Radiation]

Renames an existing radiation setup.

UI Access	Right-click a radiation setup in the project tree, and then click Rename on the shortcut menu.		
Parameters	Name	Type	Description
	<OldName>	String	Name of specified setup to be renamed.
	<NewName>	String	New name for the setup.
Return Value	None.		

Python Syntax	RenameSetup(<OldName>, <NewName>)
Python Example	<code>oModule.RenameSetup("Sphere1", "MyNearSphere")</code>

VB Syntax	RenameSetup <OldName>, <NewName>
VB Example	<code>oModule.RenameSetup "Sphere1", "MyNearSphere"</code>

SetPropValue [Radiation]

Sets the property value for the active Radiation property object.

UI Access	Edit Properties on ProjectTree objects		
Parameters	Name	Type	Description
	propPath	text string	A Property path begin with a setup name. See property path discussion here .
	Value	Text String, Number, or Boolean	New value data type is depending on the property type.
Return Value	True if the property is found and the new value is valid. Otherwise return False.		

Python Syntax	SetPropValue(propPath, value)
Python Example	<code>oRadModule.SetPropValue('Line1/Num Points', 100) ; Set the Line1 setups' "Num Points" property to 100.</code>

VB Syntax	SetPropValue(propPath, value)

VB Example	<code>SetPropValue("Color/r",111)</code>

Script Commands for Creating and Modifying Radiation Setups

[EditBoxSetup](#)

[EditFarFieldSphereSetup](#)

[EditNearFieldLineSetup](#)

[EditNearFieldRectangleSetup](#)

[EditNearFieldSphereSetup](#)

[InsertBoxSetup](#)

[InsertFarFieldSphereSetup](#)

[InsertNearFieldLineSetup](#)

[InsertNearFieldRectangleSetup](#)

[InsertNearFieldSphereSetup](#)

AddAntennaOverlay

Saves a table of antenna parameters for use as an overlay on far field plots.

UI Access	Click on Save for Overlay in the Antenna Parameters dialog.		
Parameters	Name	Type	Description
	<AntennaParams>	Array	Structured array. <code>Array("NAME:<AntennaOverlayName>", "Setup:=", <string far field radiation setup name>, "Solution:=", <string for the Solution Setup and Pass>,</code>

			"IntrinsicVariation:=", <string for variation definition>, "UseNominalDesign:=", <boolean>, "DesignVariation:=", <string>, "Calculations:=", <list of each antenna parameter to display>)
Return Value	None.		

Python Syntax	AddAntennaOverlay(<AntennaParams>)
Python Example	<pre>oModule.AddAntennaOverlay(["NAME:Antenna Parameter Overlay1", "Setup:=", "Infinite Sphere1", "Solution:=", "4500MHz : LastAdaptive", "IntrinsicVariation:=", "Freq=4.5GHz ", "UseNominalDesign:=", False, "DesignVariation:=", "", "Calculations:=", "Max U,Peak Directivity,Peak Gain,Radiation Efficiency, Front To Back Ratio,Total Efficiency,System Efficiency"])</pre>

VB Syntax	AddAntennaOverlay <AntennaParams>
VB Example	<pre>oModule.AddAntennaOverlay _ Array("NAME:Antenna Parameter Overlay1", _ "Setup:=", "Infinite Sphere1", _ "Solution:=", "Setup1 : LastAdaptive", _ "IntrinsicVariation:=", "Freq=1.8GHz", _ "UseNominalDesign:=", true, _ "Calculations:=", "Max U,Peak Directivity,Peak Gain,Radiation Efficiency,Front To Back Ratio")</pre>

AddRadFieldSourceGroup

Adds source group definitions for radiation field setup.

UI Access	Right-click on Radiation in the project tree, select Insert Far Field Setup > Source Group...		
Parameters	Name	Type	Description
	<SourceGroupParams>	Array	Structured array <pre>Array("NAME:<SourceGroupName>", Array("NAME:Sources", <SourceName>, <SourceName>...))</pre>
Return Value	None.		

Python Syntax	AddRadFieldSourceGroup(<SourceGroupParams>)
Python Example	<pre>oModule.AddRadFieldSourceGroup(["NAME:Source Group1", ["NAME:Sources", "feedpin1_T1", "feedpin1_T2"]])</pre>

VB Syntax	AddRadFieldSourceGroup <SourceGroupParams>
VB Example	<pre>oModule.AddRadFieldSourceGroup _ Array("NAME:Source Group1", _ Array("NAME:Sources", _ "feedpin1_T1", "feedpin1_T2"))</pre>

EditAntennaOverlay

Edits parameters in an existing antenna overlay.

UI Access	N/A		
Parameters	Name	Type	Description
	<AntennaOverlayName>	String	Name of the antenna overlay to be edited.

	<AntennaParams>	Array	Structured array. <pre>Array("NAME:<AntennaOverlayName>", "Setup:=", <string far field radiation setup name>, "Solution:=", <string for the Solution Setup and Pass>, "IntrinsicVariation:=", <string for variation definition>, "UseNominalDesign:=", <boolean>, "DesignVariation:=", <string>, "Calculations:=" , <list of each antenna parameter to display>)</pre>
Return Value	None.		

Python Syntax	EditAntennaOverlay(<AntennaOverlayName>, <AntennaParams>)
Python Example	<pre>oModule.EditAntennaOverlay("Antenna Parameter Overlay1", ["NAME:Antenna Parameter Overlay1", "Setup:=", "Infinite Spherel", "Solution:=", "4500MHz : LastAdaptive", "IntrinsicVariation:=", "Freq=4.5GHz ", "UseNominalDesign:=", False, "DesignVariation:=", "",</pre>

	<pre>"Calculations:=", "Max U,Peak Directivity,Peak Gain,Radiation Efficiency, Front To Back Ratio,Total Efficiency,System Efficiency"])</pre>

VB Syn- tax	EditAntennaOverlay <AntennaOverlayName>, <AntennaParams>
VB Example	<pre>oModule.AddAntennaOverlay "Antenna Parameter Overlay1", _ Array("NAME:Antenna Parameter Overlay1", _ "Setup:=", "Infinite Sphere1", _ "Solution:=", "Setup1 : LastAdaptive", _ "IntrinsicVariation:=", "Freq=1.8GHz", _ "UseNominalDesign:=", true, _ "Calculations:=", "Max U,Peak Directivity,Peak Gain,Radiation Efficiency,Front To Back Ratio")</pre>

EditBoxSetup

Edits a near-field box radiation setup.

UI Access	Project Manager > Radiation > Box. Double-click and edit information.		
Parameters	Name	Type	Description
	<BoxParams>	Array	Structured array.
			<pre>Array("NAME:<SetupName>",</pre>

			<pre> "UseCustomRadiationSurface:=", <bool>, "CustomRadiationSurface:=", <FaceListName>, "Length:=", <string>, "Width:=", <string>, "LengthSamples:=", <int>, "WidthSamples:=", <int>, "CoordSystem:=", <CSname> "Height:=", <string>, "HeightSamples:=", <int>) </pre> For UseCustomRadiationSurface parameter, provide value if True. If False, radiation boundary/PML boundaries will be used as radiation surfaces.
Return Value	None.		

Python Syntax	EditBoxSetup(<BoxParams>)
Python Example	<pre> oModule.EditBoxSetup(['NAME:MyBox', 'UseCustomRadiationSurface:=', False, 'Length:=', '20mm', 'Width:=', '20mm', 'LengthSamples:=', 21, </pre>

	<pre>'WidthSamples:=' , 21, 'CoordSystem:=' , 'Global', 'Height:=' , '20mm', 'HeightSamples:=' , 21])</pre>

VB Syntax	EditBoxSetup(<BoxParams>)
VB Example	<pre>oModule.EditBoxSetup Array("NAME:MyBox", _ "UseCustomRadiationSurface:=", false, _ "Length:=", "20mm", _ "Width:=", "20mm", _ "LengthSamples:=", 21 _ "WidthSamples:=", 21 _ "CoordSystem:=", "Global" _ "Height:=", "20mm" _ "HeightSamples:=", 21)</pre>

EditFarFieldSphereSetup

Modifies an existing far-field sphere setup.

UI Access	Double-click a radiation setup in the project tree to modify its settings.		
Parameters	Name	Type	Description
	<FarFieldSphereName>	String	Name of the Far Field Sphere Radiation Setup to be edited.
	<FarFieldSphereParams>	Array	Structured array. <pre>Array("NAME:<InfSphereName>", "UseCustomRadiationSurface:=", <boolean>, "CSDefinition:=", <string, "Theta-Phi" "Az Over EI" "EI Over AS">, "Polarization:=", <string, "Linear" "Slant">, "ThetaStart:=", "<value><unit>", "ThetaStop:=", "<value><unit>", "ThetaStep:=", "<value><unit>", "PhiStart:=", "<value><unit>", "PhiStop:=", "<value><unit>", "PhiStep:=", "<value><unit>", "UseLocalCS:=", boolean)</pre>
Return Value	None		

Python Syntax	EditFarFieldSphereSetup(<FarFieldSphereName>, <FarFieldSphereParams>)
Python Example	<pre>oModule.EditFarFieldSphereSetup("Infinite Sphere1", [</pre>

	<pre>"NAME:Infinite Sphere1", "UseCustomRadiationSurface:=", False, "ThetaStart:=" , "0deg", "ThetaStop:=" , "180deg", "ThetaStep:=" , "3deg", "PhiStart:=" , "-180deg", "PhiStop:=" , "180deg", "PhiStep:=" , "3deg", "UseLocalCS:=" , False])</pre>

VB Syntax	EditFarFieldSphereSetup <FarFieldSphereName>, <FarFieldSphereParams>
VB Example	<pre>oModule.EditFarFieldSphereSetup "Infinite Sphere1", _ Array("NAME:Infinite Sphere1",_ "UseCustomRadiationSurface:=", false, "ThetaStart:=", "0deg", _ "ThetaStop:=", "180deg", "ThetaStep:=", "2deg", "PhiStart:=", _ "-180deg", "PhiStop:=", "180deg", "PhiStep:=", "2deg", _ "UseLocalCS:=", false)</pre>

EditInfiniteSphereSetup

Modifies an infinite sphere radiation field setup.

UI Access	Double-click on a setup under Radiation in the project tree.		
Parameters	Name	Type	Description
	<InfSphereName>	String	Name of the infinite sphere setup to be edited.
	<InfSphereParams>	Array	Structured array. <pre>Array("NAME:<InfSphereName>", "UseCustomRadiationSurface:=", <boolean>, "CSDefinition:=", <string, "Theta-Phi" "Az Over EI" "EI Over AS">, "Polarization:=", <string, "Linear" "Slant">, "ThetaStart:=", "<value><unit>", "ThetaStop:=", "<value><unit>", "ThetaStep:=", "<value><unit>", "PhiStart:=", "<value><unit>", "PhiStop:=", "<value><unit>", "PhiStep:=", "<value><unit>", "UseLocalCS:=", boolean)</pre>
Return Value	None.		

Python Syntax	EditInfiniteSphereSetup(<InfSphereName>, <InfSphereParams>)
Python Example	<pre>oModule.EditInfiniteSphereSetup("Infinite Sphere1", ["NAME:Infinite Sphere1", "UseCustomRadiationSurface:=", False, "CSDefinition:=" , "Theta-Phi", "Polarization:=" , "Linear", "ThetaStart:=" , "0deg", "ThetaStop:=" , "180deg", "ThetaStep:=" , "2deg", "PhiStart:=" , "-180deg", "PhiStop:=" , "180deg", "PhiStep:=" , "2deg", "UseLocalCS:=" , False])</pre>

VB Syntax	EditInfiniteSphereSetup <InfSphereName>, <InfSphereParams>
VB Example	<pre>oModule.EditInfiniteSphereSetup "Infinite Sphere1", _ Array("NAME:Infinite Sphere1", _</pre>

	<pre> "UseCustomRadiationSurface:=", false, _ "CSDefinition:=", "Theta-Phi", _ "Polarization:=", "Linear", _ "ThetaStart:=", "0deg", "ThetaStop:=", "180deg", _ "ThetaStep:=", "2deg", "PhiStart:=", "-180deg", _ "PhiStop:=", "180deg", "PhiStep:=", "2deg", _ "UseLocalCS:=", false) </pre>

EditLineSetup

Modifies a near field line setup.

UI Access	Double-click on a setup under Radiation in the project tree.		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup to be edited.
	<LineParams>	Array	Structured array. <pre> Array("NAME:<LineName>", "UseCustomRadiationSurface:=", <boolean>, "Line:=", <string, name of specified line>, "NumPts:=", "<integer, number of points>") </pre>
Return Value	None.		

Python Syntax	EditLineSetup(<SetupName>, <LineParams>)

Python Example	<pre>oModule.EditLineSetup("Line1", ["NAME:Line1", "UseCustomRadiationSurface:=", False, "Line:=" , "Polyline1", "NumPts:=" , "1000"])</pre>

VB Syntax	EditLineSetup <SetupName>, <LineParams>
VB Example	<pre>oModule.EditLineSetup "Line1", _ Array("NAME:Line1", _ "UseCustomRadiationSurface:=", false, _ "Line:=", "Polyline1", "NumPts:=", "1000")</pre>

EditNearFieldBoxSetup

Modifies a near field box radiation field setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<BoxSetupName>	String	Name of the near field box setup to be edited.
	<BoxSetupParams>	Array	Structured array.

			<pre>Array("NAME:<BoxSetupName>", "UseCustomRadiationSurface:=", <boolean>, "Length:=", "<value><unit>", "Width:=", "<value><unit>", "LengthSamples:=", <value>, "WidthSamples:=", <value>, "CoordSystem:=", <string name of coordinate system>, "Height:=", "<value><unit>", "HeightSamples:=", <value>)</pre>
Return Value	None.		

Python Syntax	<code>EditNearFieldBoxSetup(<BoxSetupName>, <BoxSetupParams>)</code>
Python Example	<pre>oModule.EditNearFieldBoxSetup("Box1", ["NAME:Box1", "UseCustomRadiationSurface:=", False, "Length:=" , "20meter", "Width:=" , "20meter", "LengthSamples:=" , 21, "WidthSamples:=" , 21,</pre>

	<pre>"CoordSystem:=" , "Global", "Height:=" , "20meter", "HeightSamples:=" , 21 l)</pre>

VB Syntax	EditNearFieldBoxSetup <BoxSetupName>, <BoxSetupParams>
VB Example	<pre>oModule.EditNearFieldBoxSetup "Box1", _ Array("NAME:Box1", _ "UseCustomRadiationSurface:=", false, _ "Length:=", "20meter", "Width:=", "20meter", _ "LengthSamples:=", 21, "WidthSamples:=", 21, _ "CoordSystem:=", "Global", _ "Height:=", "20meter", "HeightSamples:=", 21)</pre>

EditNearFieldLineSetup

Modifies a near field line setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup to be edited.
	<LineParams>	Array	Structured array.

			<pre>Array("NAME:<LineName>", "UseCustomRadiationSurface:=", <boolean>, "Line:=", <string, name of specified line>, "NumPts:=", "<integer, number of points>")</pre>
Return Value	None.		

Python Syntax	EditNearFieldLineSetup(<SetupName>, <LineParams>)		
Python Example	<pre>oModule.EditNearFieldLineSetup("Line1", ["NAME:Line1", "UseCustomRadiationSurface:=", False, "Line:=" , "Polyline1", "NumPts:=" , "1000"])</pre>		

VB Syntax	EditNearFieldLineSetup <SetupName>, <LineParams>		
VB Example	<pre>oModule.EditNearFieldLineSetup "Line1", _ Array("NAME:Line1", _ "UseCustomRadiationSurface:=", false, _ "Line:=", "Polyline1", "NumPts:=", "1000")</pre>		

EditNearFieldRectangleSetup

Modifies a near field rectangle radiation setup.

UI Access	N/A.		
Parameters	Name	Type	Description
	<RectangleSetupName>	String	Name of rectangle radiation setup to be edited.
	<RectangleSetupParams>	Array	Structured array. Array("NAME:<RectangleSetupName>", "UseCustomRadiationSurface:=", <boolean>, "Length:=", "<value><unit>", "Width:=", "<value><unit>", "LengthSamples:=", <value>, "WidthSamples:=", <value>, "CoordSystem:=", <string name of coordinate sys- tem>)
Return Value	None.		

Python Syntax	EditNearFieldRectangleSetup(<RectangleSetupName>, <RectangleSetupParams>)
Python Example	<pre>oModule.EditNearFieldRectangleSetup("Rectangle1", ["NAME:Rectangle1",</pre>

	<pre> "UseCustomRadiationSurface:=", False, "Length:=" , "20meter", "Width:=" , "20meter", "LengthSamples:=" , 41, "WidthSamples:=" , 41, "CoordSystem:=" , "Global"] </pre>

VB Syntax	EditNearFieldRectangleSetup <RectangleSetupName>, <RectangleSetupParams>
VB Example	<pre> oModule.EditNearFieldRectangleSetup "Rectangle1", _ Array("NAME:Rectangle1", _ "UseCustomRadiationSurface:=", false, _ "Length:=", "20meter", "Width:=", "20meter", _ "LengthSamples:=", 41, "WidthSamples:=", 41, _ "CoordSystem:=", "Global") </pre>

EditNearFieldSphereSetup

Modifies a near field sphere radiation setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<SphereSetupName>	String	Name of sphere radiation field setup to be edited.

	<i><SphereSetupParams></i>	Array	Structured array. Array("NAME:<SphereSetupName>", "UseCustomRadiationSurface:=", <boolean>, "Radius:=", "<value><unit>", "ThetaStart:=", "<value><unit>", "ThetaStop:=", "<value><unit>", "ThetaStep:=", "<value><unit>", "PhiStart:=", "<value><unit>", "PhiStop:=", "<value><unit>", "PhiStep:=", "<value><unit>", "UseLocalCS:=", <boolean>)
Return Value	None.		

Python Syntax	EditNearFieldSphereSetup(<i><SphereSetupName></i> , <i><SphereSetupParams></i>)
Python Example	<pre>oModule.EditNearFieldSphereSetup("Sphere1", ["NAME:Sphere2", "UseCustomRadiationSurface:=", False, "Radius:=" , "20meter",</pre>

	<pre> "ThetaStart:=" , "0deg", "ThetaStop:=" , "180deg", "ThetaStep:=" , "2deg", "PhiStart:=" , "-180deg", "PhiStop:=" , "180deg", "PhiStep:=" , "2deg", "UseLocalCS:=" , False l) </pre>

VB Syntax	EditNearFieldSphereSetup <SphereSetupName>, <SphereSetupParams>
VB Example	<pre> oModule.EditNearFieldSphereSetup "Sphere1", _ Array("NAME:Sphere2", _ "UseCustomRadiationSurface:=", false, _ "Radius:=", "20meter", _ "ThetaStart:=", "0deg", "ThetaStop:=", "180deg", "ThetaStep:=", "2deg", _ "PhiStart:=", "-180deg", "PhiStop:=", "180deg", "PhiStep:=", "2deg", _ "UseLocalCS:=", false) </pre>

EditRadFieldSourceGroup

Modifies an existing source group definitions for radiation field setup.

UI Access	Double-click on a source group setup under Radiation in the project tree.

Parameters	Name	Type	Description
	<SourceGroupName>	String	Name of the source group setup to be edited.
	<SourceGroupParams>	Array	Structured array Array("NAME:<SourceGroupName>", Array("NAME:Sources", <SourceName>, <SourceName>...))
Return Value	None.		

Python Syntax	EditRadFieldSourceGroup(<SourceGroupName>, <SourceGroupParams>)
Python Example	<pre>oModule.EditRadFieldSourceGroup("Source Group1", ["NAME:Source Group1", ["NAME:Sources", "IncPWave1", "IncPWave2"]])</pre>

VB Syntax	EditRadFieldSourceGroup <SourceGroupName>, <SourceGroupParams>
VB Example	<pre>oModule.EditRadFieldSourceGroup "Source Group1", _</pre>

	<pre>Array("NAME:Source Group1", _ Array("NAME:Sources", _ "IncPWave1", "IncPWave2"))</pre>

EditRectangleSetup

Modifies a near field rectangle radiation setup.

UI Access	Double-click on a rectangle setup under Radiation in the project tree.		
Parameters	Name	Type	Description
	<RectangleSetupName>	String	Name of rectangle radiation setup to be edited.
	<RectangleSetupParams>	Array	Structured array. <pre>Array("NAME:<RectangleSetupName>", "UseCustomRadiationSurface:=", <boolean>, "Length:=", "<value><unit>", "Width:=", "<value><unit>", "LengthSamples:=", <value>, "WidthSamples:=", <value>, "CoordSystem:=", <string name of coordinate sys- tem>)</pre>
Return Value	None.		

Python Syntax	EditRectangleSetup(<RectangleSetupName>, <RectangleSetupParams>)
Python Example	oModule.EditRectangleSetup("Rectangle1",

	<pre>["NAME:Rectangle1", "UseCustomRadiationSurface:=", False, "Length:=" , "20meter", "Width:=" , "20meter", "LengthSamples:=" , 41, "WidthSamples:=" , 41, "CoordSystem:=" , "Global"]</pre>

VB Syntax	EditRectangleSetup <RectangleSetupName>, <RectangleSetupParams>
VB Example	<pre>oModule.EditRectangleSetup "Rectangle1", _ Array("NAME:Rectangle1", _ "UseCustomRadiationSurface:=", false, _ "Length:=", "20meter", "Width:=", "20meter", _ "LengthSamples:=", 41, "WidthSamples:=", 41, _ "CoordSystem:=", "Global")</pre>

EditSphereSetup

Modifies a near field sphere radiation setup.

UI Access	Double-click on a sphere setup under Radiation in the project tree.		
Parameters	Name	Type	Description
	<SphereSetupName>	String	Name of sphere radiation field setup to be edited.
	<SphereSetupParams>	Array	Structured array. <pre>Array("NAME:<SphereSetupName>", "UseCustomRadiationSurface:=", <boolean>, "Radius:=", "<value><unit>", "ThetaStart:=", "<value><unit>", "ThetaStop:=", "<value><unit>", "ThetaStep:=", "<value><unit>", "PhiStart:=", "<value><unit>", "PhiStop:=", "<value><unit>", "PhiStep:=", "<value><unit>", "UseLocalCS:=", <boolean>)</pre>
Return Value	None.		

Python Syntax	<code>EditSphereSetup(<SphereSetupName>, <SphereSetupParams>)</code>
Python Example	<pre>oModule.EditSphereSetup("Sphere1", ["NAME:Sphere2", "UseCustomRadiationSurface:=", False,</pre>

	<pre>"Radius:=" , "20meter", "ThetaStart:=" , "0deg", "ThetaStop:=" , "180deg", "ThetaStep:=" , "2deg", "PhiStart:=" , "-180deg", "PhiStop:=" , "180deg", "PhiStep:=" , "2deg", "UseLocalCS:=" , False])</pre>

VB Syntax	EditSphereSetup <SphereSetupName>, <SphereSetupParams>
VB Example	<pre>oModule.EditSphereSetup "Sphere1", _ Array("NAME:Sphere2", _ "UseCustomRadiationSurface:=", false, _ "Radius:=", "20meter", _ "ThetaStart:=", "0deg", "ThetaStop:=", "180deg", "ThetaStep:=", "2deg", _ "PhiStart:=", "-180deg", "PhiStop:=", "180deg", "PhiStep:=", "2deg", _ "UseLocalCS:=", false)</pre>

InsertBoxSetup

Inserts a near-field box radiation setup.

UI Access	HFSS > Radiation > Insert Near Field Setup > Box.		
Parameters	Name	Type	Description
	<BoxParams>	Array	Structured array. <pre>Array("NAME:<SetupName>", "UseCustomRadiationSurface:=", <bool>, "CustomRadiationSurface:=", <FaceListName>, "Length:=", <string>, "Width:=", <string>, "LengthSamples:=", <int>, "WidthSamples:=", <int>, "CoordSystem:=", <CSname> "Height:=", <string>, "HeightSamples:=", <int>)</pre> For UseCustomRadiationSurface parameter, provide value if True. If False, radiation boundary/PML boundaries will be used as radiation surfaces.
Return Value	None.		

Python Syntax	InsertBoxSetup(<BoxParams>)

Python Example	<pre>oModule.InsertBoxSetup(['NAME:MyBox', 'UseCustomRadiationSurface:=' , False, 'Length:=' , '20mm', 'Width:=' , '20mm', 'LengthSamples:=' , 21, 'WidthSamples:=' , 21, 'CoordSystem:=' , 'Global', 'Height:=' , '20mm', 'HeightSamples:=' , 21])</pre>

VB Syntax	InsertBoxSetup <BoxParams>
VB Example	<pre>oModule.InsertBoxSetup Array("NAME:MyBox", _ "UseCustomRadiationSurface:=", false, _ "Length:=", "20mm", _ "Width:=", "20mm", _ "LengthSamples:=", 21 _ "WidthSamples:=", 21 _</pre>

	<pre>"CoordSystem:=", "Global" _ "Height:=", "20mm" _ "HeightSamples:=", 21)</pre>

InsertFarFieldSphereSetup

Inserts a far field infinite sphere setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<SphereParams>	Array	Structured array. <pre>Array("NAME:<InfSphereName>", "UseCustomRadiationSurface:=", <boolean>, "CSDefinition:=", <string, "Theta-Phi" "Az Over EI" "EI Over AS">, "Polarization:=", <string, "Linear" "Slant">, "ThetaStart:=", "<value><unit>", "ThetaStop:=", "<value><unit>", "ThetaStep:=", "<value><unit>", "PhiStart:=", "<value><unit>", "PhiStop:=", "<value><unit>", "PhiStep:=", "<value><unit>", "UseLocalCS:=", boolean)</pre>
Return Value	None.		

Python Syntax	InsertFarFieldSphereSetup(<SphereParams>)
Python Example	<pre>oModule.InsertFarFieldSphereSetup(["NAME:Infinite Sphere1", "UseCustomRadiationSurface:=", False, "CSDefinition:=" , "Theta-Phi", "Polarization:=" , "Linear", "ThetaStart:=" , "0deg", "ThetaStop:=" , "180deg", "ThetaStep:=" , "2deg", "PhiStart:=" , "-180deg", "PhiStop:=" , "180deg", "PhiStep:=" , "2deg", "UseLocalCS:=" , False])</pre>

VB Syntax	InsertFarFieldSphereSetup<SphereParams>
VB Example	<pre>oModule.InsertFarFieldSphereSetup _</pre>

```

Array("NAME:Infinite Sphere1", _
    "UseCustomRadiationSurface:=", false, _
    "CSDefinition:=", "Theta-Phi", _
    "Polarization:=", "Linear", _
    "ThetaStart:=", "0deg", "ThetaStop:=", "180deg", _
    "ThetaStep:=", "2deg", "PhiStart:=", "-180deg", _
    "PhiStop:=", "180deg", "PhiStep:=", "2deg", _
    "UseLocalCS:=", false)

```

InsertInfiniteSphereSetup

Inserts an infinite sphere radiation field setup.

UI Access	Right-click on Radiation in the project tree, select Insert Far Field Setup > Infinite Sphere...		
Parameters	Name	Type	Description
	<InfSphereParams>	Array	Structured array. <pre> Array("NAME:<InfSphereName>", "UseCustomRadiationSurface:=", <boolean>, "CSDefinition:=", <string, "Theta-Phi" "Az Over EI" "EI Over AS">, "Polarization:=", <string, "Linear" "Slant">, "ThetaStart:=", "<value><unit>", "ThetaStop:=", "<value><unit>", "ThetaStep:=", "<value><unit>", </pre>

			<code>"PhiStart:=", "<value><unit>", "PhiStop:=", "<value><unit>", "PhiStep:=", "<value><unit>", "UseLocalCS:=", boolean)</code>
Return Value	None.		

Python Syntax	<code>InsertInfiniteSphereSetup(<InfSphereParams>)</code>
Python Example	<pre>oModule.InsertInfiniteSphereSetup(["NAME:Infinite Sphere1", "UseCustomRadiationSurface:=", False, "CSDefinition:=", , "Theta-Phi", "Polarization:=", , "Linear", "ThetaStart:=", , "0deg", "ThetaStop:=", , "180deg", "ThetaStep:=", , "2deg", "PhiStart:=", , "-180deg", "PhiStop:=", , "180deg", "PhiStep:=", , "2deg",</pre>

	<pre>"UseLocalCS:=", False])</pre>

VB Syntax	InsertInfiniteSphereSetup <InfSphereParams>
VB Example	<pre>oModule.InsertInfiniteSphereSetup _ Array("NAME:Infinite Sphere1", _ "UseCustomRadiationSurface:=", false, _ "CSDefinition:=", "Theta-Phi", _ "Polarization:=", "Linear", _ "ThetaStart:=", "0deg", "ThetaStop:=", "180deg", _ "ThetaStep:=", "2deg", "PhiStart:=", "-180deg", _ "PhiStop:=", "180deg", "PhiStep:=", "2deg", _ "UseLocalCS:=", false)</pre>

InsertLineSetup

Inserts a near field line setup.

UI Access	Right-click on Radiation in the project tree, select Insert Near Field Setup > Line...		
Parameters	Name	Type	Description
	<LineParams>	Array	Structured array. <pre>Array("NAME:<LineName>", "UseCustomRadiationSurface:=", <boolean>,</pre>

	<table><tr><td></td><td></td><td>"Line:=", <string, name of specified line>, "NumPts:=", "<integer, number of points>")</td></tr></table>					"Line:=", <string, name of specified line>, "NumPts:=", "<integer, number of points>")
		"Line:=", <string, name of specified line>, "NumPts:=", "<integer, number of points>")				
Return Value	None.					

Python Syntax	InsertLineSetup(<LineParams>)
Python Example	<pre>oModule.InsertLineSetup(["NAME:Line1", "UseCustomRadiationSurface:=", False, "Line:=" , "Polyline1", "NumPts:=" , "1000"])</pre>

VB Syntax	InsertLineSetup <LineParams>
VB Example	<pre>oModule.InsertLineSetup _ Array("NAME:Line1", _ "UseCustomRadiationSurface:=", false, _ "Line:=", "Polyline1", "NumPts:=", "1000")</pre>

InsertNearFieldBoxSetup

Inserts a near field box radiation field setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<BoxSetupParams>	Array	Structured array. <pre>Array("NAME:<BoxSetupName>", "UseCustomRadiationSurface:=", <boolean>, "Length:=", "<value><unit>", "Width:=", "<value><unit>", "LengthSamples:=", <value>, "WidthSamples:=", <value>, "CoordSystem:=", <string name of coordinate system>, "Height:=", "<value><unit>", "HeightSamples:=", <value>)</pre>
Return Value	None.		

Python Syntax	InsertNearFieldBoxSetup(<BoxSetupParams>)
Python Example	<pre>oModule.InsertNearFieldBoxSetup(["NAME:Box1",</pre>

	<pre>"UseCustomRadiationSurface:=", False, "Length:=" , "20meter", "Width:=" , "20meter", "LengthSamples:=" , 21, "WidthSamples:=" , 21, "CoordSystem:=" , "Global", "Height:=" , "20meter", "HeightSamples:=" , 21])</pre>

VB Syntax	InsertNearFieldBoxSetup <BoxSetupParams>
VB Example	<pre>oModule.InsertNearFieldBoxSetup _ Array("NAME:Box1", _ "UseCustomRadiationSurface:=", false, _ "Length:=", "20meter", "Width:=", "20meter", _ "LengthSamples:=", 21, "WidthSamples:=", 21, _ "CoordSystem:=", "Global", _ "Height:=", "20meter", "HeightSamples:=", 21)</pre>

InsertNearFieldLineSetup

Inserts a near field line setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<LineParams>	Array	Structured array. Array("NAME:<LineName>", "UseCustomRadiationSurface:=", <boolean>, "Line:=", <string, name of specified line>, "NumPts:=", "<integer, number of points>")
Return Value	None.		

Python Syntax	InsertNearFieldLineSetup(<LineParams>)
Python Example	<pre>oModule.InsertNearFieldLineSetup(["NAME:Line1", "UseCustomRadiationSurface:=", False, "Line:=" , "Polyline1", "NumPts:=" , "1000"])</pre>

VB Syntax	<code>InsertNearFieldLineSetup<LineParams></code>
VB Example	<pre>oModule.InsertNearFieldLineSetup _ Array("NAME:Line1", _ "UseCustomRadiationSurface:=", false, _ "Line:=", "Polyline1", "NumPts:=", "1000")</pre>

InsertNearFieldRectangleSetup

Inserts a near field rectangle radiation setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<code><RectangleSetupParams></code>	Array	Structured array. <pre>Array("NAME:<RectangleSetupName>", "UseCustomRadiationSurface:=", <boolean>, "Length:=", "<value><unit>", "Width:=", "<value><unit>", "LengthSamples:=", <value>, "WidthSamples:=", <value>, "CoordSystem:=", <string name of coordinate sys- tem>)</pre>
Return Value	None.		

Python Syntax	InsertNearFieldRectangleSetup(<RectangleSetupParams>)
Python Example	<pre>oModule.InsertNearFieldRectangleSetup(["NAME:Rectangle1", "UseCustomRadiationSurface:=", False, "Length:=" , "20meter", "Width:=" , "20meter", "LengthSamples:=" , 41, "WidthSamples:=" , 41, "CoordSystem:=" , "Global"])</pre>

VB Syntax	InsertNearFieldRectangleSetup <RectangleSetupParams>
VB Example	<pre>oModule.InsertNearFieldRectangleSetup _ Array("NAME:Rectangle1", _ "UseCustomRadiationSurface:=", false, _ "Length:=", "20meter", "Width:=", "20meter", _ "LengthSamples:=", 41, "WidthSamples:=", 41, _ "CoordSystem:=", "Global")</pre>

InsertNearFieldSphereSetup

Inserts a near field sphere radiation setup.

UI Access	N/A		
Parameters	Name	Type	Description
	<SphereSetupParams>	Array	Structured array. Array ("NAME:<SphereSetupName>", "UseCustomRadiationSurface:=", <boolean>, "Radius:=", "<value><unit>", "ThetaStart:=", "<value><unit>", "ThetaStop:=", "<value><unit>", "ThetaStep:=", "<value><unit>", "PhiStart:=", "<value><unit>", "PhiStop:=", "<value><unit>", "PhiStep:=", "<value><unit>", "UseLocalCS:=", <boolean>)
Return Value	None.		

Python Syntax	InsertNearFieldSphereSetup(<SphereSetupParams>)
Python Example	oModule.InsertNearFieldSphereSetup(

```
[
    "NAME:Sphere2",
    "UseCustomRadiationSurface:=", False,
    "Radius:="                , "20meter",
    "ThetaStart:="            , "0deg",
    "ThetaStop:="             , "180deg",
    "ThetaStep:="             , "2deg",
    "PhiStart:="              , "-180deg",
    "PhiStop:="               , "180deg",
    "PhiStep:="               , "2deg",
    "UseLocalCS:="           , False
]
)
```

VB Syntax	InsertNearFieldSphereSetup < <i>SphereSetupParams</i> >
VB Example	<pre>oModule.InsertNearFieldSphereSetup _ Array("NAME:Sphere2", _ "UseCustomRadiationSurface:=", false, _ "Radius:=", "20meter", _ "ThetaStart:=", "0deg", "ThetaStop:=", "180deg", "ThetaStep:=", "2deg", _ "PhiStart:=", "-180deg", "PhiStop:=", "180deg", "PhiStep:=", "2deg", _ "UseLocalCS:=", false)</pre>

InsertRectangleSetup

Inserts a near field rectangle radiation setup.

UI Access	Right-click on Radiation in the project tree ,select Insert Near Field Setup > Rectangle...		
Parameters	Name	Type	Description
	<RectangleSetupParams>	Array	Structured array. Array ("NAME:<RectangleSetupName>", "UseCustomRadiationSurface:=", <boolean>, "Length:=", "<value><unit>", "Width:=", "<value><unit>", "LengthSamples:=", <value>, "WidthSamples:=", <value>, "CoordSystem:=", <string name of coordinate sys- tem>)
Return Value	None.		

Python Syntax	InsertRectangleSetup(<RectangleSetupParams>)
Python Example	<pre>oModule.InsertRectangleSetup(["NAME:Rectangle1",</pre>

	<pre> "UseCustomRadiationSurface:=", False, "Length:=" , "20meter", "Width:=" , "20meter", "LengthSamples:=" , 41, "WidthSamples:=" , 41, "CoordSystem:=" , "Global"] </pre>

VB Syntax	InsertRectangleSetup < <i>RectangleSetupParams</i> >
VB Example	<pre> oModule.InsertRectangleSetup _ Array("NAME:Rectangle1", _ "UseCustomRadiationSurface:=", false, _ "Length:=", "20meter", "Width:=", "20meter", _ "LengthSamples:=", 41, "WidthSamples:=", 41, _ "CoordSystem:=", "Global") </pre>

InsertSphereSetup

Inserts a near field sphere radiation setup.

UI Access	Right-click on Radiation in the project tree ,select Insert Near Field Setup > Sphere...		
Parameters	Name	Type	Description
	< <i>SphereSetupParams</i> >	Array	Structured array.

			<pre> Array("NAME:<SphereSetupName>", "UseCustomRadiationSurface:=", <boolean>, "Radius:=", "<value><unit>", "ThetaStart:=", "<value><unit>", "ThetaStop:=", "<value><unit>", "ThetaStep:=", "<value><unit>", "PhiStart:=", "<value><unit>", "PhiStop:=", "<value><unit>", "PhiStep:=", "<value><unit>", "UseLocalCS:=", <boolean>) </pre>
Return Value	None.		

Python Syntax	InsertSphereSetup(<SphereSetupParams>)
Python Example	<pre> oModule.InsertSphereSetup(["NAME:Sphere2", "UseCustomRadiationSurface:=", False, "Radius:=" , "20meter", "ThetaStart:=" , "0deg", </pre>

	<pre> "ThetaStop:=" , "180deg", "ThetaStep:=" , "2deg", "PhiStart:=" , "-180deg", "PhiStop:=" , "180deg", "PhiStep:=" , "2deg", "UseLocalCS:=" , False] </pre>

VB Syntax	InsertSphereSetup < <i>SphereSetupParams</i> >
VB Example	<pre> oModule.InsertSphereSetup _ Array("NAME:Sphere2", _ "UseCustomRadiationSurface:=", false, _ "Radius:=", "20meter", _ "ThetaStart:=", "0deg", "ThetaStop:=", "180deg", "ThetaStep:=", "2deg", _ "PhiStart:=", "-180deg", "PhiStop:=", "180deg", "PhiStep:=", "2deg", _ "UseLocalCS:=", false) </pre>

Script Commands for Modifying Antenna Array Setups

[EditAntennaArraySetup](#)

EditAntennaArraySetup

Modifies the antenna array setup. There are 3 choices in the setup. The default is set to **No Array Setup**. There are two (other) kinds of arrays that the user can set: **Regular Array Setup** and **Custom Array Setup**.

UI Access	Right-click on Radiation in the project tree, select Antenna Array Setup...		
Parameters	Name	Type	Description
	<AntennaArrayParams>	Array	Structured array. <pre> Array("NAME:ArraySetupInfo", "UseOption:=", <string "No Array" "Parametric Array" "Custom Array">, <ParametricArrayParams> <CustomArrayParams>) </pre>
	<ParametricArrayParams>	Array	Structured array. <pre> Array("NAME:ParametricArray", "DesignFrequency:=", "<value><unit>", "LayoutType:=", <integer>, "CenterCellX:=", "<value><unit>", "CenterCellY:=", "<value><unit>", "CenterCellZ:=", "<value><unit>", "SpecifyDesignInWavelength:=", <boolean>, "WidthSpacing:=", "<value><unit>", "WidthSpacingInWavelength:=", "<value>", "Width:=", "<value><unit>", </pre>

			<pre> "WidthInWavelength:=", "<value>", "LengthSpacing:=", "<value><unit>", "LengthSpacingInWavelength:=", "<value>", "Length:=", "<value><unit>", "LengthInWavelength:=", "<value>", "SymmetryType:=", "<integer>", "StaggerAngle:=", "<value><unit>", "StaggerType:=", "<integer>", "UDirnX:=", "<integer>", "UDirnY:=", "<integer>", "UDirnZ:=", "<integer>", "VDirnX:=", "<integer>", "VDirnY:=", "<integer>", "VDirnZ:=", "<integer>", "WeightType:=", "<integer>", "EdgeTaperX_db:=", "<integer>", "CosineExp:=", "<integer>", "DifferentialType:=", "<integer>", "Behavior:=", "<string>", "ScanAnglePhi:=", "<value><unit>", "ScanAngleTheta:=", "<value><unit>", </pre>

			<code>"UDirnPhaseShift:=", "<value><unit>", "VDirnPhaseShift:=", "<value><unit>")</code>
	<code><CustomArrayParams></code>	Array	Structured array. <code>Array("NAME:CustomArray", "NumCells:=", <int>, <CellsParamsArray>)</code>
	<code><CellsParamsArray></code>	Array	Structured array. <code>Array("NAME:Cell", <CellParams>, <CellParams>, ...)</code>
	<code><CellParams></code>	Array	Structured array. <code>Array("Name:<CellName>", "XCoord:=", <double>, "YCoord:=", <double>, "ZCoord:=", <double>, "Amplitude:=", <double>, "Phase:=", <double>)</code>
Return Value	None.		

Python Syntax	<code>EditAntennaArraySetup(<AntennaArrayParams>)</code>
Python Example	<code>oModule.EditAntennaArraySetup(</code>

```
[
"NAME:ArraySetupInfo",
"UseOption:="          , "ParametricArray",
["NAME:ParametricArray",
"DesignFrequency:="    , "1GHz",
"LayoutType:="         , 1,
"CenterCellX:="        , "0mm",
"CenterCellY:="        , "0mm",
"CenterCellZ:="        , "0mm",
"SpecifyDesignInWavelength:=", False,
"WidthSpacing:="       , "60mm",
"WidthSpacingInWavelength:=", "0.200138457118891",
"Width:="              , "480mm",
"WidthInWavelength:="  , "1.60110765695113",
"LengthSpacing:="      , "60mm",
"LengthSpacingInWavelength:=", "0.200138457118891",
"Length:="             , "360mm",
"LengthInWavelength:=" , "1.20083074271335",
"SymmetryType:="       , 0,
"StaggerAngle:="       , "0deg",
"StaggerType:="        , 0,
```

	<pre>"UDirnX:=" , "0", "UDirnY:=" , "0", "UDirnZ:=" , "-1", "VDirnX:=" , "1", "VDirnY:=" , "0", "VDirnZ:=" , "0", "WeightType:=" , 3, "EdgeTaperX_db:=" , "-200", "CosineExp:=" , "1", "DifferentialType:=" , 0, "Behavior:=" , "UseSlaveSettings", "ScanAnglePhi:=" , "135deg", "ScanAngleTheta:=" , "45deg", "UDirnPhaseShift:=" , "0deg", "VDirnPhaseShift:=" , "0deg"] l)</pre>

VB Syntax	EditAntennaArraySetup < <i>AntennaArrayParams</i> >
VB Example	oModule.EditAntennaArraySetup _

```

Array("NAME:ArraySetupInfo", "UseOption:=",
"NoArray")

oModule.EditAntennaArraySetup _
Array("NAME:ArraySetupInfo",
"UseOption:=", "CustomArray", _
Array("NAME:CustomArray", _
"NumCells:=", 3, _
Array("NAME:Cell", _
Array("NAME:Cell_1", _
"XCoord:=", 0, "YCoord:=", 0, "ZCoord:=", 0, _
"Amplitude:=", 1, "Phase:=", 0), _
Array("NAME:Cell_2", _
"XCoord:=", 0.06729, "YCoord:=", "ZCoord:=", 0, _
"Amplitude:=", 1, "Phase:=", 0), _
Array("NAME:Cell_3", _
"XCoord:=", 0.13458, "YCoord:=", 0, "ZCoord:=", 0, _
"Amplitude:=", 1, "Phase:=", 0))))

```

Script Commands for Exporting Antenna Parameters and Max Field Parameters

[ExportRadiationParametersToFile](#)

[ExportElementPatternToFile](#)

ExportElementPatternToFile

Exports far field data for specified elements to a file.

UI Access	Export Element Pattern button in Antenna Parameters dialog box.		
Parameters	Name	Type	Description
	<ExportElementPatternFileParams>	Array	Structured array. Array("ExportFileName:=", <string file name include path>, "SetupName:=" , <string radiation setup to use>, "IntrinsicVariationKey:=", <string frequency at which to extract the parameters>, "DesignVariationKey:=", <string design variations at which to extract the parameters.>, "SolutionName:=", <string solution to use>, "SourceGroupName:=", <string source group to use>)
Return Value	None.		

Python Syntax	ExportElementPatternToFile(<ExportElementPatternFileParams>)
Python Example	<pre>oModule.ExportElementPatternToFile(["ExportFileName:=", "C:\Documents\exportelement2.ffd", "SetupName:=", "3D", "IntrinsicVariationKey:=", "Freq=\'28GHz\'", "DesignVariationKey:=", "antipad_rad=\'0.14999999999999999mm\'", "SolutionName:=", "Setup1 : LastAdaptive"]) </pre>

VB Syntax	ExportElementPatternToFile <ExportElementPatternFileParams>
VB Example	<pre>oModule.ExportElementPatternToFile Array("ExportFileName:=", "C:\Documents\exportelement1.ffd", _ "SetupName:=", "3D", _ "IntrinsicVariationKey:=", "Freq=" & Chr(39) & "28GHz" & Chr(39) & "", _ "DesignVariationKey:=", "antipad_rad=" & Chr(39) & "0.14999999999999999mm" & Chr(39) & "", _ "SolutionName:=", "Setup1 : LastAdaptive") </pre>

ExportFieldsToFile

Exports fields to a file.

UI Access	From the Antenna Parameters dialog box, Export Fields to .ffd file.		
Parameters	Name	Type	Description
	<ExportParams>	Array	Structured array. Array("ExportFileName:=", <string file name include path>, "SetupName:=", <string name of the Radiation Far Field or Near Field setup>, "IntrinsicVariationKey:=", <string from the Antenna Parameters dialog.>, "DesignVariationKey:=", <string>, "SolutionName:=" , <string name of solution to use>, "Quantity:=" , <string antenna Parameter name. If empty, all parameters are exported.>)
Return Value	None.		

Python Syntax	ExportFieldsToFile (<ExportParams>)
Python Example	<pre>oModule.ExportFieldsToFile(["ExportFileName:=", "C:/exportfields.ffd", "SetupName:=", "Infinite Sphere1", "IntrinsicVariationKey:=", "Freq='4.5GHz'",</pre>

	<pre> "DesignVariationKey:=", "", "SolutionName:=", "4500MHz : LastAdaptive", "Quantity:=", "" l) </pre>

VB Syntax	ExportFieldsToFile <ExportParams>
VB Example	<pre> oModule.ExportFieldsToFile _ Array("ExportFileName:=", "C:/exportfields.ffd", _ "SetupName:=", "Infinite Spherel", _ "IntrinsicVariationKey:=", "Freq=" & Chr(39) & "6GHz" & Chr(39) & "", _ "DesignVariationKey:=", "", "SolutionName:=", "Setup1 : LastAdaptive", _ "SourceGroupName:=", "Source Group1", _ "Quantity:=", "") </pre>

ExportParametersToFile

Exports antenna parameters to a file.

UI Access	Click Export in the Antenna Parameters dialog.		
Parameters	Name	Type	Description
	<DefParams>	Array	Structured array. Array(

			<code>"ExportFileName:=", <string file name include path>, "SetupName:=", <string name of setup to export>, "IntrinsicVariationKey:=", <string>, "DesignVariationKey:=", <string>, "SolutionName:=", <string name of solution to export>, "Quantity:=", <string name of quantity, empty to export all>)</code>
Return Value	None.		

Python Syntax	<code>ExportParametersToFile(<DefParams>)</code>
Python Example	<pre>oModule.ExportParametersToFile(["ExportFileName:=", "C:/exportparams.txt", "SetupName:=", "Infinite Sphere1", "IntrinsicVariationKey:=", "Freq=\'4.5GHz\'", "DesignVariationKey:=", "", "SolutionName:=", "4500MHz : LastAdaptive", "Quantity:=", ""])</pre>

VB Syntax	ExportParametersToFile(<DefParams>
VB Example	<pre>oModule.ExportParametersToFile _ Array("ExportFileName:=", "C:/exportparams.txt", _ "SetupName:=", "Infinite Sphere1", _ "IntrinsicVariationKey:=", "Freq=\'4.5GHz\'", _ "DesignVariationKey:=", "", _ "SolutionName:=", "4500MHz : LastAdaptive", _ "Quantity:=", "")</pre>

ExportRadiationFieldsToFile

Exports radiation fields to a file.

UI Access	N/A		
Parameters	Name	Type	Description
	<ExportParams>	Array	Structured array. <pre>Array("ExportFileName:=", <string file name include path>, "SetupName:=", <string name of the Radiation Far Field or Near Field setup>, "IntrinsicVariationKey:=", <string from the Antenna Parameters dialog.>, "DesignVariationKey:=", <string>, "SolutionName:=", <string name of solution to use>, "Quantity:=", <string antenna Parameter name. If</pre>

	<table><tr><td></td><td></td><td>empty, all parameters are exported.>)</td></tr></table>					empty, all parameters are exported.>)
		empty, all parameters are exported.>)				
Return Value	None.					

Python Syntax	ExportRadiationFieldsToFile (<ExportParams>)
Python Example	<pre>oModule.ExportRadiationFieldsToFile(["ExportFileName:=", "C:/exportfields.ffd", "SetupName:=", "Infinite Sphere1", "IntrinsicVariationKey:=", "Freq=\'4.5GHz\'", "DesignVariationKey:=", "", "SolutionName:=", "4500MHz : LastAdaptive", "Quantity:=", ""])</pre>

VB Syntax	ExportRadiationFieldsToFile <ExportParams>
VB Example	<pre>oModule.ExportRadiationFieldsToFile _ Array("ExportFileName:=", "C:/exportfields.ffd", _ "SetupName:=", "Infinite Sphere1", _</pre>

	<pre> "IntrinsicVariationKey:=", "Freq=" & Chr(39) & "6GHz" & Chr(39) & "", _ "DesignVariationKey:=", "", "SolutionName:=", "Setup1 : LastAdaptive", _ "SourceGroupName:=", "Source Group1", _ "Quantity:=", "") </pre>

ExportRadiationParametersToFile

Exports radiation parameters to a file. This command can be used to export the max quantities of a near-field setup and, in the case of far fields, the antenna parameters to the specified file.

UI Access	HFSS > Radiation > Compute Max/Antenna Params		
Parameters	Name	Type	Description
	<ExportToFileParams>	Array	Structured array. <pre> Array("ExportFileName:=", <string file name include path> "SetupName:=", <string> "IntrinsicVariationKey:=", <string>, "DesignVariationKey:=", <string>, "SolutionName:=", <string>) </pre>
Return Value	None.		

Python Syntax	ExportRadiationParametersToFile(<ExportToFileParams>)
Python Example	oModule.ExportRadiationParametersToFile(


```
[  
  "ExportFileName:=", "C:\projects\exportantparams.txt",  
  "SetupName:=", "Infinite Sphere1",  
  "IntrinsicVariationKey:=", "Freq='10GHz'",  
  "DesignVariationKey:=", "",  
  "SolutionName:=", "LastAdaptive"  
])
```

VB Syntax	ExportRadiationParametersToFile <ExportToFileParams>
VB Example	<pre>oModule.ExportRadiationParametersToFile _ Array("ExportFileName:=", _ "C:\projects\exportantparams.txt", _ "SetupName:=", "Infinite Sphere1", _ "IntrinsicVariationKey:=", "Freq='10GHz'", _ "DesignVariationKey:=", "", "SolutionName:=", "LastAdaptive")</pre>

16 - Radiation Setup Manager Module Script Commands

Radiation Setup Manager commands should be executed by the **RadiationSetupMgr** module.

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

```
oModule.CommandName <args>
```

The commands in this module create and change radiation boundaries which can be modified by the [Radiation module](#) script commands. Radiation boundaries are divided into a far-field boundary in the shape of an infinite sphere and near-field boundaries in the shapes of a sphere, line, or plane. Each boundary uses a different set of parameters. The following commands create and edit the radiation boundaries.

- [Add](#) or [edit](#) a far field infinite sphere
- [Add](#) or [edit](#) a near-field line
- [Add](#) or [edit](#) a near-field plane
- [Add](#) or [edit](#) a near-field sphere

Add (Infinite Sphere)

Adds a far-field radiation boundary in the shape of an infinite sphere.

Note: Only *Name* and *Type* are required parameters. If you set a custom Theta or Phi parameter, then you must set all Theta and Phi parameters. If you have a local coordinate system, then you must set the *CS* and *UseLocalCS* parameters.

UI Access	In the Project Manager under the design name, right click Radiation and in the right-click menu, click Insert Far Field Setup>Infinite Sphere .		
Parameters	Name	Type	Description
	<Name>	String	A user-defined name for the infinite sphere.
	<Type>	String	Type of radiation setup. It should be IntiniteSphereFarField .
	<VersionID>	Integer	Optional. Set to 0 .

	<ID>	Integer	Optional. Identifier that is created differently for every setup.
	<StartTheta >	String	Optional. Starting point for theta in the spherical coordinate system for the sphere being defined. The default value is " 0deg ". Use the form "<num> deg ".
	<StartPhi>	String	Optional. Starting point for phi in the spherical coordinate system for the sphere being defined. The default value is " -180deg ". Use the form "<num> deg ".
	<StopTheta>	String	Optional. Stopping point for theta in the spherical coordinate system for the sphere being defined. The default value is " 180deg ". Use the form "<num> deg ".
	<StopPhi>	String	Optional. Stopping point for phi in the spherical coordinate system for the sphere being defined. The default value is " 180deg ". Use the form "<num> deg ".
	<StepTheta>	String	Optional. Step size between the start and the stop theta values at which the far field is calculated. The default value is " 2deg ". Use the form "<num> deg ".
	<StepPhi>	String	Optional. Step size between the start and the stop phi values at which the far field is calculated. The default value is " 2deg ". Use the form "<num> deg ".
	<CS>	Integer	Optional. The coordinate system to use. When coordinate system is local, set this number to the index of the local coordinate system. Set this value to -1 for a global coordinate system. The default value is -1 .
	<UseLocalCS>	Boolean	Optional. True to select a local coordinate system, and False to use the global coordinate system. The default value is False .
Return Value	None.		

Python Syntax	oModule.Add(

	<pre> ["NAME:<Name>", "Type:=", "InfiniteSphereFarField", "VersionID:=", 0, "ID:=", <ID>, "StartTheta:=", "<StartTheta>deg", "StartPhi:=", "<StartPhi>deg", "StopTheta:=", "<StopTheta>deg", "StopPhi:=", "<StopPhi>deg", "StepTheta:=", "<StepTheta>deg", "StepPhi:=", "<StepPhi>deg", "CS:=", <CS>, "UseLocalCS:=", <UseLocalCS>] </pre>
Python Example	<pre> oDesign = oDesign.GetModule("RadiationSetupMgr") oModule.Add(["NAME:Infinite Sphere 1", "Type:=", "InfiniteSphereFarField", "VersionID:=", 0, "ID:=", 5, </pre>

	<pre> "StartTheta:=", "0deg", "StartPhi:=", "-180deg", "StopTheta:=", "180deg", "StopPhi:=", "180deg", "StepTheta:=", "2deg", "StepPhi:=", "2deg", "CS:=", 1, "UseLocalCS:=", False] </pre>

VB Syntax	<pre> oModule.Add Array("NAME:<Name>", "Type:=", "InfiniteSphereFarField", "VersionID:=", 0, "ID:=", <ID>, "StartTheta:=", "<StartTheta>deg", "StartPhi:=", "<StartPhi>deg", "StopTheta:=", "<StopPhi>deg", "StopPhi:=", "<StopPhi>deg", "StepTheta:=", "<StepTheta>deg", "StepPhi:=", "<StepPhi>deg", "CS:=", <CS>, "UseLocalCS:=", <UseLocalCS>) </pre>
VB Example	<pre> Set oModule = oDesign.GetModule("RadiationSetupMgr") oModule.Add Array("NAME:Infinite Sphere 1", "Type:=", "InfiniteSphereFarField", "VersionID:=", _ 0, "ID:=", 0, "StartTheta:=", "0deg", "StartPhi:=", "-180deg", "StopTheta:=", _ "180deg", "StopPhi:=", "180deg", "StepTheta:=", "2deg", "StepPhi:=", "2deg", "CS:=", _ -1, "UseLocalCS:=", false) </pre>

Edit (RadiationSetupMgr for Infinite Sphere)

Changes a far-field radiation boundary in the shape of an infinite sphere.

Note: Only *Name* and *Type* are required parameters. If you set a custom Theta or Phi parameter, then you must set all Theta and Phi parameters. If you have a local coordinate system, then you must set the *CS* and *UseLocalCS* parameters.

UI Access	In the Project Manager under the design name > Radiation , double click on the name of the infinite sphere.		
Parameters	Name	Type	Description
	<Name>	String	A user-defined name for the infinite sphere.
	<Type>	String	Type of radiation setup. It should be IntiniteSphereFarField .
	<VersionID>	Integer	Optional. Set to 0 .
	<ID>	Integer	Optional. Identifier that is created differently for every setup.
	<StartTheta >	String	Optional. Starting point for theta in the spherical coordinate system for the sphere being defined. Use the form "<num> deg ".
	<StartPhi>	String	Optional. Starting point for phi in the spherical coordinate system for the sphere being defined. Use the form "<num> deg ".
	<StopTheta>	String	Optional. Stopping point for theta in the spherical coordinate system for the sphere being defined. Use the form "<num> deg ".
	<StopPhi>	String	Optional. Stopping point for phi in the spherical coordinate system for the sphere being defined. Use the form "<num> deg ".
	<StepTheta>	String	Optional. Step size between the start and the stop theta values at which the far field is calculated. Use the form "<num> deg ".
	<StepPhi>	String	Optional. Step size between the start and the stop phi values at which the far field is calculated. Use the form "<num> deg ".
	<CS>	Integer	Optional. The coordinate system to use. When coordinate system is local, set this number to the index of the local coordinate system. Set this value to -1 for a global coordinate system.
	<UseLocalCS>	Boolean	Optional. True to select a local coordinate system, and False to use the global coordinate system.

Return Value	None.

Python Syntax	<pre>oModule.Edit(["NAME:<Name>", "Type:=", "InfiniteSphereFarField", "VersionID:=", 0, "ID:=", <ID>, "StartTheta:=", "<StartTheta>deg", "StartPhi:=", "<StartPhi>deg", "StopTheta:=", "<StopTheta>deg", "StopPhi:=", "<StopPhi>deg", "StepTheta:=", "<StepTheta>deg", "StepPhi:=", "<StepPhi>deg", "CS:=", <CS>, "UseLocalCS:=", <UseLocalCS>])</pre>
Python Example	<pre>oDesign = oDesign.GetModule("RadiationSetupMgr") oModule.Edit(["NAME:<Name>", "Type:=", "InfiniteSphereFarField", "VersionID:=", 0, "ID:=", <ID>, "StartTheta:=", "<StartTheta>deg", "StartPhi:=", "<StartPhi>deg", "StopTheta:=", "<StopTheta>deg", "StopPhi:=", "<StopPhi>deg", "StepTheta:=", "<StepTheta>deg", "StepPhi:=", "<StepPhi>deg", "CS:=", <CS>, "UseLocalCS:=", <UseLocalCS>])</pre>

	<pre>["NAME:Infinite Sphere 1", "Type:=", "InfiniteSphereFarField", "VersionID:=", 0, "ID:=", 5, "StartTheta:=", "0deg", "StartPhi:=", "-180deg", "StopTheta:=", "180deg", "StopPhi:=", "180deg", "StepTheta:=", "2deg", "StepPhi:=", "2deg", "CS:=", 1, "UseLocalCS:=", False]</pre>

VB Syntax	<pre>oModule.Edit Array("NAME:<Name>", "Type:=", "InfiniteSphereFarField", "VersionID:=", 0, "ID:=", <ID>, "StartTheta:=", "<StartTheta>deg", "StartPhi:=", "<StartPhi>deg", "StopTheta:=", "<StopPhi>deg", "StopPhi:=", "<StopPhi>deg", "StepTheta:=", "<StepTheta>deg", "StepPhi:=", "<StepPhi>deg", "CS:=", <CS>, "UseLocalCS:=", <UseLocalCS>)</pre>
VB Example	<pre>Set oModule = oDesign.GetModule("RadiationSetupMgr") oModule.Edit Array("NAME:Infinite Sphere 1", "Type:=", "InfiniteSphereFarField", "VersionID:=", _ 0, "ID:=", 0, "StartTheta:=", "0deg", "StartPhi:=", "-180deg", "StopTheta:=", _</pre>


```
"180deg", "StopPhi:=", "180deg", "StepTheta:=", "2deg", "StepPhi:=", "2deg", "CS:=", _
-1, "UseLocalCS:=", false)
```

Add (Near Field Line)

Adds a near-field radiation boundary in the shape of a line. To calculate the near fields, you must have a 3D line defined across the design area before creating this type of radiation boundary.

UI Access	In the Project Manager under the design name, right click Radiation and in the right-click menu, click Insert Near Field Setup>Line .		
Parameters	Name	Type	Description
	<Name>	String	A user-defined name for the near field line.
	<Type>	String	Type of radiation setup. It should be NearFieldLine .
	<VersionID>	Integer	Optional. Set to 0 .
	<ID>	Integer	Optional. Identifier that is created differently for every setup.
	<NFLineID >	Integer	Optional. Index of the 3D line in the design area across which the near fields are calculated.
Return Value	<NumSamples>	String	Optional. Number of points (samples) needed to calculate the near fields across the line. Use the form "<num>". The default value is "1000" .
	None.		

Python Syntax	<pre>oModule.Add(["NAME:<Name>",</pre>

	<pre> "Type:=", "NearFieldLine", "VersionID:=", 0, "ID:=", <ID>, "NFLineID:=", "<NFLineID>", "NumSamples:=", "<NumSamples>"])</pre>
Python Example	<pre> oDesign = oDesign.GetModule("RadiationSetupMgr") oModule.Add(["NAME:Line 1", "Type:=", "NearFieldLine", "VersionID:=", 0, "ID:=", 7, "NFLineID:=", 312, "NumSamples:=", "1000"])</pre>

VB Syntax	<pre>oModule.Add Array("NAME:<Name>", "Type:=", "NearFieldLine", "VersionID:=", 0, "ID:=", <ID>, "NFLineID:=", <NFLineID>, "NumSamples:=", "<NumSamples>")</pre>
VB Example	<pre> Set oModule = oDesign.GetModule("RadiationSetupMgr") oModule.Add Array("NAME:Line 1", "Type:=", "NearFieldLine", "VersionID:=", 0, "ID:=",</pre>

	<pre>— 4, "NFLineID:", 312, "NumSamples:", "1000")</pre>

Edit (RadiationSetupMgr for Near Field Line)

Changes a near-field radiation boundary in the shape of a line. To calculate the near fields, you must have a 3D line defined across the design area before creating this type of radiation boundary.

UI Access	In the Project Manager under the design name > Radiation , double click on the name of the line.		
Parameters	Name	Type	Description
	<Name>	String	A user-defined name for the near field line.
	<Type>	String	Type of radiation setup. It should be NearFieldLine .
	<VersionID>	Integer	Optional. Set to 0 .
	<ID>	Integer	Optional. Identifier that is created differently for every setup.
	<NFLineID >	Integer	Optional. Index of the 3D line in the design area across which the near fields are calculated.
	<NumSamples>	String	Optional. Number of points (samples) needed to calculate the near fields across the line. Use the form "<num>".
Return Value	None.		

Python Syntax	<pre>oModule.Edit(["NAME:<Name>", "Type:=", "NearFieldLine",</pre>

	<pre> "VersionID:=", 0, "ID:=", <ID>, "NFLineID:=", "<NFLineID>", "NumSamples:=", "<NumSamples>"]) </pre>
Python Example	<pre> oDesign = oDesign.GetModule("RadiationSetupMgr") oModule.Edit(["NAME:Line 1", "Type:=", "NearFieldLine", "VersionID:=", 0, "ID:=", 7, "NFLineID:=", 312, "NumSamples:=", "1000"]) </pre>

VB Syntax	<pre> oModule.Edit Array("NAME:<Name>", "Type:=", "NearFieldLine", "VersionID:=", 0, "ID:=", <ID>, "NFLineID:=", <NFLineID>,"NumSamples:=", "<NumSamples>") </pre>
VB Example	<pre> Set oModule = oDesign.GetModule("RadiationSetupMgr") oModule.Edit Array("NAME:Line 1", "Type:=", "NearFieldLine", "VersionID:=", 0, "ID:=", — </pre>

```
4, "NFLineID:=", 312, "NumSamples:=", "1000")
```

Add (Near Field Plane)

Adds a near-field radiation boundary in the shape of a plane. This plane is calculated in reference to a rectangle defined by the X and Y parameters.

UI Access	In the Project Manager under the design name, right click Radiation and in the right-click menu, click Insert Near Field Setup>Plane .		
Parameters	Name	Type	Description
	<Name>	String	A user-defined name for the near field sphere.
	<Type>	String	Type of radiation setup. It should be NearFieldSphere .
	<VersionID>	Integer	Optional. Set to 0 .
	<ID>	Integer	Optional. Identifier that is created differently for every setup.
	<MinX>	String	Optional. Starting X value for the plane that the near fields will be calculated for. The default value is " -41.5mm ". Use the form "<num><unit>".
	<MaxX>	String	Optional. Final X value for the plane that the near fields will be calculated for. The default value is " 8.5mm ". Use the form "<num><unit>".
	<MinY>	String	Optional. Starting Y value for the plane that the near fields will be calculated for. The default value is " 16mm ". Use the form "<num><unit>".
	<MaxY>	String	Optional. Final Y value for the plane that the near fields will be calculated for. The default value is " 16mm ". Use the form "<num><unit>".
	<Height>	String	Optional. Height of the reference rectangle. The default value is " 1.652145mm ". Use the form "<num>".
	<ExpFactor>	String	Optional. The X and Y dimensions are multiplied by this exponential factor to determine new dimensions for the plane. For example, set

			to 1 to use the default calculated dimensions of the plane or set to 2 to double each dimension of the plane. The default value is " 1 ". Use the form "<num>".
	<UseExpFactor>	Boolean	Optional. Set to True to add the exponent factor to the plane dimensions or False to use the automatically calculated plane dimensions or user-defined plane dimensions. The default value is False .
	<CSID>	Integer	Optional. An identifier for the coordinate system.
Return Value	None.		

Python Syntax	<pre> oModule.Add(["NAME:<Name>", "Type:=", "NearFieldPlane", "VersionID:=", 0, "ID:=", <ID>, "MinX:=", "<MinX>", "MaxX:=", "<MaxX>", "MinY:=", "<MinY>", "MaxY:=", "<MaxY>", "Height:=", "<Height>", "ExpFactor:=", "<ExpFactor>", "UseExpFactor:=", <UseExpFactor>, "CSID:=", <CSID> </pre>

	]) oDesign = oDesign.GetModule("RadiationSetupMgr") oModule.Add(["NAME:Plane 1", "Type:=", "NearFieldPlane", "VersionID:=", 0, "ID:=", 7, "MinX:=", "-50mm", "MaxX:=", "25mm", "MinY:=", "-18.5mm", "MaxY:=", "31.5mm", "Height:=", "5.6mm", "ExpFactor:=", "1", "UseExpFactor:=", False, "CSID:=", -4])])
Python Example	

VB Syntax	oModule.Add Array("NAME:<Name>", "Type:=", "NearFieldPlane", "VersionID:=", 0, "ID:=", <ID>, "MinX:=", "<MinX>", "MaxX:=", "<MaxX>", "MinY:=", "<MinY>", "MaxY:=", "<MaxY>", "Height:=", "<Height>", "ExpFactor:=", "<ExpFactor>",

	"CSID:=",<CSID>)
VB Example	<pre> Set oModule = oDesign.GetModule("RadiationSetupMgr") oModule.Add Array("NAME:Plane 1", "Type:=", "NearFieldPlane", "VersionID:=", 0, "ID:=", _ 2, "MinX:=", "-49mm", "MaxX:=", "19mm", "MinY:=", "-11mm", "MaxY:=", "29mm", "Height:=", _ "4.652145mm", "ExpFactor:=", "1", "UseExpFactor:=", false, "CSID:=", -3) </pre>

Edit (RadiationSetupMgr for Near Field Plane)

Changes a near-field radiation boundary in the shape of a plane. This plane is calculated in reference to a rectangle defined by the X and Y parameters.

UI Access	In the Project Manager under the design name > Radiation , double click on the name of the plane.		
Parameters	Name	Type	Description
	<Name>	String	A user-defined name for the near field sphere.
	<Type>	String	Type of radiation setup. It should be NearFieldSphere .
	<VersionID>	Integer	Optional. Set to 0 .
	<ID>	Integer	Optional. Identifier that is created differently for every setup.
	<MinX>	String	Optional. Starting X value for the plane that the near fields will be calculated for. Use the form "<num><unit>".
	<MaxX>	String	Optional. Final X value for the plane that the near fields will be calculated for. Use the form "<num><unit>".
	<MinY>	String	Optional. Starting Y value for the plane that the near fields will be calculated for. Use the form "<num><unit>".
	<MaxY>	String	Optional. Final Y value for the plane that the near fields will be calculated for. Use the form "<num><unit>".
	<Height>	String	Optional. Height of the reference rectangle. Use the form "<num>".
	<ExpFactor>	String	Optional. The X and Y dimensions are multiplied by this exponential

			factor to determine new dimensions for the plane. For example, set to 1 to use the default calculated dimensions of the plane or set to 2 to double each dimension of the plane. Use the form "<num>".
	<UseExpFactor>	Boolean	Optional. Set to True to add the exponent factor to the plane dimensions or False to use the automatically calculated plane dimensions or user-defined plane dimensions.
	<CSID>	Integer	Optional. An identifier for the coordinate system.
Return Value	None.		

Python Syntax	<pre>oModule.Edit(["NAME:<Name>", "Type:=", "NearFieldPlane", "VersionID:=", 0, "ID:=", <ID>, "MinX:=", "<MinX>", "MaxX:=", "<MaxX>", "MinY:=", "<MinY>", "MaxY:=", "<MaxY>", "Height:=", "<Height>", "ExpFactor:=", "<ExpFactor>",</pre>		

	<pre> "UseExpFactor:=", <UseExpFactor>, "CSID:=", <CSID>]) </pre>
Python Example	<pre> oDesign = oDesign.GetModule("RadiationSetupMgr") oModule.Edit(["NAME:Plane 1", "Type:=", "NearFieldPlane", "VersionID:=", 0, "ID:=", 7, "MinX:=", "-50mm", "MaxX:=", "25mm", "MinY:=", "-18.5mm", "MaxY:=", "31.5mm", "Height:=", "5.6mm", "ExpFactor:=", "1", "UseExpFactor:=", False, "CSID:=", -4]) </pre>

VB Syn-	oModule.Edit Array("NAME:<Name>", "Type:=", "NearFieldPlane", "VersionID:=", 0, "ID:=", <ID>, "MinX:=", "<MinX>",

tax	"MaxX:=", "<MaxX>", "MinY:=", "<MinY>", "MaxY:=", "<MaxY>", "Height:=", "<Height>", "ExpFactor:=", "<ExpFactor>", "CSID:=", "<CSID>")
VB Example	<pre>Set oModule = oDesign.GetModule("RadiationSetupMgr") oModule.Edit Array("NAME:Plane 1", "Type:=", "NearFieldPlane", "VersionID:=", 0, "ID:=", _ 2, "MinX:=", "-49mm", "MaxX:=", "19mm", "MinY:=", "-11mm", "MaxY:=", "29mm", "Height:=", _ "4.652145mm", "ExpFactor:=", "1", "UseExpFactor:=", false, "CSID:=", -3)</pre>

Add (Near Field Sphere)

Adds a near-field radiation boundary in the shape of a sphere.

Note: Only *Name* and *Type* are required parameters. If you set a custom Theta or Phi parameter, then you must set all Theta and Phi parameters. If you have a local coordinate system, then you must set the *CS* and *UseLocalCS* parameters.

UI Access	In the Project Manager under the design name, right click Radiation and in the right-click menu, click Insert Near Field Setup>Sphere .		
Parameters	Name	Type	Description
	<Name>	String	A user-defined name for the near field sphere.
	<Type>	String	Type of radiation setup. It should be NearFieldSphere .
	<VersionID>	Integer	Optional. Set to 0 .
	<ID>	Integer	Optional. Identifier that is created differently for every setup.
	<StartTheta >	String	Optional. Starting point for theta in the spherical coordinate system for the sphere being defined. The default value is " 0deg ". Use the form "<num> deg ".

	<StartPhi>	String	Optional. Starting point for phi in the spherical coordinate system for the sphere being defined. The default value is " -180deg ". Use the form "<num> deg ".
	<StopTheta>	String	Optional. Stopping point for theta in the spherical coordinate system for the sphere being defined. The default value is " 180deg ". Use the form "<num> deg ".
	<StopPhi>	String	Optional. Stopping point for phi in the spherical coordinate system for the sphere being defined. The default value is " 180deg ". Use the form "<num> deg ".
	<StepTheta>	String	Optional. Step size between the start and the stop theta values at which the far field is calculated. The default value is " 2deg ". Use the form "<num> deg ".
	<StepPhi>	String	Optional. Step size between the start and the stop phi values at which the far field is calculated. The default value is " 2deg ". Use the form "<num> deg ".
	<CS>	Integer	Optional. The coordinate system to use. When coordinate system is local, set this number to the index of the local coordinate system. Set this value to -1 for a global coordinate system. The default value is -1 .
	<UseLocalCS>	Boolean	Optional. True to select a local coordinate system, and False to use the global coordinate system. The default value is False .
	<NFRadius>	String	Optional. Radius of the sphere we want to calculate near fields on. The default value is " 20mm ". Use the form "<num><unit>".
Return Value	None.		

Python Syntax	<pre>oModule.Add(["NAME:<Name>", "Type:=", "InfiniteSphereFarField",</pre>

	<pre>"VersionID:=", 0, "ID:=", <ID>, "StartTheta:=", "<StartTheta>deg", "StartPhi:=", "<StartPhi>deg", "StopTheta:=", "<StopTheta>deg", "StopPhi:=", "<StopPhi>deg", "StepTheta:=", "<StepTheta>deg", "StepPhi:=", "<StepPhi>deg", "CS:=", <CS>, "UseLocalCS:=", <UseLocalCS>, "NFRadius:=", "<NFRadius>"])</pre>
Python Example	<pre>oDesign = oDesign.GetModule("RadiationSetupMgr") oModule.Add(["NAME:Sphere 1", "Type:=", "NearFieldSphere", "VersionID:=", 0, "ID:=", 7, "StartTheta:=", "0deg",</pre>

	<pre> "StartPhi:=", "-180deg", "StopTheta:=", "180deg", "StopPhi:=", "180deg", "StepTheta:=", "2deg", "StepPhi:=", "2deg", "CS:=", -1, "UseLocalCS:=", False, "NFRRadius:=", "20mm"]) </pre>

VB Syntax	<pre> oModule.Add Array("NAME:<Name>", "Type:=", "NearFieldSphere", "VersionID:=", 0, "ID:=", <ID>, "StartTheta:=", "<StartTheta>deg", "StartPhi:=", "<StartPhi>deg", "StopTheta:=", "<StopPhi>deg", "StopPhi:=", "<StopPhi>deg", "StepTheta:=", "<StepTheta>deg", "StepPhi:=", "<StepPhi>deg", "CS:=", <CS>, "UseLocalCS:=", <UseLocalCS>,"NFRRadius:=", " <NFRRadius>") </pre>
VB Example	<pre> Set oModule = oDesign.GetModule("RadiationSetupMgr") oModule.Add Array("NAME:Sphere 1", "Type:=", "NearFieldSphere", "VersionID:=", 0, "ID:=", _ 3, "StartTheta:=", "0deg", "StartPhi:=", "-180deg", "StopTheta:=", "180deg", "StopPhi:=", _ "180deg", "StepTheta:=", "2deg", "StepPhi:=", "2deg", "CS:=", -1, "UseLocalCS:=", _ false, "NFRRadius:=", "20mm") </pre>

Edit (RadiationSetupMgr for Near Field Sphere)

Changes a near-field radiation boundary in the shape of a sphere.

Note: Only *Name* and *Type* are required parameters. If you set a custom Theta or Phi parameter, then you must set all Theta and Phi parameters. If you have a local coordinate system, then you must set the *CS* and *UseLoca/CS* parameters.

UI Access	In the Project Manager under the design name > Radiation , double click on the name of the near-field sphere.		
Parameters	Name	Type	Description
	<Name>	String	A user-defined name for the near field sphere.
	<Type>	String	Type of radiation setup. It should be NearFieldSphere .
	<VersionID>	Integer	Optional. Set to 0 .
	<ID>	Integer	Optional. Identifier that is created differently for every setup.
	<StartTheta >	String	Optional. Starting point for theta in the spherical coordinate system for the sphere being defined. Use the form "<num> deg ".
	<StartPhi>	String	Optional. Starting point for phi in the spherical coordinate system for the sphere being defined. Use the form "<num> deg ".
	<StopTheta>	String	Optional. Stopping point for theta in the spherical coordinate system for the sphere being defined. Use the form "<num> deg ".
	<StopPhi>	String	Optional. Stopping point for phi in the spherical coordinate system for the sphere being defined. Use the form "<num> deg ".
	<StepTheta>	String	Optional. Step size between the start and the stop theta values at which the far field is calculated. Use the form "<num> deg ".
	<StepPhi>	String	Optional. Step size between the start and the stop phi values at which the far field is calculated. Use the form "<num> deg ".
	<CS>	Integer	Optional. The coordinate system to use. When coordinate system is local, set this number to the index of the local coordinate system. Set this value to -1 for a global coordinate system.

	<UseLocalCS>	Boolean	Optional. True to select a local coordinate system, and False to use the global coordinate system.
	<NFRadius>	String	Optional. Radius of the sphere we want to calculate near fields on. Use the form "<num><unit>".
Return Value	None.		

Python Syntax	<pre> oModule.Edit(["NAME:<Name>", "Type:=" , "InfiniteSphereFarField", "VersionID:=" , 0, "ID:=" , <ID>, "StartTheta:=" , "<StartTheta>deg", "StartPhi:=" , "<StartPhi>deg", "StopTheta:=" , "<StopTheta>deg", "StopPhi:=" , "<StopPhi>deg", "StepTheta:=" , "<StepTheta>deg", "StepPhi:=" , "<StapPhi>deg", "CS:=" , <CS>, "UseLocalCS:=" , <UseLocalCS>, "NFRadius:=" , "<NFRadius>"]) </pre>

Python Example	<pre>oDesign = oDesign.GetModule("RadiationSetupMgr") oModule.Edit(["NAME:Sphere 1", "Type:=", "NearFieldSphere", "VersionID:=", 0, "ID:=", 7, "StartTheta:=", "0deg", "StartPhi:=", "-180deg", "StopTheta:=", "180deg", "StopPhi:=", "180deg", "StepTheta:=", "2deg", "StepPhi:=", "2deg", "CS:=", -1, "UseLocalCS:=", False, "NFRadius:=", "20mm"])</pre>

VB Syntax	<pre>oModule.Edit Array("NAME:<Name>", "Type:=", "NearFieldSphere", "VersionID:=", 0, "ID:=", <ID>, "StartTheta:=", "<StartTheta>deg", "StartPhi:=", "<StartPhi>deg", "StopTheta:=", "<StopPhi>deg", "StopPhi:=", "<StopPhi>deg",</pre>

	<code>"StepTheta:=", "<StepTheta>deg", "StepPhi:=", "<StepPhi>deg", "CS:=", "<CS>", "UseLocalCS:=", <UseLocalCS>,"NFRadius:=", " <NFRadius>")</code>
VB Example	<pre>Set oModule = oDesign.GetModule("RadiationSetupMgr") oModule.Edit Array("NAME:Sphere 1", "Type:=", "NearFieldSphere", "VersionID:=", 0, "ID:=", _ 3, "StartTheta:=", "0deg", "StartPhi:=", "-180deg", "StopTheta:=", "180deg", "StopPhi:=", _ "180deg", "StepTheta:=", "2deg", "StepPhi:=", "2deg", "CS:=", -1, "UseLocalCS:=", _ false, "NFRadius:=", "20mm")</pre>

This page intentionally
left blank.

17 - 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 Results > Create Document . Choose a document name.		
Parameters	Name	Type	Description
	<data>	Array	Data the defines the document.
	<traces>	Array	trace data for the inputs in 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 Syntax	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("Nom-
inal"), _
"Pulse_Width:=", Array("Nominal"), "Data_Rate:=", Array("Nominal"), _
"Length:=", Array("Nominal")), Array("Probe Component:=", Array("DQ0")), Array()))

```

DeleteAllDocuments

Deletes all documents in the object.

UI Access	Right click on Documents in the Project Manager and click Delete All Documents .
Parameters	None.
Return Value	None.

Python Syntax	DeleteAllDocuments()
Python Example	oModule.DeleteAllDocuments()

VB Syntax	DeleteAllDocuments

VB Example	<code>oModule.DeleteAllDocuments</code>

DeleteDocument

Deletes a specified document.

UI Access	Right click on the created document in the Project Manager under Results > Documents and click Delete .		
Parameters	Name	Type	Description
	<name>	String	Name of the document to be deleted.
Return Value	None.		

Python Syntax	<code>DeleteDocument(<names>)</code>
Python Example	<code>oModule.DeleteDocument("Project Design Summary")</code>

VB Syntax	<code>DeleteDocument <names></code>
VB Example	<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.

UI Access	Right click on the created document in the Project Manager under Results > Documents and click Modify document .

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.		

Python Syntax	EditDocument(<name>,<data>,<traces>)
Python Example	<pre>oModule.EditDocument("Design Summary", ["NAME:Design Summary", "", "SysLib", "DesignSummary", ["NAME:Inputs"]], ["NAME:DocTraces"])</pre>

VB Syntax	EditDocument <name>,<data>,<traces>

VB Example	<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

UI Access	NA		
Parameters	Name	Type	Description
	<separator>	String	Separator used to convey the directory "level"
Return Value	None.		

Python Syntax	GetDocumentDefinitionNames(<separator>)
Python Example	oModule.GetDocumentDefinitionNames("/")

VB Syntax	GetDocumentDefinitionNames <separator>
VB Example	oModule.GetDocumentDefinitionNames "/"

GetDocumentNames

Retrieves the names for all documents.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing document names.

Python Syntax	GetDocumentNames()
Python Example	<code>oModule.GetDocumentNames()</code>

VB Syntax	GetDocumentNames
VB Example	<code>oModule.GetDocumentNames</code>

RenameDocument

Changes the name of a document.

UI Access	Right click on the created document in the Project Manager under Results> Documents and click Rename .		
Parameters	Name	Type	Description
	<oldName>	String	Current name of the document
	<newName>	String	New name of the document

Return Value	None.

Python Syntax	RenameDocument(<oldName>, <newName>)
Python Example	oModule.RenameDocument("Design Summary", "Project Design Summary")

VB Syntax	RenameDocument <oldName>, <newName>
VB Example	oModule.RenameDocument "Design Summary", "Project Design Summary"

SaveHtmlDocumentAs

Saves a pre-existing HTML file to a different name and/or location.

UI Access	Right click on the created document in the Project Manager under Results > Documents and click Save As > HTML .		
Parameters	Name	Type	Description
	<name>	String	Name of the document to be saved.
	<saveTo>	String	File path to save the document to.
Return Value	none		

Python Syntax	SaveHtmlDocumentAs(<name>, <saveTo>)
Python Example	oModule.SaveHtmlDocumentAs("Design Summary 1", "DS1.html")

VB Syntax	SaveHtmlDocumentAs <name> <saveTo>
VB Example	<code>oModule.SaveHtmlDocumentAs "Design Summary 1" "DS1.html"</code>

SavePdfDocumentAs

Saves a pre-existing PDF file to a different name and/or location.

UI Access	Right click on the created document in the Project Manager under Results > Documents and click Save As > PDF .		
Parameters	Name	Type	Description
	<name>	String	Name of the document to be saved.
	<saveTo>	String	File path to save the document at.
Return Value	None.		

Python Syntax	SavePdfDocumentAs(<name>, <saveTo>)
Python Example	<code>oModule.SavePdfDocumentAs("Design Summary 1", "DS1.pdf")</code>

VB Syntax	SavePdfDocumentAs <name>, <saveTo>
VB Example	<code>oModule.SavePdfDocumentAs "Design Summary 1", "DS1.pdf"</code>

UpdateAllDocuments

Refreshes the contents of all created documents. This action is made on the folder rather than the individual document.

UI Access	Right click on Results > Documents in the Project Manager and click Update All Documents .
Parameters	None.
Return Value	None.

Python Syntax	UpdateAllDocuments()
Python Example	<code>oModule.UpdateAllDocuments()</code>

VB Syntax	UpdateAllDocuments
VB Example	<code>oModule.UpdateAllDocuments</code>

UpdateDocument

Refreshes the contents of the selected document.

UI Access	Right click on the created document in the Project Manager under Results > Documents and click Update Document .		
Parameters	Name	Type	Description
	<name>	String	Name of the document to be updated
Return Value	None.		

Python Syntax	UpdateDocument(<name>)
Python Example	oModule.UpdateDocument("Test UDD Report")

VB Syntax	UpdateDocument <name>
VB Example	oModule.UpdateDocument "Test UDD Report"

ViewHtmlDocument

Displays a pre-existing document as HTML.

UI Access	Right click on the created document in the Project Manager under Results > Documents and click View Xml Document .		
Parameters	Name	Type	Description
	<name>	String	Name of the document to be viewed as a HTML
Return Value	None		

Python Syntax	ViewHtmlDocument(<name>)
Python Example	oModule.ViewHtmlDocument("Design Summary 1")

VB Syntax	ViewHtmlDocument <name>
VB Example	oModule.ViewHtmlDocument "Design Summary 1"

ViewPdfDocument

Displays a pre-existing document as a PDF file.

UI Access	Right click on the created document in the Project Manager under Results > Documents and click View PDF Document .		
Parameters	Name	Type	Description
	<name>	String	Name of the document to be viewed as a PDF
Return Value	None.		

Python Syntax	ViewPdfDocument(<name>)
Python Example	oModule.ViewPdfDocument("Design Summary 1")

VB Syntax	ViewPdfDocument <name>
VB Example	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",	' Input name
"Solution",	' Solution Input Type
"Data Line Metrics",	' Input Description
-1,	' Solution ID
-1),	' Report ID

' This array contains the trace input:

Array("NAME:DQ0",	' Input name
"Trace",	' Trace Input Type
"DQ0",	' Input Description
-1,	' Solution ID
-1),	' Report ID

' This array contains the text input:

Array("NAME:Name",	' Input name
"Text",	' Text Input Type
"User Name",	' Input Description
Array("Sita Ramesh")),	' Default Value

' This array contains the Bool input:

Array("NAME:Summary",	' Input name
"Bool",	' Boolean Input Type
"Display Summary",	' Input Description
Array(true)),	' 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",
        "probel",
        "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.

18 - 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](#)

[GetUserDefinedSolutionNames](#)

[GetUserDefinedSolutionProperties](#)

CreateUserDefinedSolution

Creates a new user defined solution.

UI Access	Right-click on Results > Create User Defined Solution .		
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 <i><LibType></i> .
	<i><PropList></i>	Array	Strings specify name-value pairs corresponding to the UDS properties specified in the plugin file. For example: <code>Array("multiply_factor:=", "2.0", "component_name:=", "resistor1")</code>
	<i><ProbeSelections></i>	Array	Name of the probe being specified. Note: this must match a probe name specified in the UDS plugin file.
	<i><DynamicProbes></i>	Array	Array of <i><ProbeSelection></i> 's, representing the probes that are used by dynamic-probes.
Return Value	String name of created user defined solution.		

Python Syntax	<code>CreateUserDefinedSolution(<SoluName>, <LibType>, <RelativePath>, <PropList>, <ProbeSelections>, <DynamicProbes>)</code>		
Python Example	<pre>oModule.CreateUserDefinedSolution("ConstantTimestep1", "SysLib", "ConstantTimestep", [], [], [])</pre>		

VB Syntax	<code>CreateUserDefinedSolution <SoluName>, <LibType>, <RelativePath>, <PropList>, <ProbeSelections>, <DynamicProbes></code>		
VB Example	<pre>oModule.CreateUserDefinedSolution "User Defined Solution 1", _ "SysLib", "Example", _</pre>		

	<pre>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(), _ Array("Freq:=", Array("All")), _ Array("Probe Component:=", Array("Freq")), Array()))</pre>

DeleteUserDefinedSolutions

Deletes one or more user defined solutions.

UI Access	Delete button from the User Defined Solutions dialog.		
Parameters	Name	Type	Description
	<SoluNames>	Array	Array of strings containing names of User Defined Solutions to be deleted.
Return Value	None.		

Python Syntax	DeleteUserDefinedSolutions(<SoluNames>)
Python Example	<code>oModule.DeleteUserDefinedSolutions(["Solution1", "Solution2"])</code>

VB Syntax	DeleteUserDefinedSolutions <SoluNames>
VB Example	<code>oModule.DeleteUserDefinedSolutions _ Array("Solution1", "Solution2")</code>

EditUserDefinedSolution

Modifies an existing user defined solution.

UI Access	Edit button from the User Defined Solutions dialog box.		
Parameters	Name	Type	Description
	<SoluName>	String	Name of user defined solution to be edited.
	<NewSoluName>	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: <code>Array("multiply_factor:=", "2.0", "component_name:=", "resistor1")</code>
	<ProbeSelections>	Array	Name of the probe being specified. Note: this must match a probe name spe-

			cified in the UDS plugin file.
	<DynamicProbes>	Array	Array of <ProbeSelection>'s, representing the probes that are used by dynamic-probes.
Return Value	String name of update user defined solution.		

Python Syntax	EditUserDefinedSolution(<SoluName>, <NewSoluName>, <LibType>, <RelativePath>, <PropList>, <ProbeSelections>, <DynamicProbes>)		
Python Example	<pre>oModule.EditUserDefinedSolution("ConstantTimestep1" "ConstantTimestep1After", "SysLib", "ConstantTimestep", [], [], [])</pre>		

VB Syntax	EditUserDefinedSolution <SoluName>, <NewSoluName>, <LibType>, <RelativePath>, <PropList>, <ProbeSelections>, <DynamicProbes>		
VB Example	<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(), _</pre>		

```
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()))
```

GetUserDefinedSolutionNames

Retrieves user defined solution names.

UI Access	N/A
Parameters	None.
Return Value	Array of strings containing solution names.

Python Syntax	GetUserDefinedSolutionNames()
Python Example	oModule.GetUserDefinedSolutionNames()

VB Syntax	GetUserDefinedSolutionNames
VB Example	oModule.GetUserDefinedSolutionNames

GetUserDefinedSolutionProperties

Obtains properties for a specified user defined solution.

UI Access	N/A		
Parameters	Name	Type	Description
	<SoluName>	String	Name of a specified user defined solution.
Return Value	Array of strings containing properties values.		

Python Syntax	GetUserDefinedSolutionProperties(<SoluName>)
Python Example	<code>oModule.GetUserDefinedSolutionProperties("ConstantTimestep1")</code>

VB Syntax	GetUserDefinedSolutionProperties <SoluName>
VB Example	<code>oModule.GetUserDefinedSolutionProperties "ConstantTimestep1"</code>

This page intentionally
left blank.

19 - 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 Network Data Explorer tab, select Differential Pairs in the NDE ribbon to open the Differential Pairs window. Then select differential pairs from the Pairs group box and click Add Pairs >> .		
Parameters	Name	Type	Description
	<positiveTerminal>	Integer	The portNumber of the positive terminal.
	<negativeTerminal>	Integer	The portNumber of the negative terminal.
	<diffName>	String	The new name of the differential pin (e.g., Diff1).
	<commName>	String	The new name of the common pin (e.g., Comm1).
	<diffImpedance>	Double	The differential pin characteristic impedance.
	<commImpedance>	Double	The common pin characteristic impedance.
	<matched>	Boolean	Specify whether to use matched pair.
	Note: The default value is False .		
Return Value	Boolean.		

Python Syntax	AddDiffPair()
Python Example	<pre>success = oPostProc.AddDiffPair(2, 1, "Diff1", "Comm1", 100, 25)</pre>

VB Syntax	AddDiffPair

VB Example	<code>Set success = oPostProc.AddDiffPair 2, 1, "Diff1", "Comm1", 100, 25</code>

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.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Cascade from the drop-down menu.		
Parameters	Name	Type	Description
	<code><dataArray></code>	Array	These network data objects are cascaded together to create the new network data.
Return Value	Network IDispatch. Note: Cascade returns None if the specified network data does not have compatible terminal ports defined.		

Python Syntax	<code>Cascade()</code>
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Cascade([oData1, oData2, oData3])</pre>

VB Syntax	Cascade
VB Example	<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.

UI Access	From the Network Data Explorer tab, select Differential Pairs in the NDE ribbon to open the Differential Pairs window. Then select differential pairs from the rightmost box and click << Remove Pairs .
Parameters	None.
Return Value	None.

Python Syntax	ClearDiffPairs()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.ClearDiffPairs()</pre>

VB Syntax	ClearDiffPairs
VB Example	<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.

UI Access	From the Network Data Explorer tab, select the network data object to clone. Then select Clone from the NDE ribbon.
Parameters	None.
Return Value	Network IDispatch.

Python Syntax	Clone()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData1 = oData.Clone()</pre>

VB Syntax	Clone
VB Example	<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.

UI Access	From the Network Data Explorer tab, click Close in the NDE ribbon, or select File > Close .
Parameters	None.
Return Value	None.

Python Syntax	<code>Close()</code>
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData.Close()</pre>

VB Syntax	<code>Close</code>
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") oData.Close</pre>

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 Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Combine from the drop-down menu.		
Parameters	Name <dataArray>	Type Array	Description These network data objects are combined to create the new network data.
	<freqTol>	Double	If $\text{abs}(f1 - f2) < \text{freqTol}$, f1 and f2 are considered the same frequency. Note: The default value is 100 .
Return Value	Network IDispatch. Note: Combine returns None if the network data does not have the same number of ports.		

Python Syntax	Combine()
Python Example	<code>oNDE = oDesktop.GetTool("ndExplorer")</code>

	<code>oData4 = oNDE.Combine([oData1, oData2, oData3], 1000)</code>

VB Syntax	Combine
VB Example	<code>Set oNDE = oDesktop.GetTool("ndExplorer")</code> <code>Set oData4 = oNDE.Combine Array(oData1, oData2, oData3), 1000</code>

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, dembedding the frontData and the backData from the front and back of the originalData.

Note: Deembed opens and interacts with **SPISim** to complete.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Deembed from the drop-down menu to open the Deembed... window, and choose Given 3 S-params in order of: Total, Front, Back from the Settings group box.		
Parameters	Name	Type	Description
	<code><originalData></code>	Object	Original network data that is deembeded.
	<code><frontData></code>	Object	Network data that is deembeded from the front of the original.
	<code><backData></code>	Object	Network data that is deembeded from the back of the original.
Return Value	Network IDispatch.		

Note: Deembed returns **None** if the specified network data does not have compatible terminal ports defined.

Python Syntax	Deembed()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Deembed(oData1, oDataFront, oDataBack)</pre>

VB Syntax	Deembed
VB Example	<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 Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Deembed from the drop-down menu to open the Deembed... window, and choose Given 3 S-params in order of: Total, Back from the Settings group box.

Parameters	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	Network IDispatch.		
	Note: DeembedBack returns None if the specified network data does not have compatible terminal ports defined.		

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, dembedding the frontData from the front of the originalData.

Note: DeembedFront opens and interacts with **SPISim** to complete.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Deembed from the drop-down menu to open the Deembed... window, and choose Given 3 S-params in order of: Total, Front from the Settings group box.		
Parameters	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	Network IDispatch. Note: DeembedFront returns None if the specified network data does not have compatible terminal ports defined.		

Python Syntax	DeembedFront()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.DeembedFront(oData1, oDataFront)</pre>

VB Syntax	DeembedFront
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer")</pre>

	Set oData2 = oNDE.DeembedFront oData1, oDataFront

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](#).

UI Access	From the Network Data Explorer tab, select Differential Pairs in the NDE ribbon to open the Differential Pairs window. Once differential pairs have been added to the rightmost box (i.e., select differential pairs from the Pairs group box and click Add Pairs >>), remove check marks from the Enabled column to deactivate the corresponding differential pairs.
Parameters	None.
Return Value	None.

Python Syntax	DisableDiffPairs()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.DisableDiffPairs()</pre>

VB Syntax	DisableDiffPairs
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer")</pre>

	<code>oPostProc.DisableDiffPairs</code>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

EnableDiffPairs

Enables all differential pair definitions.

UI Access	From the Network Data Explorer tab, select Differential Pairs in the NDE ribbon to open the Differential Pairs window. Once differential pairs have been added to the rightmost box (i.e., select differential pairs from the Pairs group box and click Add Pairs >>), add check marks to the Enabled column to activate the corresponding differential pairs.
Parameters	None.
Return Value	None.

Python Syntax	<code>EnableDiffPairs()</code>
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oPostProc.EnableDiffPairs()</pre>

VB Syntax	<code>EnableDiffPairs</code>
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set success = oPostProc.EnableDiffPairs</pre>

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).

UI Access	From the Network Data Explorer tab, select File > Save As to open a Save As explorer window. Then select Citifile (*.cit) from the Save as type: drop-down menu.		
Parameters	Name	Type	Description
	<filePath>	String	The full path to the file.
	<matrixType>	Integer	One of the following: • 0 – S-Parameters • 1 – Y-Parameters • 2 – Z-Parameters • 3 – Gamma • 4 – Impedance Note: The default value is 0.
	<formalType>	Integer	One of the following: • 0 – Magnitude/Angle (degrees) • 1 – Real/Imaginary • 2 – dB/Angle (degrees)

			Note: The default value is 0 .
	<precision>	Integer	The number of significant figures. Note: The default value is 10 .
	<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. Note: The default value is an empty array.
Return Value	Boolean True if file was successfully written; else False.		

Python Syntax	ExportCitiFile()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.ExportCitiFile("D:\folder\nel133.cit", 0, 1, 12, [])</pre>

VB Syntax	ExportCitiFile
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set success = oData.ExportCitiFile "D:\folder\nel133.cit", 0, 1, 12, Array()</pre>

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"
```

```

"RelativeErrorToleranc:=", 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
)

```

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 Network Data Explorer tab, select File > Save As to open a Save As explorer window. Then select MATLAB (*.m) from the Save as type: drop-down menu.		
Parameters	Name	Type	Description
	<filePath>	String	The full path to the file.
	<matrixType>	Integer	One of the following: • 0 – S-Parameters • 1 – Y-Parameters • 2 – Z-Parameters

			• 3 – Gamma • 4 – Impedance Note: The default value is 0.
	<i><formalType></i>	Integer	One of the following: • 0 – Magnitude/Angle (degrees) • 1 – Real/Imaginary • 2 – dB/Angle (degrees) Note: The default value is 0.
	<i><precision></i>	Integer	The number of significant figures. Note: The default value is 10.
	<i><frequencies></i>	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. Note: The default value is an empty array.
Return Value	Boolean True if file was successfully written; else False.		

Python Syntax	ExportMatlab()
Python	<code>oNDE = oDesktop.GetTool("ndExplorer")</code>

Example	<code>success = oData.ExportMatlab("D:\folder\nel33.m", 0, 1, 12, [])</code>

VB Syntax	ExportMatlab
VB Example	<pre>Set oNDE = oDesktop.GetTool "ndExplorer" Set success = oData.ExportMatlab "D:\folder\nel33.m", 0, 1, 12, Array()</pre>

ExportNMFData

Exports s-parameters in neutral file format (*.nmf).

UI Access	Results > Solution Data. On the Matrix tab, click Export tab. Then click Export > S-parameter . Select Neutral Model format and export.		
Parameters	Name	Type	Description
	<sourceName>	Array	String names of solutions. Use more than one array entry for a combined Interpolating sweep.
	<outputFileName>	String	Full path of output file, including *.nmf extension.
	<reduceMat>	String	Either "Original" or name of reduced matrix.
	<refImp>	Double	Reference impedance.
	<freqs>	Array	Either "All" or array of doubles representing frequencies.
	<variationKey>	String	Design variation key. Pass empty string for the current nominal variation.
	<displayFormat>	String	"MagPhase", "Reallmag", "DbPhase".
	<pass>	Integer	Optional. The pass to export. This is ignored if the <i>sourceName</i> is a frequency sweep. Leaving out this value or specifying -1 gets all passes.
	<ComplexFormat>	Integer	Optional. The format to use for the exported data. Options are 0 = Magnitude/Phase

			1= Real/Imaginary. 2= db/Phase.
	<Precision>	Integer	Optional. Number of digits precision. If not specified, the default is 6.
Return Value	None.		

Python Syntax	ExportNMFData (<sourceName>, <outputFileName>,<reduceMat>, <refImp>, <freqs>,<variationKey>, <displayFormat>, <pass>, <ComplexFormat>, <Precision>)		
Python Example	<pre>oDesign.ExportNMFData("Setup2 : LastAdaptive", "C:/outputs/data.nmf, "Original", 50, [10000000, 20000000, 30000000, 40000000, 50000000, 60000000, 70000000, 80000000, 90000000, 100000000], "", "MagPhase", 0, 1, 11)</pre>		

VB Syntax	ExportNMFData(<sourceName>, <outputFileName>,<reduceMat>, <refImp>, <freqs>,<variationKey>, <displayFormat>, <pass>, <ComplexFormat>, <Precision>)		
VB Example	<pre>dim solns[1] solns[0] = "Setup1:Sweep1" solns[1] = "Setup1:Sweep2" dim freqs[5] freqs[0] = 10000000 freqs[1] = 20000000 freqs[2] = 30000000 freqs[3] = 40000000 freqs[4] = 50000000</pre>		

```

freqs[5] = 60000000

oDesign.ExportNMFData(solns, "C:/outputs/data.nmf", "Original", 3.2133, freqs,
"", "MagPhase", 2, 1, 11)

```

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 syntax, but ignored if DoRenorm is false.
	<DataType>	String	Optional. Type: "S", "Y", or "Z". The matrix to export.
	<pass>	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.
	<ComplexFormat>	Integer	Optional. Type: "0", "1", or "2" The format to use for the exported data. 0 = Magnitude/Phase. 1 = Real/Imaginary. 2 = db/Phase.
	<Precision>	Integer	Optional. Touchstone number of digits precision. Default if not specified is 15.
	<UseExportFreqs>	Boolean	Specifies whether to use export frequencies.
	<IncludeGammaComments>	Boolean	Specifies whether to include Gamma and Impedance comments.
	<SupportNonStdExport>	Boolean	Specifies whether to support non-standard Touchstone extensions for mixed reference impedance.
Return Value	None.		

Python Syntax	ExportNetworkData(<DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>, <FreqsArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <SupportNonStdExport>])
Python	oModule.ExportNetworkData("", ["Setup1:Sweep1"], 3, "C://Documents/package_

Example	HFSSDesign1.s4p", ['all'], True, 50, "S", -1, 0, 15, True, True, True

VB Syntax	ExportNetworkData <DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>, <FreqsArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <SupportNonStdExport>]
VB Example	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

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 Network Data Explorer tab, select File > Save As to open a Save As explorer window. Then select Data Table (spreadsheet) (*.tab) from the Save as type: drop-down menu.

Parameters	Name	Type	Description
	<filePath>	String	The full path to the file.
	<matrixType>	Integer	One of the following: • 0 – S-Parameters • 1 – Y-Parameters • 2 – Z-Parameters • 3 – Gamma • 4 – Impedance Note: The default value is 0.
	<formalType>	Integer	One of the following: • 0 – Magnitude/Angle (degrees) • 1 – Real/Imaginary • 2 – dB/Angle (degrees) Note: The default value is 0.
	<precision>	Integer	The number of significant figures. Note: The default value is 10.
	<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. Note: The default value is an empty array.

	<renormalize>	Boolean	If True , will renormalize the data to the value of renormImpedance before writing. Note: The default value is False .
	<renormImpedance>	Double	Data will be renormalized to this impedance before writing only if renormalize is True . Note: The default value is 50 ohms .
Return Value	Boolean True if file was successfully written; else False.		

Python Syntax	ExportSpreadsheet()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData = oNDE.Open(<model path>) success = oData.ExportSpreadsheet("D:\folder\ne133.tab", 0, 1, 12, [], True, 23)</pre>

VB Syntax	ExportSpreadsheet
VB Example	<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).

UI Access	From the Network Data Explorer tab, select File > Save As to open a Save As explorer window. Then select Touchstone Format 1.0 (*.snp) from the Save as type: drop-down menu.		
Parameters	Name	Type	Description
	<filePath>	String	The full path to the file.
	<matrixType>	Integer	One of the following: • 0 – S-Parameters • 1 – Y-Parameters • 2 – Z-Parameters • 3 – Gamma • 4 – Impedance Note: The default value is 0.
	<formalType>	Integer	One of the following: • 0 – Magnitude/Angle (degrees) • 1 – Real/Imaginary • 2 – dB/Angle (degrees)

			Note: The default value is 0 .
	<precision>	Integer	The number of significant figures. Note: The default value is 10 .
	<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. Note: The default value is an empty array.
	<renormalize>	Boolean	If True , will renormalize the data to the value of renormImpedance before writing. Note: The default value is False .
	<renormImpedance>	Double	Data will be renormalized to this impedance before writing only if renormalize is True . Note: The default value is 50 ohms .
	<freqUnits>	String	One of the following: "Hz", "KHz", "MHz", "GHz". Note: The default value is "Hz" .
Return Value	Boolean True if file was successfully written; else False.		

Python	ExportTouchstone()

Syntax	
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.ExportTouchstone("D:\folder\ne133.s2p", 0, 1, 12, [], True, 23, "GHz")</pre>

VB Syntax	ExportTouchstone
VB Example	<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 Network Data Explorer tab, select File > Save As to open a Save As explorer window. Then select Touchstone 2 Format (*.ts) from the Save as type: drop-down menu.		
Parameters	Name	Type	Description
	<filePath>	String	The full path to the file.
	<matrixType>	Integer	One of the following: • 0 – S-Parameters • 1 – Y-Parameters

		• 2 – Z-Parameters • 3 – Gamma • 4 – Impedance Note: The default value is 0.
<formalType>	Integer	One of the following: • 0 – Magnitude/Angle (degrees) • 1 – Real/Imaginary • 2 – dB/Angle (degrees) Note: The default value is 0.
<precision>	Integer	The number of significant figures. Note: The default value is 10.
<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. Note: The default value is an empty array.
<renormalize>	Boolean	If True, will renormalize the data to the value of renormImpedance before writing. Note: The default value is False.
<renormImpedance>	Double	Data will be renormalized to this impedance before writing only if renormalize is True.

			Note: The default value is 50 ohms .
	<freqUnits>	String	One of the following: "Hz", "KHz", "MHz", "GHz". Note: The default value is "Hz" .
Return Value	Boolean True if file was successfully written; else False.		

Python Syntax	ExportTouchstone2()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.ExportTouchstone2("D:\folder\nel133.ts", 0, 1, 12, [], True, 23, "GHz")</pre>

VB Syntax	ExportTouchstone2
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set success = oData.ExportTouchstone2 "D:\folder\nel133.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.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Extract 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 will be retained (integers between 1 and <i>n</i>).
Return Value	Network IDispatch.		

Python Syntax	Extract()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Extract(oData1, [3, 2])</pre>

VB Syntax	Extract
VB Example	<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.

UI Access	None.
Parameters	None.
Return Value	Array of doubles.

Python Syntax	GetFrequencies()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") Freqs = oData.GetFrequencies()</pre>

VB Syntax	GetFrequencies
VB Example	<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.

UI Access	None.
Parameters	None.
Return Value	Integer number of frequencies.

Python Syntax	GetFrequencyCount()
Python Example	<pre>oData = oDesktop.GetTool("ndExplorer") numFreqs = oData.GetFrequencyCount()</pre>

VB Syntax	GetFrequencyCount
VB Example	<pre>Set oData = oDesktop.GetTool("ndExplorer") Set numFreqs = oData.GetFrequencyCount</pre>

Warning: The following command is preliminary. Its syntax may change. Be prepared to make changes to legacy scripts.

GetName

Returns the name of the network data.

UI Access	None.
Parameters	None.
Return Value	String name.

Python Syntax	GetName()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") strName = oData.GetName()</pre>

VB Syntax	GetName
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") strName = oNDE.GetName</pre>

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.

UI Access	From the Network Data Explorer tab, select Edit Ports in the NDE ribbon to open the Port properties window.
Parameters	None.
Return Value	Integer number of ports.

Python Syntax	<code>GetPortCount()</code>
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") num = oData.GetPortCount()</pre>

VB Syntax	<code>GetPortCount</code>
VB Example	<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.

The PortNumber is the index of the name port in the original data and is not changed by any other settings applied to the oPostProc, including the Reorder settings. It is permanently linked to the port name and can be used interchangeably with the port name in many of the PostProcSettings script functions.

UI Access	From the Network Data Explorer tab, ensure the Full Port Names check box is activated. Port numbers will be represented in the object portNames.		
Parameters	Name	Type	Description
	<portName>	String	The designation of the port number.
Return Value	Integer.		

Python Syntax	GetPortNumber()		
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") index = oPostProc.GetPortNumber("Input35")</pre>		

VB Syntax	GetPortNumber		
VB Example	<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.

UI Access	None.
Parameters	None.
Return Value	PostProcSettings IDispatch

Python Syntax	GetPostProcSettings()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc = oData.GetPostProcSettings()</pre>

VB Syntax	GetPostProcSettings
VB Example	<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.

GetSolutionVariation

Returns the network data object corresponding to a variation of the solution.

UI Access	None.		
Parameters	Name	Type	Description
	<solutionID>	Integer	ID of the solution (obtained with LoadSolution).
	<variation>	String	The variation key.
Return Value	Network IDispatch.		

Python Syntax	GetSolutionVariation()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData = oNDE.GetSolutionVariation("O, "offset='0.1'")</pre>

VB Syntax	GetSolutionVariation
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") oData = oNDE.GetSolutionVariation "O, "offset='0.1'"</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.

UI Access	None.
Parameters	None.
Return Value	String variation key.

Python Syntax	GetVariation()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") variation = oData.GetVariation()</pre>

VB Syntax	GetVariation
VB Example	<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.

UI Access	None.		
Parameters	Name	Type	Description
	<NetworkData>	Object	The second network data to compare against.
	<matrixType>	Integer	One of the following: • 0 – S-Parameters • 1 – Y-Parameters • 2 – Z-Parameters • 3 – Gamma • 4 – Impedance Note: The default value is 0.
	<comparePortNames>	Boolean	If True , port names must be identical. Note: The default value is True.
	<compareNoise>	Boolean	If True , and both network data have noise data, the comparison will fail unless the noise values also match. Note: The default value is 10.
	<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.
			Note: The default value is 1e-14 .
	<i><absoluteTolerance></i>	Double	If True , then $\text{abs}(\text{value1} - \text{value2}) < \text{absoluteTolerance}$.
			Note: The default value is 0 .
Return Value	Boolean True if the compared network data have the same frequency and matrix values; otherwise, False.		

Python Syntax	HasSameData()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") success = oData.HasSameData(oData2, 0, True, False, 1e-10, 0)</pre>

VB Syntax	HasSameData
VB Example	<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.

UI Access	After analyzing a design, from the Project Manager window, expand the Project Tree and Analysis folders. Then right-click on the completed analysis icon and select Network Data Explorer to open the solution in the Network Data Explorer tab.		
Parameters			
	Name	Type	Description
	<projectName>	String	Full path to the Electronics Desktop file.
	<designName>	String	Name of the design.
	<solutionName>	String	Name of the solution.
Return Value	Integer ID (identifying the solution); < 0 if the solution is not loaded.		

Python Syntax	LoadSolution()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") id = oNDE.LoadSolution("D:\folder\Tee.aedt", "HFSS_Test", "Setup1:Sweep1")</pre>

VB Syntax	LoadSolution
VB Example	<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.

UI Access	From the Network Data Explorer tab, select Open in the NDE ribbon, or select File > Open to open an explorer window. Then navigate to the required S-parameter file.		
Parameters	Name	Type	Description
	<fileName>	String	Full path to the file containing S-parameters.
Return Value	Network IDispatch.		

Python Syntax	Open()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData = oNDE.Open("D:\folder\test.s2p")</pre>

VB Syntax	Open
VB Example	<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.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Rename 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.
	<portNames>	Array of Strings	The new names (one for each port).
Return Value	Network IDispatch.		

Python Syntax	Rename()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Rename(oData1, ["input", "control", "output"])</pre>

VB Syntax	Rename
VB Example	<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.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Renormalize from the drop-down menu.		
Parameters	Name <oData>	Type Object	Description Data from this object is copied to a new network data and the copy is transformed.
	<portImpedances>	Array of Doubles	The new impedance values to be applied to each port.
Return Value	Network IDispatch. Note: Renormalize returns None if there is not an impedance value for each port.		

Python Syntax	Renormalize()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Renormalize(oData1, [3, 2])</pre>

VB Syntax	Renormalize
VB Example	<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 Network Data Explorer tab, select Edit Ports in the NDE ribbon to open the Port properties window. Click+drag the terminal port rows to reorder them. Make changes, as necessary, then click OK .		
Parameters	Name	Type	Description
	<newOrder>	Array of Integers	Must have one entry for each port; all numbers 1 to n 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 Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Reorder 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 in new order (integers between 1 and <i>n</i>).
Return Value	Network IDispatch. Note: Reorder returns None if the array argument does not contain all the terminal port numbers, in any order.		

Python Syntax	Reorder()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Reorder([3, 2, 1])</pre>

VB Syntax	Reorder
VB Example	<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.

Resetting the oPostProcSetting object does not change any data; it only changes the settings object. If you want to change a data object to apply the changed settings, you must follow the Reset function call with oData.SetPostProcSettings (oPostProc)

UI Access	From the Network Data Explorer tab, select Reset postprocessing in the NDE ribbon.

Parameters	None.
Return Value	None.

Python Syntax	Reset()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.Reset()</pre>

VB Syntax	Reset
VB Example	<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.

UI Access	Through the UI, the terminal ports can only be edited individually. From the Network Data Explorer tab, select Edit Ports in the NDE ribbon to open the Port properties window. Make changes, then click OK .		
Parameters	Name	Type	Description
	<impedances>	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.

Return Value	None.

Python Syntax	SetAllPortImpedances()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.SetAllPortImpedances([23, "2+3i", 50]) -or- oData.SetAllPortImpedances(50)</pre>

VB Syntax	SetAllPortImpedances
VB Example	<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.

UI Access	From the Network Data Explorer tab, select Edit Ports in the NDE ribbon to open the Port properties window. Make changes, then click OK .

Parameters	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
Return Value	None.		

Python Syntax	SetPortDeembedDistance()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.SetPortDeembedDistance(2, "20mm")</pre>

VB Syntax	SetPortDeembedDistance
VB Example	<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.

UI Access	From the Network Data Explorer tab, select Edit Ports in the NDE ribbon to open the Port properties window. Make changes, then click OK .

Parameters	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.
Return Value	None.		

Python Syntax	SetPortImpedance()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oPostProc.SetPortImpedance(2, "25+3i")</pre>

VB Syntax	SetPortImpedance
VB Example	<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.

UI Access	None.		
Parameters	Name	Type	Description
	<PostProcSettings>	Object	The settings that will be applied to the data.
Return Value	None.		

Python Syntax	<code>SetPostProcSettings()</code>		
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData.SetPostProcSettings(oPostProc)</pre>		

VB Syntax	<code>SetPostProcSettings</code>		
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") oData.SetPostProcSettings oPostProc</pre>		

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. Note: The default value is 10 .
	<enforceCausality>	Boolean	Casual data is calculated before smoothing. Note: The default value is False .
Return Value	Network IDispatch Note: Network IDispatch is returned only if the Smooth function was successfully able to perform the smoothing operation on the supplied data and meet the maximum 0.025 error tolerance. Otherwise it will return a < null object >. The user may want to try changing other input parameters, for example, <code>enforceCausality</code> .		

Python Syntax	Smooth()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Smooth(oData1, 3, True)</pre>

VB Syntax	Smooth
VB Example	<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.

UI Access	From the Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Stretch 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.
	<factor>	Double	The transmission line lengths are multiplied by this number.
Return Value	Network IDispatch. Note: Stretch returns None if the network data argument does not represent a matrix of transmission lines.		

Python Syntax	Stretch()
Python Example	<pre>oNDE = oDesktop.GetTool("ndExplorer") oData2 = oNDE.Stretch(oData1, 3.1)</pre>

VB Syntax	Stretch
VB Example	<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 Network Data Explorer tab, select Transforms in the NDE ribbon. Then select Terminate 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 <i>n</i>).

	<table><tr><td><termImpedances></td><td>Array of Complex Numbers</td><td>The impedance values used to terminate the specified ports.</td></tr></table>	<termImpedances>	Array of Complex Numbers	The impedance values used to terminate the specified ports.
<termImpedances>	Array of Complex Numbers	The impedance values used to terminate the specified ports.		
Return Value	Network IDispatch. Note: Terminate returns None 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>

VB Syntax	Terminate
VB Example	<pre>Set oNDE = oDesktop.GetTool("ndExplorer") Set oData2 = oNDE.Terminate oData1, Array(2, 3), Array(23, "10+2i")</pre>

This page intentionally
left blank.

20 - CompInstance Script Commands

CompInstance commands should be executed by the **oDesign2** or **oPropHost2** object.

```
Set oDesign2 = CompInstance.GetParentDesign
```

```
Set oPropHost2 = CompInstance.GetPropHost
```

Callback Scripting Using CompInstance 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 `CompInstance`.

You can obtain an interface to a `CompInstance` in a schematic or layout by calling `oEditor.GetCompInstanceFromRefDes(refDes)`. For more information see [Layout Scripting](#) and Schematic Scripting. This interface is also available as a `CompInstance` object in [CompInstance 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:

[ComplInstance Functions](#)

ComplInstance Functions

Following are commands that can be used to manipulate properties from a ComplInstance script.

The topics for this section include:

[GetComponentName](#)

[GetEditor](#)

[GetInstanceID](#)

[GetInstanceName](#)

[GetParentDesign](#)

[GetPropHost](#)

[GetPropServerName](#)

GetComponentName

Returns the name of the component corresponding to this ComplInstance.

UI Access	NA		
Parameters	Name	Type	Description

	None
Return Value	String name of component (e.g. MS_TRL) and stores it in "name"

Python Syntax	GetComponentName()
Python Example	<code>name = CompInstance.GetComponentName()</code>

VB Syntax	GetComponentName()
VB Example	<code>name = CompInstance.GetComponentName()</code>

GetInstanceID [Component Instance]

Use: Returns the instanceID of the CompInstance.

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.

Python Syntax	GetInstanceID()

Python Example	<code>id = CompInstance.GetInstanceID()</code>

GetInstanceName [Component Instance]

Use: Returns the instance name of the component corresponding to this CompInstance.

Command: None

Syntax: GetInstanceName()

Return Value: String

VB Example: `name = CompInstance.GetInstanceName();`

Returns instanceName (e.g. A7) of compInstance and stores it in "name".

Note that the Instance Name is not the same as the RefDes.

GetParentDesign

Use: Returns an interface to the compInstance'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 compInstance.

This interface can be used to call Design functions. See: [Design Object Script Commands](#).

Python Syntax	GetParentDesign()
Python Example	<code>oDesign2 = CompInstance.GetParentDesign()</code>

GetPropHost

Use: Returns an interface to the PropHost of the CompInstance, which gives access to its properties.

Command: None

Syntax: GetPropHost()

VB Example: `Set oPropHost2 = CompInstance.GetPropHost()`;

Returns the interface to the properties of the compInstance.

This interface can be used to call PropHost functions; for more information see [Callback Scripting Using PropHost Object](#).

Python Syntax	GetPropHost()
Python Example	<code>oPropHost2 = CompInstance.GetPropHost()</code>

GetPropServerName

Use: Returns the PropServerName of the Component corresponding to this CompInstance.

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".

Python Syntax	GetPropServerName()
Python Example	<code>name = CompInstance.GetPropServerName()</code>

21 - Simulation Setup Commands

This section presents the commands that are available in the SolveSetups module. These commands should be executed by the **oDesign** object. For example:

```
oDesign.GetModule("SolveSetups")
```

```
oModule.CommandName <args>
```

The following commands are described in this section:

[Add \(Layout Editor\)](#)

AddSimulationSetup [Nexxim]

[AddMeshOperation](#)

[AddSweep](#)

[Analyze](#)

[AnalyzeSweep](#)

[Delete](#)

[DeleteGFD](#)

[DeleteSweep](#)

[DynamicMeshOverlays](#)

[Edit](#)

[EditSweep](#)

[ExportToHFSS](#)

[ExportToQ3D](#)

[GenerateMesh](#)

[GetAllSolutionNames](#)

[GetGFDDData](#)

[GetMeshOperations](#)

[GetSetupData](#)

[GetSetups](#)

[GetSweeps](#)

[LayoutMeshOverlay](#)

[Layout3DMeshOverlay](#)

[ListVariations](#)

[RefreshMeshOverlays](#)

[Rename](#)

RenameSimulationSetup

[RenameSweep](#)

[Validate](#)

Add (Layout Editor)

Adds a new setup.

Note: The following tables display **example** parameters applicable to a Near Field solution setup. Selecting a different setup will also add, remove, or change the applicable parameters.

UI Access	From HFSS 3D Layout , choose Solution Setup and then select from the available solution setups.		
Parameters	Name	Type	Description
	<SetupArray>	Array	An array that contains these parameters in order: • <NAME:SetupName> • <NameArray> • <CustomSetup> • <Position> • <SimSetupType> • <SimSettingsArray> • <SolveSetupType> This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.
	<NAME:SetupName>	String	User-entered name of the new setup.
	<NameArray>	Array	An array that contains these parameters in order: • <NAME:Properties> • <Enable>
	<NAME:Properties>	String	Identifies the contents of the array (i.e., Properties).
	<Enable>	Boolean	True – setup is active False – setup is inactive
	<SimSetupType>	String	Identifies the type of setup (e.g., kSIwaveNearField).
	<SimSettingArray>	Array	<i>Optional.</i> An array that contains these parameters in order: • <NAME:SimulationSettings> • <Enabled> • <UseSISettings> • <UseCustomSettings> • <SISliderPros> • <PISliderPros>

		• <SIWAdvancedArray> • <SIWDCArray> • <SIWDCAdvancedArray> • <SIWSPARAMArray> • <NearFieldSettingArray> • <NAME:SweepDataList> This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.
<NAME:SimulationSettings>	String	Identifies the contents of the array (i.e., SimulationSettings).
<Enabled>	Boolean	True – array settings are active False – array settings are inactive
<UseSISettings>	Boolean	True – uses SI Simulation settings False – uses PI Simulation settings
<UseCustomSettings>	Boolean	True – overrides <UseSISettings> and activates <SIWAdvancedArray> parameters False groups by range
<SISliderPros>	Integer	0 – optimizes simulation for speed 1 – optimizes simulation to balance speed and accuracy 2 – optimizes simulation for accuracy
<PISliderPros>	Integer	0 – optimizes simulation for speed 1 – optimizes simulation to balance speed and accuracy 2 – optimizes simulation for accuracy
<SIWAdvancedArray>	Array	<i>Optional.</i> An array that contains these parameters in order: • <NAME:SIWAdvancedSettings> • <IncludeCoPlaneCoupling> • <IncludeInterPlaneCoupling> • <IncludeSplitPlaneCoupling>

		• <IncludeFringeCoupling> • <IncludeTraceCoupling> • <XtalkThreshold> • <MaxCoupledLines> • <MinVoidArea> • <MinPadAreaToMesh> • SnapLengthThreshold> • <MeshAutoMatic> • <MeshFrequency> • <ReturnCurrentDistribution> • <INcludeVISources> • <IncludeInfGnd> • <InfGndLocation> • <PerformERC> • <IgnoreNonFunctionalPads> This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.
<NAME:SIWAdvancedSettings>	String	Identifies the contents of the array (i.e., SIWAdvancedSettings).
<IncludeCoPlaneCoupling>	Boolean	True – coplane coupling active False – coplane coupling inactive
<IncludeInterPlaneCoupling>	Boolean	True – intra-plane coupling active False – intra-plane coupling inactive
<IncludeSplitPlaneCoupling>	Boolean	True – split-plane coupling active False – split-plane coupling inactive
<IncludeFringeCoupling>	Boolean	True – cavity field coupling active False – cavity field coupling inactive
<IncludeTraceCoupling>	Boolean	True – trace coupling active False – trace coupling inactive

<XtalkThreshold>	Integer	Maximum cross-talk threshold, in decibels.
<MaxCoupledLines>	Integer	The number of allowed coupled lines.
<MinVoidArea>	Integer	Minimum size of voids; voids larger than the minimum are meshed.
<MinPadAreaToMesh>	Integer	Minimum size of pads; pads larger than the minimum are meshed.
<SnapLengthThreshold>	Integer	Minimum distance of vertices; vertices less than the minimum are snapped.
<MeshAutoMatic>	Boolean	True – mesh are automatically refined False – mesh are not automatically refined
<MeshFrequency>	Integer	Specific frequency at which mesh are refined.
<ReturnCurrentDistribution>	Boolean	True – returns current distribution is traced False – returns current distribution is not traced
<IncludeVISources>	Boolean	True – parasitic elements from voltage/current sources are included False – parasitic elements are not included
<IncludeInfGnd>	Boolean	True – infinite ground plane is included False – infinite ground plane is not included
<InfGndLocation>	Integer	Distance below bottom layer in stackup to place infinite ground plane.
<PerformERC>	Boolean	True – performs ERC during simulation False – does not perform ERC during simulation
<IgnoreNonFunctionalPads>	Boolean	True – excludes non-functional pads from resonant mode, frequency sweep, and SYZ simulations False – does not exclude non-functional pads from resonant mode, frequency sweep, and SYZ simulations.
<SIWSPParamSettings>	Array	<i>Optional.</i> An array that contains these parameters in order: • <UseStateSpace> • <Interpolation> • <Extrapolation>

		• <DCBehavior> This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.
<UseStateSpace>	Boolean	True – uses the State Space Model False – uses a Custom Model
<Interpolation>	Integer	0 – sets Interpolation to Point 1 – sets Interpolation to Point 2 – sets Interpolation to Point
<Extrapolation>	Integer	0 – sets Extrapolation to Zero padding 1 – sets Extrapolation to Same as last point 2 – sets Extrapolation to Linear extrapolation from last 2 points 3 – sets Extrapolation to Constant magnitude linear phase extrapolation
<DCBehavior>	Integer	0 – sets DC Behavior to Zero padding 1 – sets DC Behavior to Same as last point 2 – sets DC Behavior to Linear extrapolation from last 2 points 3 – sets DC Behavior to Constant magnitude linear phase extrapolation 4 – sets DC Behavior to Series Capacitor 5 – sets DC Behavior to Leave all signal lines open circuited
<SIWSNearFieldArray>	Array	An array that contains these parameters in order: • <NAME:SIWNearFieldSettings> • <UseExternalSources> • <ExternalSourceFilePath> • <InterpolateSpectrum> • <IncludeInfGnd> • <PosXoffset> • <NegXoffset>

		• <PosYoffset> • <NegYoffset> • <PosZoffset> • <NegZoffset> • <MinAdaptPasses> • <MaxAdaptPasses> • <NewTrianglesPerPass> • <GlobalErrorTol> • <UseUserEdgeLength> • <MeshEdgeLength> • <ExportData> • <ExportDataPath> This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.
<NAME:SIWNearFieldSettings>	String	Identifies the contents of the array (i.e., SIWNearFieldSettings).
<UseExternalSources>	Boolean	True – uses external Excitations definitions False – uses Excitations definitions defined in project
<ExternalSourceFilePath>	String	Directory path to external Excitations definitions file.
<InterpolateSpectrum>	Boolean	True – interpolates spectrum at missing frequency points (external Excitations definitions file only) False – does not interpolate spectrum at missing frequency points
<IncludeInfGnd>	Boolean	True – infinite ground plane is included False – infinite ground plane is not included
<PosXoffset>	Integer	Value determines the +X offset of a cube that encloses the design; the Near Field is evaluated on the surface of this cuboid.
<NegXoffset>	Integer	Value determines the -X offset of a cube that encloses the

			design; the Near Field is evaluated on the surface of this cuboid.
	<PosYoffset>	Integer	Value determines the +Y offset of a cube that encloses the design; the Near Field is evaluated on the surface of this cuboid.
	<NegYoffset>	Integer	Value determines the -Y offset of a cube that encloses the design; the Near Field is evaluated on the surface of this cuboid.
	<PosZoffset>	Integer	Value determines the +Z offset of a cube that encloses the design; the Near Field is evaluated on the surface of this cuboid.
	<NegZoffset>	Integer	Value determines the -Z offset of a cube that encloses the design; the Near Field is evaluated on the surface of this cuboid.
	<MinAdaptPasses>	Integer	Minimum number of passes the adaptive analysis will make, even if convergence criteria are met.
	<MaxAdaptPasses>	Integer	Maximum number of passes the adaptive analysis will make, unless convergence criteria are met.
	<NewTrianglesPerPass>	Integer	Percentage of triangles to add per pass.
	<GlobalErrorTol>	Integer	Maximum relative number of errors to allow between two successive adaptive passes.
	<UseUserEdgeLength>	Boolean	True – uses custom maximum edge length False – automatically determines maximum edge length
	<MeshEdgeLength>	Integer	Maximum edge length.
	<ExportData>	Boolean	True – exports Near Field data False – does not export Near Field data
	<ExportDataPath>	String	Path to directory in which to save Near Field data.
	<NAME:SweepDataList>	String	Identifies the determining factors of the Near Field simulation (i.e., sweeps) (i.e., SweepDataList).
	<SolveSetupType>	String	Identifies the type of setup (i.e., SIwaveNearField).
Return Value	None		

Python Syntax	oModule.Add([<NAME:SIwaveNearField1>, [<NAME:Properties>, <Enable>], <SimSetupType>, [<NAME:SimulationSettings>, <Enabled>, <UseSISettings>, <UseCustomSettings>, <SISliderPos>, <PISliderPos>],

	<pre>[<NAME:SIWAdvancedSettings>, <IncludeCoPlaneCoupling>, <IncludeInterPlaneCoupling>, <IncludeSplitPlaneCoupling>, <IncludeFringeCoupling>, <IncludeTraceCoupling>, <XtalkThreshold>, <MaxCoupledLines>, <MinVoidArea>, <MinPadAreaToMesh>, <SnapLengthThreshold>, <MeshAutoMatic>, <MeshFrequency>, <ReturnCurrentDistribution>, <IncludeVISources>, <IncludeInfGnd>, <InfGndLocation>, <PerformERC>, <IgnoreNonFunctionalPads>], [<NAME:SIWSPParamSettings>, <UseStateSpace>, <Inter- polation>, <Extrapolation>, <DCBehavior>], [<NAME:SIWNearFieldSettings>, <UseExternalSources>, <Extern- alSourceFilePath>, <InterpolateSpectrum>, <IncludeInfGnd>, <PosXoffset>, <NegXoffset>, <PosYoffset>, <NegYoffset>, <PosZoffset>, <NegZoffset>, <MinAdaptPasses>, <MaxAdaptPasses>, <NewTri- anglesPerPass>, <GlobalErrorTol>, <UseUserEdgeLength>, <MeshEdgeLength>, <ExportData>, <ExportDataPath>]], [<NAME:SweepDataList>], <SolveSetupType>]]</pre>
Python Example	<pre>oModule.Add(["NAME:SIwaveNearField1", ["NAME:Properties", "Enable:=" , "true"], "SimSetupType:=" , "kSIwaveNearField", ["NAME:SimulationSettings", "Enabled:=" , True, "UseSISettings:=" , False, "UseCustomSettings:=" , False,</pre>

```

        "SISliderPos:="          , 2,
        "PISliderPos:="          , 0,

    [
        "NAME:SIWAdvancedSettings",
        "IncludeCoPlaneCoupling:=", False,
        "IncludeInterPlaneCoupling:=", False,
        "IncludeSplitPlaneCoupling:=", False,
        "IncludeFringeCoupling:=", False,
        "IncludeTraceCoupling:=", False,
        "XtalkThreshold:="        , "-34",
        "MaxCoupledLines:="        , 12,
        "MinVoidArea:="            , "2mm2",
        "MinPadAreaToMesh:="        , "1mm2",
        "MinPlaneAreaToMesh:="      , "6.25e-6mm2",
        "SnapLengthThreshold:="     , "2.5um",
        "MeshAutoMatic:="          , True,
        "MeshFrequency:="           , "4GHz",
        "ReturnCurrentDistribution:=", False,
        "IncludeVISources:="        , False,
        "IncludeInfGnd:="           , False,
        "InfGndLocation:="          , "0mm",
    ]

```

```
"PerformERC:="          , False,
"IgnoreNonFunctionalPads:=", True
],
[
  "NAME:SIWSParmSettings",
  "UseStateSpace:="      , True,
  "Interpolation:="      , 1,
  "Extrapolation:="      , 3,
  "DCBehavior:="         , 3
],
[
  "NAME:SIWNearFieldSettings",
  "UseExternalSources:="  , False,
  "ExternalSourceFilePath:=", "",
  "InterpolateSpectrum:=" , True,
  "IncludeInfGnd:="       , False,
  "PosXoffset:="          , "1mm",
  "NegXoffset:="           , "1mm",
  "PosYoffset:="           , "1mm",
  "NegYoffset:="           , "1mm",
```

```

        "PosZoffset:="          , "1mm",
        "NegZoffset:="          , "1mm",
        "MinAdaptPasses:="      , 1,
        "MaxAdaptPasses:="      , 10,
        "NewTrianglesPerPass:=" , 20,
        "GlobalErrorTol:="      , 3,
        "UseUserEdgeLength:="    , False,
        "MeshEdgeLength:="      , "1mm",
        "ExportData:="          , False,
        "ExportDataPath:="      , ""
    ]
],
[
    "NAME:SweepDataList"
],
"SolveSetupType:="          , "SIwaveNearField"
])

```

```

oModule.Add (<NAME:SIwaveNearField1>, (<NAME:Properties>, <Enable>), <SimSetupType>, (<NAME:SimulationSettings>, <Enabled>, <UseS
terPlaneCoupling>, <IncludeSplitPlaneCoupling>, <IncludeFringeCoupling>, <IncludeTraceCoupling>, <XtalkThreshold>, <MaxCoupledLines>, <I
<IncludeVISources>, <IncludeInfGnd>, <InfGndLocation>, <PerformERC>, <IgnoreNonFunctionalPads>), (<NAME:SIWSParmSettings>, <UseS
alSourceFilePath>, <InterpolateSpectrum>, <IncludeInfGnd>, <PosXoffset>, <NegXoffset>, <PosYoffset>, <NegYoffset>, <PosZoffset>, <NegZoff

```

<ExportData>, <ExportDataPath>)), (<NAME:SweepDataList>, <SolveSetupType>)

V-B-E-X-a-m-p-l-e
oModule.Add Array("NAME:SIwaveNearField1", Array("NAME:Properties", "Enable:=", "true"), "SimSetupType:=",
"UseCustomSettings:=", False, "SISliderPos:=", 2, "PISliderPos:=", 0, Array("NAME:SIWAdvancedSettings", "U
False, "IncludeFringeCoupling:=", False, "IncludeTraceCoupling:=", False, "XtalkThreshold:=", "-34", "Max
"6.25e-6mm2", "SnapLengthThreshold:=", "2.5um", "MeshAutoMatic:=", True, "MeshFrequency:=", "4GHz", "Retu
tion:=", "0mm", "PerformERC:=", False, "IgnoreNonFunctionalPads:=", True), Array("NAME:SIWDCSettings", "U
"DCSliderPos:=", 1), Array("NAME:SIWDCAdvancedSettings", "DcMinPlaneAreaToMesh:=", "0.25mm2", "DcMinVoidA
"MaxNumPasses:=", 5, "MinNumPasses:=", 1, "PercentLocalRefinement:=", 20, "EnergyError:=", 2, "MeshBws:=",
aSides:=", 8), Array("NAME:SIWSPParamSettings", "UseStateSpace:=", True, "Interpolation:=", 1, "Extrapolat
alSourceFilePath:=", "", "InterpolateSpectrum:=", True, "IncludeInfGnd:=", False, "PosXoffset:=", "1mm",
"NegZoffset:=", "1mm", "MinAdaptPasses:=", 1, "MaxAdaptPasses:=", 10, "NewTrianglesPerPass:=", 20, "Globa
"ExportDataPath:=", "")), Array("NAME:SweepDataList", "SolveSetupType:=", "SIwaveNearField"))

AddMeshOperation

Adds a mesh operation to a specified solution.

UI Access	From the Project Manager , right-click a solution and select Assign Mesh Operation > [Any Submenu Item].		
Parameters	Name	Type	Description
	<name>	String	The name of the solution for which you want to apply a mesh operation.
	<MeshArray>	Array	Structured array: Array("NAME:<mesh operation name>", "RefineInside:=" , <boolean True for inside; False for on surface>,

		<pre> "Type:=" , <string "LengthBased" or "SkinDepthLengthBased">, "Enabled:=" , <boolean True to enable operation; False to disable>, Array("NAME:Assignment", <MeshEntityInfo>, <MeshEntityInfo>, <MeshEntityInfo>,...) "Region:=" , <string name of region, if selecting one>, "RestrictElem:=" , <boolean True to restrict number of mesh elements>, "NumMaxElem:=" , <string containing maximum number of mesh elements>, "RestrictLength:=" , <boolean True to restrict length of mesh elements>, "MaxLength:=" , <string containing maximum length of mesh element> </pre>
	<MeshEntityInfo>	Array Structured array. Add one MeshEntityInfo array for each entity selected. <pre> Array("NAME:MeshEntityInfo", "IsFcSel:=" , <boolean>, "EntID:=" , <integer>, "FcIDs:=" , <array>, ["NAME:MeshBody", </pre>

			<pre>"Id:=" , <integer>, "Nam:=" , <string>, "Layer:=" , <string layer name>, "Sheet:=" , <boolean>, "Net:=" , <string net name or "<no-net>">, "OrigNet:=" , <string net name or "<no-net>">], "BBox:=" , <array>]</pre>
Return Value	None.		

Python Syntax	AddMeshOperation(<SolutionName>, <MeshArray>)
Python Example	<pre>oModule.AddMeshOperation('HFSS Solution 1', ["NAME:Length 2", "RefineInside:=" , False, "Type:=" , "LengthBased", "Enabled:=" , True, [</pre>

```

"NAME:Assignment",
[
  "NAME:MeshEntityInfo",
  "IsFcSel:="      , False,
  "EntID:="        , -1,
  "FcIDs:="        , [],
  [
    "NAME:MeshBody",
    "Id:="          , -1,
    "Nam:="         , "",
    "Layer:="       , "UNNAMED_ 0",
    "Sheet:="       , True,
    "Net:="         , "<no-net>",
    "OrigNet:="     , "<no-net>"
  ],
  "BBox:="         , []
],
[
  "NAME:MeshEntityInfo",
  "IsFcSel:="      , False,
  "EntID:="        , -1,

```

```
"FcIDs:="      , [],  
  [  
    "NAME:MeshBody",  
    "Id:="      , -1,  
    "Nam:="     , "",  
    "Layer:="   , "UNNAMED_ 0",  
    "Sheet:="   , True,  
    "Net:="     , "PCIE_VSS_C",  
    "OrigNet:=" , "PCIE_VSS_C"  
  ],  
  "BBox:="     , []  
],  
],  
"Region:="     , "EXTENTS:rect_1076",  
"RestrictElem:=" , False,  
"NumMaxElem:="  , "1000",  
"RestrictLength:=", True,  
"MaxLength:="   , "0.2mm"  
]  
)
```


VB Syntax	AddMeshOperation <SolutionName> <MeshArray>
VB Example	<pre>oModule.AddMeshOperation "Setup2", Array("NAME:Length 1", "RefineInside:=", false, "Type:=", _ "LengthBased", "Enabled:=", true, Array("NAME:Assignment", Array("NAME:MeshEntityInfo", "IsFcSel:=", _ false, "EntID:=", -1, "FcIDs:=", Array(), Array("NAME:MeshBody", "Id:=", -1, "Nam:=", _ "", "Layer:=", "Top_L1", "Net:=", "neg", "OrigNet:=", "neg"), "BBox:=", Array()), Array ("NAME:MeshEntityInfo", "IsFcSel:=", _ false, "EntID:=", -1, "FcIDs:=", Array(), Array("NAME:MeshBody", "Id:=", -1, "Nam:=", _ "", "Layer:=", "Top_L1", "Net:=", "pos", "OrigNet:=", "pos"), "BBox:=", Array()), "Region:=", _ "user1:rect_192", "RestrictElem:=", false, "NumMaxElem:=", "1000", "RestrictLength:=", _ true, "MaxLength:=", "0.2mm")</pre>

AddSweep (Layout Editor)

Adds a sweep to a setup.

UI Access	From the Project Manager window, right-click a setup and select Add Frequency Sweep .		
Parameters	Name	Type	Description

	<SetupName>	String	Name of the setup.
	<Parameters>	Array	A structured array containing the name and parameters for the new frequency sweep.
Return Value	None		

Python Syntax	oModule.AddSweep(<SetupName>, [<Parameters>])
Python Example	<pre>oModule.AddSweep("Q3D1", ["NAME:Sweep1", ["NAME:Properties", "Enable:=" , "true"], ["NAME:Sweeps", "Variable:=" , "Sweep1", "Data:=" , "LIN 10MHz 100MHz 10MHz", "OffsetF1:=" , False, "Synchronize:=" , 0],],</pre>

```

"GenerateSurfaceCurrent:=", False,
"SaveRadFieldsOnly:=" , False,
"ZoSelected:=" , False,
"SAbsError:=" , 0.005,
"ZoPercentError:=" , 1,
"GenerateStateSpace:=" , False,
"EnforcePassivity:=" , False,
"PassivityTolerance:=" , 0.0001,
"UseQ3DForDC:=" , False,
"ResimulateDC:=" , False,
"MaxSolutions:=" , 250,
"InterpUseSMatrix:=" , True,
"InterpUsePortImpedance:=", True,
"InterpUsePropConst:=" , True,
"InterpUseFullBasis:=" , True,
"AdvDCExtrapolation:=" , False,
"MinSolvedFreq:=" , "0.01GHz",
"AutoSMatOnlySolve:=" , True,
"MinFreqSMatrixOnlySolve:=", "1MHz",
"CustomFrequencyString:=", "",
"AllEntries:=" , False,

```

	<pre> "AllDiagEntries:=" , False, "AllOffDiagEntries:=" , False, "MagMinThreshold:=" , 0.01, "InterpMinSubranges:=" , 1, "InterpMinSolutions:=" , 0, "FreqSweepType:=" , "kDiscrete"])</pre>

VB Syntax	oModule.AddSweep <SetupName>, (<Parameters>)
VB Example	<pre> oModule.AddSweep "Q3D1", Array("NAME:Sweep1", Array("NAME:Properties", "Enable:=", "true"), Array("NAME:Sweeps", "Variable:=", "Sweep1", "Data:=", "LIN 10MHz 100MHz 10MHz", "OffsetF1:=", False, "Syn- chronize:=", 0), "GenerateSurfaceCurrent:=", False, "SaveRadFieldsOnly:=", False, "ZoSelected:=", False, "SAbsError:=", 0.005, "ZoPer- centError:=", 1, "GenerateStateSpace:=", False, "EnforcePassivity:=", False, "PassivityTolerance:=", 0.0001, "UseQ3DForDC:=", False, "Res- imulateDC:=", False, "MaxSolutions:=", 250, "InterpUseSMatrix:=", True, "InterpUsePortImpedance:=", True, "InterpUsePropConst:=", True, "InterpUseFullBasis:=", True, "AdvDCExtrapolation:=", False, "MinSolvedFreq:=", "0.01GHz", "AutoSMatOnlySolve:=", True, "MinFreqSMatrixOnlySolve:=", "1MHz", "CustomFrequencyString:=", "", "AllEntries:=", False,</pre>

"AllDiagEntries:=", False, "AllOffDiagEntries:=", False, "MagMinThreshold:=", 0.01, "InterpMinSubranges:=", 1, "InterpMinSolutions:=", 0, "FreqSweepType:=", "kDiscrete")

Analyze (Layout Editor)

Use: Simulates the given setup

Syntax: Analyze(String setup1) // setup name

VB Example: oModule.Analyze <setup1>

AnalyzeSweep (Layout Editor)

Use: Simulates the given sweep

Syntax: AnalyzeSweep(String setup1) // setup name

STRING sweep1) // sweep name

VB Example: oModule.AnalyzeSweep <setup1> <sweep1>

Delete (Layout Editor)

Deletes a setup.

UI Access	From the Project Manager window, right-click a setup and select Delete .		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup.
Return Value	None		

Python Syntax	oModule.Delete(<SetupName>)
Python Example	oModule.Delete("Q3D2")

VB Syntax	<code>oModule.Delete <SetupName></code>
VB Example	<code>oModule.Delete "Q3D2"</code>

DeleteGFD (Setup HFSS 3D Layout)

Deletes the specified GFD (i.e., Green's Function Data).

UI Access	None.		
Parameters	Name	Type	Description
	<GFDName>	String	Name of Green's Function Data file in the active design.
Return Value	None		

Python Syntax	<code>DeleteGFD(<GFDName>)</code>
Python Example	<code>oEditor.DeleteGFD("GDS_File.gds")</code>

VB Syntax	<code>DeleteGFD<GFDName></code>
VB Example	<code>oEditor.DeleteGFD "GDS_File.gds"</code>

DeleteSweep (Layout Editor)

Deletes a sweep.

UI Access	From the Project Manager window, right-click a setup and select Delete .		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup.
	<SweepName>	String	Name of the sweep.
Return Value	None		

Python Syntax	oModule.DeleteSweep(<SetupName>, <SweepName>)		
Python Example	oModule.DeleteSweep("Q3D2", "Sweep2")		

VB Syntax	oModule.DeleteSweep <SetupName>, <SweepName>		
VB Example	oModule.DeleteSweep "Q3D2", "Sweep2"		

=

DynamicMeshOverlays (Layout Editor)

Use: Turns on dynamic mesh overlays

Syntax: DynamicMeshOverlays(STRING setup) // setup name

VB Example: oModule. DynamicMeshOverlay <setup>

Edit (Layout Editor)

Edits a setup.

Note: The following tables display **example** parameters applicable to a Near Field solution setup. Selecting a different setup will also add, remove, or change the applicable parameters.

UI Access	From the Project Manager window, right-click a setup and select Properties .		
Parameters	Name	Type	Description
	<SetupArray>	Array	An array that contains these parameters in order: • <NAME:SetupName> • <NameArray> • <CustomSetup> • <Position> • <SimSetupType> • <SimSettingsArray> • <SolveSetupType> This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.
	<NAME:SetupName>	String	User-entered name of the setup.
	<NameArray>	Array	An array that contains these parameters in order: • <NAME:Properties> • <Enable>
	<NAME:Properties>	String	Identifies the contents of the array (i.e., Properties).
	<Enable>	Boolean	True – setup is active False – setup is inactive
	<SimSetupType>	String	Identifies the type of setup (e.g., kSIwaveNearField).
	<SimSettingArray>	Array	<i>Optional.</i> An array that contains these parameters in order: • <NAME:SimulationSettings>

		• <Enabled> • <UseSISettings> • <UseCustomSettings> • <SISliderPros> • <PISliderPros> • <SIWAdvancedArray> • <SIWDCArray> • <SIWDCAdvancedArray> • <SIWSParamArray> • <NearFieldSettingArray> • <NAME:SweepDataList> This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.
<NAME:SimulationSettings>	String	Identifies the contents of the array (i.e., SimulationSettings).
<Enabled>	Boolean	True – array settings are active False – array settings are inactive
<UseSISettings>	Boolean	True – uses SI Simulation settings False – uses PI Simulation settings
<UseCustomSettings>	Boolean	True – overrides <UseSISettings> and activates <SIWAdvancedArray> parameters False groups by range
<SISliderPros>	Integer	0 – optimizes simulation for speed 1 – optimizes simulation to balance speed and accuracy 2 – optimizes simulation for accuracy
<PISliderPros>	Integer	0 – optimizes simulation for speed 1 – optimizes simulation to balance speed and accuracy 2 – optimizes simulation for accuracy
<SIWAdvancedArray>	Array	<i>Optional.</i> An array that contains these parameters in order: • <NAME:SIWAdvancedSettings>

			• <IncludeCoPlaneCoupling> • <IncludeInterPlaneCoupling> • <IncludeSplitPlaneCoupling> • <IncludeFringeCoupling> • <IncludeTraceCoupling> • <XtalkThreshold> • <MaxCoupledLines> • <MinVoidArea> • <MinPadAreaToMesh> • <SnapLengthThreshold> • <MeshAutoMatic> • <MeshFrequency> • <ReturnCurrentDistribution> • <INcludeVISources> • <IncludeInfGnd> • <InfGndLocation> • <PerformERC> • <IgnoreNonFunctionalPads> This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.
	<NAME:SIWAdvancedSettings>	String	Identifies the contents of the array (i.e., SIWAdvancedSettings).
	<IncludeCoPlaneCoupling>	Boolean	True – coplane coupling active False – coplane coupling inactive
	<IncludeInterPlaneCoupling>	Boolean	True – intra-plane coupling active False – intra-plane coupling inactive

<IncludeSplitPlaneCoupling>	Boolean	True – split-plane coupling active False – split-plane coupling inactive
<IncludeFringeCoupling>	Boolean	True – cavity field coupling active False – cavity field coupling inactive
<IncludeTraceCoupling>	Boolean	True – trace coupling active False – trace coupling inactive
<XtalkThreshold>	Integer	Maximum cross-talk threshold, in decibels.
<MaxCoupledLines>	Integer	The number of allowed coupled lines.
<MinVoidArea>	Integer	Minimum size of voids; voids larger than the minimum are meshed.
<MinPadAreaToMesh>	Integer	Minimum size of pads; pads larger than the minimum are meshed.
<SnapLengthThreshold>	Integer	Minimum distance of vertices; vertices less than the minimum are snapped.
<MeshAutoMatic>	Boolean	True – mesh are automatically refined False – mesh are not automatically refined
<MeshFrequency>	Integer	Specific frequency at which mesh are refined.
<ReturnCurrentDistribution>	Boolean	True – returns current distribution is traced False – returns current distribution is not traced
<IncludeVISources>	Boolean	True – parasitic elements from voltage/current sources are included False – parasitic elements are not included
<IncludeInfGnd>	Boolean	True – infinite ground plane is included False – infinite ground plane is not included
<InfGndLocation>	Integer	Distance below bottom layer in stackup to place infinite ground plane.
<PerformERC>	Boolean	True – performs ERC during simulation False – does not perform ERC during simulation
<IgnoreNonFunctionalPads>	Boolean	True – excludes non-functional pads from resonant mode, frequency sweep, and SYZ simulations False – does not exclude non-functional pads from resonant mode, frequency sweep, and SYZ simulations.
<SIWSParmSettings>	Array	<i>Optional.</i> An array that contains these parameters in order:

		• <UseStateSpace> • <Interpolation> • <Extrapolation> • <DCBehavior> This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.
<UseStateSpace>	Boolean	True – uses the State Space Model False – uses a Custom Model
<Interpolation>	Integer	0 – sets Interpolation to Point 1 – sets Interpolation to Point 2 – sets Interpolation to Point
<Extrapolation>	Integer	0 – sets Extrapolation to Zero padding 1 – sets Extrapolation to Same as last point 2 – sets Extrapolation to Linear extrapolation from last 2 points 3 – sets Extrapolation to Constant magnitude linear phase extrapolation
<DCBehavior>	Integer	0 – sets DC Behavior to Zero padding 1 – sets DC Behavior to Same as last point 2 – sets DC Behavior to Linear extrapolation from last 2 points 3 – sets DC Behavior to Constant magnitude linear phase extrapolation 4 – sets DC Behavior to Series Capacitor 5 – sets DC Behavior to Leave all signal lines open circuited
<SIWSNearFieldArray>	Array	An array that contains these parameters in order: • <NAME:SIWNearFieldSettings> • <UseExternalSources> • <ExternalSourceFilePath>

		• <InterpolateSpectrum> • <IncludeInfGnd> • <PosXoffset> • <NegXoffset> • <PosYoffset> • <NegYoffset> • <PosZoffset> • <NegZoffset> • <MinAdaptPasses> • <MaxAdaptPasses> • <NewTrianglesPerPass> • <GlobalErrorTol> • <UseUserEdgeLength> • <MeshEdgeLength> • <ExportData> • <ExportDataPath> This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.
<NAME:SIWNearFieldSettings>	String	Identifies the contents of the array (i.e., SIWNearFieldSettings).
<UseExternalSources>	Boolean	True – uses external Excitations definitions False – uses Excitations definitions defined in project
<ExternalSourceFilePath>	String	Directory path to external Excitations definitions file.
<InterpolateSpectrum>	Boolean	True – interpolates spectrum at missing frequency points (external Excitations definitions file only) False – does not interpolate spectrum at missing frequency points
<IncludeInfGnd>	Boolean	True – infinite ground plane is included False – infinite ground plane is not included

	<PosXoffset>	Integer	Value determines the +X offset of a cube that encloses the design; the Near Field is evaluated on the surface of this cuboid.
	<NegXoffset>	Integer	Value determines the -X offset of a cube that encloses the design; the Near Field is evaluated on the surface of this cuboid.
	<PosYoffset>	Integer	Value determines the +Y offset of a cube that encloses the design; the Near Field is evaluated on the surface of this cuboid.
	<NegYoffset>	Integer	Value determines the -Y offset of a cube that encloses the design; the Near Field is evaluated on the surface of this cuboid.
	<PosZoffset>	Integer	Value determines the +Z offset of a cube that encloses the design; the Near Field is evaluated on the surface of this cuboid.
	<NegZoffset>	Integer	Value determines the -Z offset of a cube that encloses the design; the Near Field is evaluated on the surface of this cuboid.
	<MinAdaptPasses>	Integer	Minimum number of passes the adaptive analysis will make, even if convergence criteria are met.
	<MaxAdaptPasses>	Integer	Maximum number of passes the adaptive analysis will make, unless convergence criteria are met.
	<NewTrianglesPerPass>	Integer	Percentage of triangles to add per pass.
	<GlobalErrorTol>	Integer	Maximum relative number of errors to allow between two successive adaptive passes.
	<UseUserEdgeLength>	Boolean	True – uses custom maximum edge length False – automatically determines maximum edge length
	<MeshEdgeLength>	Integer	Maximum edge length.
	<ExportData>	Boolean	True – exports Near Field data False – does not export Near Field data
	<ExportDataPath>	String	Path to directory in which to save Near Field data.
	<NAME:SweepDataList>	String	Identifies the determining factors of the Near Field simulation (i.e., sweeps) (i.e., SweepDataList).
	<SolveSetupType>	String	Identifies the type of setup (i.e., SIwaveNearField).
Return Value	None		

Python Syntax	<code>oModule.Edit([<NAME:SIwaveNearField1>, [<NAME:Properties>, <Enable>], <SimSetupType>, [<NAME:SimulationSettings>, <Enabled>, <UseSISettings>, <UseCustomSettings>, <SISliderPos>, <PISliderPos>], [<NAME:SIWAdvancedSettings>, <IncludeCoPlaneCoupling>, <IncludeInterPlaneCoupling>, <IncludeSplitPlaneCoupling>, <IncludeFringeCoupling>, <IncludeTraceCoupling>, <XtalkThreshold>, <MaxCoupledLines>, <MinVoidArea>, <MinPadAreaToMesh>, <SnapLengthThreshold>, <MeshAutoMatic>, <MeshFrequency>, <ReturnCurrentDistribution>, <IncludeVISources>, <IncludeInfGnd>, <InfGndLocation>, <PerformERC>, <IgnoreNonFunctionalPads>], [<NAME:SIWSParmSettings>, <UseStateSpace>, <Inter- polation>, <Extrapolation>, <DCBehavior>], [<NAME:SIWNearFieldSettings>, <UseExternalSources>, <Extern- alSourceFilePath>, <InterpolateSpectrum>, <IncludeInfGnd>, <PosXoffset>, <NegXoffset>, <PosYoffset>, <NegYoffset>, <PosZoffset>, <NegZoffset>, <MinAdaptPasses>, <MaxAdaptPasses>, <NewTri- anglesPerPass>, <GlobalErrorTol>, <UseUserEdgeLength>, <MeshEdgeLength>, <ExportData>, <ExportDataPath>]], [<NAME:SweepDataList>], <SolveSetupType>])</code>
Python Example	<pre>oModule.Edit(["NAME:SIwaveNearField1", ["NAME:Properties", "Enable:=" , "true"], "SimSetupType:=" , "kSIwaveNearField", ["NAME:SimulationSettings", "Enabled:=" , True, "UseSISettings:=" , False,</pre>

```
"UseCustomSettings:="      , False,
"SISSliderPos:="           , 2,
"PISSliderPos:="           , 0,
[
  "NAME:SIWAdvancedSettings",
  "IncludeCoPlaneCoupling:=", False,
  "IncludeInterPlaneCoupling:=", False,
  "IncludeSplitPlaneCoupling:=", False,
  "IncludeFringeCoupling:=", False,
  "IncludeTraceCoupling:=", False,
  "XtalkThreshold:="        , "-34",
  "MaxCoupledLines:="       , 12,
  "MinVoidArea:="           , "2mm2",
  "MinPadAreaToMesh:="      , "1mm2",
  "MinPlaneAreaToMesh:="    , "6.25e-6mm2",
  "SnapLengthThreshold:="   , "2.5um",
  "MeshAutoMatic:="         , True,
  "MeshFrequency:="         , "4GHz",
  "ReturnCurrentDistribution:=", False,
  "IncludeVISources:="      , False,
```

```

        "IncludeInfGnd:="          , False,
        "InfGndLocation:="        , "0mm",
        "PerformERC:="            , False,
        "IgnoreNonFunctionalPads:=", True
    ],
    [
        "NAME:SIWSParmSettings",
        "UseStateSpace:="          , True,
        "Interpolation:="          , 1,
        "Extrapolation:="          , 3,
        "DCBehavior:="             , 3
    ],
    [
        "NAME:SIWNearFieldSettings",
        "UseExternalSources:="      , False,
        "ExternalSourceFilePath:=", "",
        "InterpolateSpectrum:="     , True,
        "IncludeInfGnd:="          , False,
        "PosXoffset:="              , "1mm",
        "NegXoffset:="              , "1mm",
        "PosYoffset:="              , "1mm",
    ]

```

```
"NegYoffset:="      , "1mm",
"PosZoffset:="      , "1mm",
"NegZoffset:="      , "1mm",
"MinAdaptPasses:="  , 1,
"MaxAdaptPasses:="  , 10,
"NewTrianglesPerPass:=" , 20,
"GlobalErrorTol:="  , 3,
"UseUserEdgeLength:=" , False,
"MeshEdgeLength:="  , "1mm",
"ExportData:="      , False,
"ExportDataPath:="   , ""
]
],
[
  "NAME:SweepDataList"
],
"SolveSetupType:="  , "SIwaveNearField"
])
```

```
oModule.Edit(<NAME:SIwaveNearField1>, (<NAME:Properties>, <Enable>), <SimSetupType>, (<NAME:SimulationSettings>, <Enabled>, <UseSplitPlaneCoupling>, <IncludeSplitPlaneCoupling>, <IncludeFringeCoupling>, <IncludeTraceCoupling>, <XtalkThreshold>, <MaxCoupledLines>, <IncludeVISources>, <IncludeInfGnd>, <InfGndLocation>, <PerformERC>, <IgnoreNonFunctionalPads>), (<NAME:SIWSPParamSettings>, <UseStateSpace>, <InterpolateSpectrum>, <IncludeInfGnd>, <PosXoffset>, <NegXoffset>, <PosYoffset>, <NegYoffset>, <PosZoffset>, <NegZoffset>, <ExportData>, <ExportDataPath>)), (<NAME:SweepDataList>, <SolveSetupType>)
```

```
V-
B
E-
x-
a-
m-
p-
l-
e
oModule.Edit Array("NAME:SIwaveNearField1", Array("NAME:Properties", "Enable:=", "true"), "SimSetupType:=",
"UseCustomSettings:=", False, "SISliderPos:=", 2, "PISliderPos:=", 0, Array("NAME:SIWAdvancedSettings", "
False, "IncludeFringeCoupling:=", False, "IncludeTraceCoupling:=", False, "XtalkThreshold:=", "-34", "Max
"6.25e-6mm2", "SnapLengthThreshold:=", "2.5um", "MeshAutoMatic:=", True, "MeshFrequency:=", "4GHz", "Retu
tion:=", "0mm", "PerformERC:=", False, "IgnoreNonFunctionalPads:=", True), Array("NAME:SIWDCSettings", "U
"DCSliderPos:=", 1), Array("NAME:SIWDCAdvancedSettings", "DcMinPlaneAreaToMesh:=", "0.25mm2", "DcMinVoidA
"MaxNumPasses:=", 5, "MinNumPasses:=", 1, "PercentLocalRefinement:=", 20, "EnergyError:=", 2, "MeshBws:=",
aSides:=", 8), Array("NAME:SIWSPParamSettings", "UseStateSpace:=", True, "Interpolation:=", 1, "Extrapolat
alSourceFilePath:=", "", "InterpolateSpectrum:=", True, "IncludeInfGnd:=", False, "PosXoffset:=", "1mm",
"NegZoffset:=", "1mm", "MinAdaptPasses:=", 1, "MaxAdaptPasses:=", 10, "NewTrianglesPerPass:=", 20, "Globa
"ExportDataPath:=", "")), Array("NAME:SweepDataList", "SolveSetupType:=", "SIwaveNearField"))
```

EditSweep (Layout Editor)

Edits a sweep.

UI Access	From the Project Manager window, right-click a setup and select Add Frequency Sweep .		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup.
	<Parameters>	Array	A structured array containing the name and parameters for the frequency sweep.

Return Value	None

Python Syntax	<code>oModule.EditSweep(<SetupName>, [<Parameters>])</code>
Python Example	<pre>oModule.EditSweep("Q3D1", ["NAME:Sweep1", ["NAME:Properties", "Enable:=" , "true"], ["NAME:Sweeps", "Variable:=" , "Sweep1", "Data:=" , "LIN 10MHz 100MHz 10MHz", "OffsetF1:=" , False, "Synchronize:=" , 0], "GenerateSurfaceCurrent:=", False, "SaveRadFieldsOnly:=" , False,</pre>

```
"ZoSelected:=" , False,  
"SAbsError:=" , 0.005,  
"ZoPercentError:=" , 1,  
"GenerateStateSpace:=" , False,  
"EnforcePassivity:=" , False,  
"PassivityTolerance:=" , 0.0001,  
"UseQ3DForDC:=" , False,  
"ResimulateDC:=" , False,  
"MaxSolutions:=" , 250,  
"InterpUseSMatrix:=" , True,  
"InterpUsePortImpedance:=" , True,  
"InterpUsePropConst:=" , True,  
"InterpUseFullBasis:=" , True,  
"AdvDCExtrapolation:=" , False,  
"MinSolvedFreq:=" , "0.01GHz",  
"AutoSMatOnlySolve:=" , True,  
"MinFreqSMatrixOnlySolve:=" , "1MHz",  
"CustomFrequencyString:=" , "",  
"AllEntries:=" , False,  
"AllDiagEntries:=" , False,  
"AllOffDiagEntries:=" , False,
```


	<pre> "MagMinThreshold:=" , 0.01, "InterpMinSubranges:=" , 1, "InterpMinSolutions:=" , 0, "FreqSweepType:=" , "kDiscrete"]) </pre>

VB Syntax	oModule.EditSweep <SetupName>, (<Parameters>)
VB Example	<pre> oModule.EditSweep "Q3D1", Array("NAME:Sweep1", Array("NAME:Properties", "Enable:=", "true"), Array("NAME:Sweeps", "Variable:=", "Sweep1", "Data:=", "LIN 10MHz 100MHz 10MHz", "OffsetF1:=", False, "Syn- chronize:=", 0), "GenerateSurfaceCurrent:=", False, "SaveRadFieldsOnly:=", False, "ZoSelected:=", False, "SAbsError:=", 0.005, "ZoPer- centError:=", 1, "GenerateStateSpace:=", False, "EnforcePassivity:=", False, "PassivityTolerance:=", 0.0001, "UseQ3DForDC:=", False, "Res- imulateDC:=", False, "MaxSolutions:=", 250, "InterpUseSMatrix:=", True, "InterpUsePortImpedance:=", True, "InterpUsePropConst:=", True, "InterpUseFullBasis:=", True, "AdvDCExtrapolation:=", False, "MinSolvedFreq:=", "0.01GHz", "AutoSMatOnlySolve:=", True, "MinFreqSMatrixOnlySolve:=", "1MHz", "CustomFrequencyString:=", "", "AllEntries:=", False, "AllDiagEntries:=", False, "AllOffDiagEntries:=", False, "MagMinThreshold:=", 0.01, "Inter- pMinSubranges:=", 1, "InterpMinSolutions:=", 0, "FreqSweepType:=", "kDiscrete") </pre>

ExportToHFSS (Layout Editor)

Use: Exports an HFSS 3D Layout design to a new project with an equivalent HFSS 3D design. This command is only valid on HFSS Setup types.

Syntax: ExportToHfss(<setupName>, <pathToExportFile>)

VB Example: oSetup.ExportToHfss("HFSS Setup 1", "C:\Temp\myexport.aedt")

ExportToQ3D (Layout Editor)

Use: Exports an HFSS 3D Layout design to a new project containing an equivalent Q3D design. This command is only valid on HFSS Setup types.

Syntax: ExportToQ3d(<setupName>, <pathToExportFile>)

VB Example: oSetup.ExportToQ3d("HFSS Setup 1", "C:\Temp\myexport.aedt")

GetAllSolutionNames (Layout Editor)

Use: Returns a list of Ensemble solution names

Syntax: GetAllSolutionNames

Return Value: Array of Strings

Parameters: None

VB Example: Dim setupArr

setupArr = oModule.GetAllSolutionNames

GetGFDData (Setup HFSS 3D Layout)

Shows all available GFD (i.e., Green's Function Data) files.

UI Access	None.

Parameters	None.
Return Value	Array of strings containing the names of identified Green's Function Data files.

Python Syntax	GetGFDData()
Python Example	<code>oEditor.GetGDData()</code>

VB Syntax	GetGFDData<DefName>
VB Example	<code>oEditor.GetGFDData</code>

GetMeshOperations

Get the names of all mesh operations for a given setup.

UI Access	N/A
Parameters	None.
Return Value	Array of names of all mesh operations.

Python Syntax	GetMeshOperations
Python Example	<pre>oModule.GetMeshOperations() res = oModule.GetMeshOperations("Setup1")</pre>

	['Length 1', 'Length 2', 'SkinDepth 1']

VB Syntax	GetMeshOperations
VB Example	<pre>oModule.GetMeshOperations " " res = oModule.GetMeshOperations "Setup1" ['Length 1', 'Length 2', 'SkinDepth 1']</pre>

GetSetupData (Layout Editor)

Get setup data.

Command: None

Syntax: GetSetupData <setup>

Return Value: String array of setup information.

Parameters: <setup>

Type: string

VB Example: oSetup = oDesign.GetModule("SolveSetups")

setup = oSetup.GetSetups()

dat = oSetup.GetSetupData(setup[0])

LayoutMeshOverlay (Layout Editor)

Use: Overlay a mesh for a setup in layout

Syntax: LayoutMeshOverlay(STRING setup) // setup name

VB Example: oModule.LayoutMeshOverlay <setup>

Layout3DMeshOverlay (Layout Editor)

Use: Overlay a mesh for a setup in 3D

Syntax: Layout3DMeshOverlay(String setup) // setup name

Example: oModule.Layout3DMeshOverlay <setup>

ListVariations (Layout Editor)

Use: Get a list of solution variations for a given solution

Syntax: ListVariations(String SolutionName) // solution name

Example: oModule.ListVariations("NWA1")

PullExcitationsFromTransient (Layout Editor)

Uses a Circuit or HFSS 3D Layout transient excitation source as an external source for a near field simulation.

UI Access	From the Project Manager window, right-click a setup and select Pull Excitations .		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the near field setup.
	<ExcitationSource>	String	Name of the transient setup.
	<EnableExternalExcitation>	Boolean	True – transient excitation used as an external source False – transient excitation not used as the source
Return Value	None		

Python Syntax	oModule.PullExcitationsFromTransient(<SetupName>, <ExcitationSource>, <EnableExternalExcitation>)		

Python Example	<code>oModule.PullExcitationsFromTransient("Near Field Setup 1", "TransientSetup1", True)</code>

VB Syntax	<code>oModule.PullExcitationsFromTransient<SetupName>, <ExcitationSource>, <EnableExternalExcitation></code>
VB Example	<code>oModule.PullExcitationsFromTransient "Near Field Setup 1", "TransientSetup1", True</code>

RefreshMeshOverlays (Layout Editor)

Use: Refreshes mesh display for a setup

Syntax: RefreshMeshOverlays(STRING setup) // setup name

Example: `oModule.RefreshMeshOverlays <setup>`

Rename (Layout Editor)

Renames a setup.

UI Access	From the Project Manager window, right-click a setup and select Rename .		
Parameters	Name	Type	Description
	<SetupName>	String	Current name of the setup.
	<NewName>	String	User-entered new name of the setup.
Return Value	None		

Python Syntax	<code>oModule.Rename(<SetupName>, <NewName>)</code>
Python Example	<code>oModule.Rename ("Q3D1", "Q3D2")</code>

VB Syntax	<code>oModule.Rename<OldName>, <NewName></code>
VB Example	<code>oModule.Rename "Q3D1", "Q3D2"</code>

RenameSweep (Layout Editor)

Renames a simulation sweep.

UI Access	From the Project Manager window, expand a setup. Then right-click a sweep and select Rename .		
Parameters	Name	Type	Description
	<SetupName>	String	Name of the setup.
	<SweepName>	String	Current name of the sweep.
	<NewSweepName>	String	User-entered new name of the sweep.
Return Value	None		

Python Syntax	<code>oModule.RenameSweep(<SetupName>, <SweepName>, <NewSweepName>)</code>
Python Example	<code>oModule.RenameSweep("Q3D2", "Sweep1", "Sweep2")</code>

VB Syntax	<code>oModule.RenameSweep <SetupName>, <SweepName>, <NewSweepName></code>
VB Example	<code>oModule.RenameSweep "Q3D2", "Sweep1", "Sweep2"</code>

22 - Excitations Commands

The following commands are available in the Excitations module. These commands should be executed by the **oDesign** object. For example:

```
oDesign.GetModule("Excitations" or "Boundaries")  
oModule.CommandName <args>
```

The following commands are described in this section:

[Add](#)

[AddRefPort](#)

[AddRefPortUsingEdges](#)

[ConvertPlaneToTrace](#)

[CoupleEdgePorts](#)

[DecoupleEdgePorts](#)

[Delete](#)

[DeleteExcitation](#)

[DeleteSource](#)

[DeleteProbePortAndVia](#)

[Edit](#)

[EditCircuitPort](#)

[EditExcitations](#)

[GetAllBoundariesList](#)

[GetAllPortsList](#)

[RemoveRefPort](#)

[Rename](#)

[RenameSource](#)

[SelectInLayout](#)

Add [Excitation HFSS 3D Layout]

Use: Adds an excitation

Syntax: Add(ARRAY excitation_data) // excitation data

Example:

```
oModule. Add <excitation_data>
```

AddRefPort (Layout Editor)

Use: Adds a reference to an existing port

Syntax: AddRefPort(STRING portname) // name of the port

STRING edgeportname) // name of the edgeport

VB Example: oModule. AddRefPort portname edgeportname

AddRefPortUsingEdges (Layout Editor)

Use: Creates an edge port with reference using the specified edge

Command: None

Syntax: <port name> ,

 Array("Name:EdgeRefs",

"edge:=", Array(<edge information>)

Return Value: None

Parameters: <port name>: string that will name the created edge port

 <edge information>: describes the edge.

There are two choices: primitive edge or via edge.

<edge information> for a primitive edge:

"et:=", "pe", "prim:=", <primitive name>, "edge:=", <edge number>

<primitive name>: text that is the name of the primitive to use

<edge number>: an integer indexing the edge of the primitive to use

<edge information> for via edge:

"et:=", "pse", "sel:=", <via name>, "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 name>: 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)

```
oModule.AddRefPortUsingEdges "Port1", Array("NAME:EdgeRefs", "edge=",  
Array("et=", "pe", "prim=", "rect_3", "edge=", 1))
```

```
oModule.AddRefPortUsingEdges "Port1", Array("NAME:EdgeRefs", "edge=",  
Array("et=", "pse", "sel=", "via_5", "layer=", 10, "sx=", 0.0015, "sy=", 0.0015,  
"ex=", -0.0015, "ey=", 0.0015, "h=", 0, "rad=", 0))
```

ConvertPlaneToTrace (Layout Editor)

Use: Convert planes to traces

Syntax: ConvertPlaneToTrace <polygon information>

Return Value: None

Parameters: <Polygon_Data>

Type: array

VB Example: oEditor.ConvertPlanetoTrace Array("NAME:elements", "poly_10", "poly_9", "poly_8", ...)

ConvertTraceToPlane (Layout Editor)

Use: Convert traces to planes

Syntax: ConvertTraceToPlane <line information>

Return Value: None

Parameters: <Line_Data>

Type: array

VB Example: oEditor.ConvertTracetoPlane Array("NAME:elements", " line _10", " line _9", " line _8", ...)

CoupleEdgePorts

Use: Couples valid edge ports

Syntax: CoupleEdgePorts(VARIANT coupledlist) //entry containing ports to be coupled

Example: oModule.CoupleEdgePorts Array("Port1", "Port2")

CreateGroundPortComponent (Layout Editor)

Use: Create a ground port component

Command: Right-click selected port > Port > Nexxim Ports > Add Grounds at Unconnected Pins

Return Value: None

Parameters: <Component_Data>

Type: array

VB Example: oEditor.CreateGroundPortComponent Array("NAME:elements", "Comp37", "Comp10", ...)

CreateInterfaceGround (Layout Editor)

Use: Create an interface ground

Command: Right-click selected port > Port > Nexxim Ports > Circuit Ground

Return Value: None

Parameters: <Port_Data>

Type: array

VB Example: oEditor.CreateInterfaceGround Array("NAME:NexximPort", "Name:=", "NexximGnd3", _
"X:=", -0.00103067385498434, "Y:=", -0.00738649582490325, "Rot:=", 0)

CreateInterfacePort (Layout Editor)

Use: Create an interface port

Command: Right-click selected port > Port > Nexxim Ports > Circuit Interface

Syntax: CreateInterfacePort <Port_Data>

Parameters: <Port_Data>

Type: array

VB Example: oEditor.CreateInterfacePort Array("NAME:NexximPort", "Name:=", "Port1", _
"X:=", -0.00309202168136835, "Y:=", 0.00314928125590086, "Rot:=", 0)

CreateInterfacePortComponent (Layout Editor)

Use: Create an interface port component

Command: Right-click selected port > Port > Nexxim Ports > Add Interface Ports at Unconnected Pins

Syntax: CreateInterfacePortComponent <Port_Data>

Parameters: <Component_Data>

Type: array

VB Example: oEditor.CreateInterfacePortComponent Array("NAME:elements", "Comp37", "Comp10", ...)

DecoupleEdgePorts

Use: Decouples edge ports

Syntax: DecoupleEdgePorts(

VARIANT decoupledlist) //entry containing ports to be decoupled

Example: oModule.DecoupleEdgePorts Array("Port1:T1", "Port1:T2")

Delete[Excitation HFSS 3D Layout]

Use: Deletes an excitation

Syntax: Delete(STRING excitation_name) // excitation name

Example: oModule.Delete <excitation_name>

Delete [Interface Port]

Use: Deletes an interface port

Syntax: Delete(STRING: Interface Port name)

VB Example: oModule.Delete "PortName"

Delete[Excitation HFSS 3D Layout]

Use: Deletes an excitation

Syntax: Delete(STRING excitation_name) // excitation name

Example: oModule. Delete <excitation_name>

DeleteProbePortAndVia (Layout Editor)

Use: Deletes a coaxial probe port and the associated via

Syntax: DeleteProbePortAndVia(STRING port_name) // Probe Port name

STRING via_name) //Via name

Example: oModule. DeleteProbePortAndVia <port_name> <via_name>

DeleteSource [Interface Source]

Use: Deletes an interface source

Syntax: DeleteSource (STRING: Interface Source name)

VB Example: oModule. DeleteSource "SourceName"

Edit (Layout Editor)

Edits a setup.

Note: The following tables display **example** parameters applicable to a Near Field solution setup. Selecting a different setup will also add, remove, or change the applicable parameters.

UI Access	From the Project Manager window, right-click a setup and select Properties .		
Parameters	Name	Type	Description
	<SetupArray>	Array	An array that contains these parameters in order: • <NAME:SetupName> • <NameArray> • <CustomSetup> • <Position> • <SimSetupType> • <SimSettingsArray> • <SolveSetupType> This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.
	<NAME:SetupName>	String	User-entered name of the setup.
	<NameArray>	Array	An array that contains these parameters in order: • <NAME:Properties> • <Enable>
	<NAME:Properties>	String	Identifies the contents of the array (i.e., Properties).
	<Enable>	Boolean	True – setup is active False – setup is inactive
	<SimSetupType>	String	Identifies the type of setup (e.g., kSIwaveNearField).
	<SimSettingArray>	Array	<i>Optional.</i> An array that contains these parameters in order: • <NAME:SimulationSettings> • <Enabled>

		• <UseSISettings> • <UseCustomSettings> • <SISliderPros> • <PISliderPros> • <SIWAdvancedArray> • <SIWDCArray> • <SIWDCAdvancedArray> • <SIWSParamArray> • <NearFieldSettingArray> • <NAME:SweepDataList> This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.
<NAME:SimulationSettings>	String	Identifies the contents of the array (i.e., SimulationSettings).
<Enabled>	Boolean	True – array settings are active False – array settings are inactive
<UseSISettings>	Boolean	True – uses SI Simulation settings False – uses PI Simulation settings
<UseCustomSettings>	Boolean	True – overrides <UseSISettings> and activates <SIWAdvancedArray> parameters False groups by range
<SISliderPros>	Integer	0 – optimizes simulation for speed 1 – optimizes simulation to balance speed and accuracy 2 – optimizes simulation for accuracy
<PISliderPros>	Integer	0 – optimizes simulation for speed 1 – optimizes simulation to balance speed and accuracy 2 – optimizes simulation for accuracy
<SIWAdvancedArray>	Array	<i>Optional.</i> An array that contains these parameters in order:

		• <NAME:SIWAdvancedSettings> • <IncludeCoPlaneCoupling> • <IncludeInterPlaneCoupling> • <IncludeSplitPlaneCoupling> • <IncludeFringeCoupling> • <IncludeTraceCoupling> • <XtalkThreshold> • <MaxCoupledLines> • <MinVoidArea> • <MinPadAreaToMesh> • SnapLengthThreshold> • <MeshAutoMatic> • <MeshFrequency> • <ReturnCurrentDistribution> • <INcludeVISources> • <IncludeInfGnd> • <InfGndLocation> • <PerformERC> • <IgnoreNonFunctionalPads> This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.
<NAME:SIWAdvancedSettings>	String	Identifies the contents of the array (i.e., SIWAdvancedSettings).
<IncludeCoPlaneCoupling>	Boolean	True – coplane coupling active False – coplane coupling inactive
<IncludeInterPlaneCoupling>	Boolean	True – intra-plane coupling active False – intra-plane coupling inactive
<IncludeSplitPlaneCoupling>	Boolean	True – split-plane coupling active

		False – split-plane coupling inactive
<IncludeFringeCoupling>	Boolean	True – cavity field coupling active False – cavity field coupling inactive
<IncludeTraceCoupling>	Boolean	True – trace coupling active False – trace coupling inactive
<XtalkThreshold>	Integer	Maximum cross-talk threshold, in decibels.
<MaxCoupledLines>	Integer	The number of allowed coupled lines.
<MinVoidArea>	Integer	Minimum size of voids; voids larger than the minimum are meshed.
<MinPadAreaToMesh>	Integer	Minimum size of pads; pads larger than the minimum are meshed.
<SnapLengthThreshold>	Integer	Minimum distance of vertices; vertices less than the minimum are snapped.
<MeshAutoMatic>	Boolean	True – mesh are automatically refined False – mesh are not automatically refined
<MeshFrequency>	Integer	Specific frequency at which mesh are refined.
<ReturnCurrentDistribution>	Boolean	True – returns current distribution is traced False – returns current distribution is not traced
<IncludeVISources>	Boolean	True – parasitic elements from voltage/current sources are included False – parasitic elements are not included
<IncludeInfGnd>	Boolean	True – infinite ground plane is included False – infinite ground plane is not included
<InfGndLocation>	Integer	Distance below bottom layer in stackup to place infinite ground plane.
<PerformERC>	Boolean	True – performs ERC during simulation False – does not perform ERC during simulation
<IgnoreNonFunctionalPads>	Boolean	True – excludes non-functional pads from resonant mode, frequency sweep, and SYZ simulations False – does not exclude non-functional pads from resonant mode, frequency sweep, and SYZ simulations.

<SIWSParmSettings>	Array	<i>Optional.</i> An array that contains these parameters in order: • <UseStateSpace> • <Interpolation> • <Extrapolation> • <DCBehavior> This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.
<UseStateSpace>	Boolean	True – uses the State Space Model False – uses a Custom Model
<Interpolation>	Integer	0 – sets Interpolation to Point 1 – sets Interpolation to Point 2 – sets Interpolation to Point
<Extrapolation>	Integer	0 – sets Extrapolation to Zero padding 1 – sets Extrapolation to Same as last point 2 – sets Extrapolation to Linear extrapolation from last 2 points 3 – sets Extrapolation to Constant magnitude linear phase extrapolation
<DCBehavior>	Integer	0 – sets DC Behavior to Zero padding 1 – sets DC Behavior to Same as last point 2 – sets DC Behavior to Linear extrapolation from last 2 points 3 – sets DC Behavior to Constant magnitude linear phase extrapolation 4 – sets DC Behavior to Series Capacitor 5 – sets DC Behavior to Leave all signal lines open circuited
<SIWSNearFieldArray>	Array	An array that contains these parameters in order: • <NAME:SIWNearFieldSettings> • <UseExternalSources> • <ExternalSourceFilePath>

			• <InterpolateSpectrum> • <IncludeInfGnd> • <PosXoffset> • <NegXoffset> • <PosYoffset> • <NegYoffset> • <PosZoffset> • <NegZoffset> • <MinAdaptPasses> • <MaxAdaptPasses> • <NewTrianglesPerPass> • <GlobalErrorTol> • <UseUserEdgeLength> • <MeshEdgeLength> • <ExportData> • <ExportDataPath> This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.
	<NAME:SIWNearFieldSettings>	String	Identifies the contents of the array (i.e., SIWNearFieldSettings).
	<UseExternalSources>	Boolean	True – uses external Excitations definitions False – uses Excitations definitions defined in project
	<ExternalSourceFilePath>	String	Directory path to external Excitations definitions file.
	<InterpolateSpectrum>	Boolean	True – interpolates spectrum at missing frequency points (external Excitations definitions file only) False – does not interpolate spectrum at missing frequency points

	<IncludeInfGnd>	Boolean	True – infinite ground plane is included False – infinite ground plane is not included
	<PosXoffset>	Integer	Value determines the +X offset of a cube that encloses the design; the Near Field is evaluated on the surface of this cuboid.
	<NegXoffset>	Integer	Value determines the -X offset of a cube that encloses the design; the Near Field is evaluated on the surface of this cuboid.
	<PosYoffset>	Integer	Value determines the +Y offset of a cube that encloses the design; the Near Field is evaluated on the surface of this cuboid.
	<NegYoffset>	Integer	Value determines the -Y offset of a cube that encloses the design; the Near Field is evaluated on the surface of this cuboid.
	<PosZoffset>	Integer	Value determines the +Z offset of a cube that encloses the design; the Near Field is evaluated on the surface of this cuboid.
	<NegZoffset>	Integer	Value determines the -Z offset of a cube that encloses the design; the Near Field is evaluated on the surface of this cuboid.
	<MinAdaptPasses>	Integer	Minimum number of passes the adaptive analysis will make, even if convergence criteria are met.
	<MaxAdaptPasses>	Integer	Maximum number of passes the adaptive analysis will make, unless convergence criteria are met.
	<NewTrianglesPerPass>	Integer	Percentage of triangles to add per pass.
	<GlobalErrorTol>	Integer	Maximum relative number of errors to allow between two successive adaptive passes.
	<UseUserEdgeLength>	Boolean	True – uses custom maximum edge length False – automatically determines maximum edge length
	<MeshEdgeLength>	Integer	Maximum edge length.
	<ExportData>	Boolean	True – exports Near Field data False – does not export Near Field data
	<ExportDataPath>	String	Path to directory in which to save Near Field data.
	<NAME:SweepDataList>	String	Identifies the determining factors of the Near Field simulation (i.e., sweeps) (i.e., SweepDataList).
	<SolveSetupType>	String	Identifies the type of setup (i.e., SIwaveNearField).
Return Value	None		

Python Syntax	oModule.Edit([<NAME:SIwaveNearField1>, [<NAME:Properties>, <Enable>], <SimSetupType>, [<NAME:SimulationSettings>, <Enabled>, <UseSISettings>, <UseCustomSettings>, <SISliderPos>, <PISliderPos>], [<NAME:SIWAdvancedSettings>, <IncludeCoPlaneCoupling>, <IncludeInterPlaneCoupling>, <IncludeSplitPlaneCoupling>, <IncludeFringeCoupling>, <IncludeTraceCoupling>, <XtalkThreshold>, <MaxCoupledLines>, <MinVoidArea>, <MinPadAreaToMesh>, <SnapLengthThreshold>, <MeshAutoMatic>, <MeshFrequency>, <ReturnCurrentDistribution>, <IncludeVISources>, <IncludeInfGnd>, <InfGndLocation>, <PerformERC>, <IgnoreNonFunctionalPads>], [<NAME:SIWSParmSettings>, <UseStateSpace>, <Interpolation>, <Extrapolation>, <DCBehavior>], [<NAME:SIWNearFieldSettings>, <UseExternalSources>, <ExternalSourceFilePath>, <InterpolateSpectrum>, <IncludeInfGnd>, <PosXoffset>, <NegXoffset>, <PosYoffset>, <NegYoffset>, <PosZoffset>, <NegZoffset>, <MinAdaptPasses>, <MaxAdaptPasses>, <NewTrianglesPerPass>, <GlobalErrorTol>, <UseUserEdgeLength>, <MeshEdgeLength>, <ExportData>, <ExportDataPath>]], [<NAME:SweepDataList>], <SolveSetupType>])
Python Example	<pre> oModule.Edit(["NAME:SIwaveNearField1", ["NAME:Properties", "Enable:=" , "true"], "SimSetupType:=" , "kSIwaveNearField", ["NAME:SimulationSettings", "Enabled:=" , True, </pre>

```

"UseSISettings:="      , False,
"UseCustomSettings:="  , False,
"SISliderPos:="        , 2,
"PISliderPos:="        , 0,
[
  "NAME:SIWAdvancedSettings",
  "IncludeCoPlaneCoupling:=", False,
  "IncludeInterPlaneCoupling:=", False,
  "IncludeSplitPlaneCoupling:=", False,
  "IncludeFringeCoupling:=", False,
  "IncludeTraceCoupling:=", False,
  "XtalkThreshold:="    , "-34",
  "MaxCoupledLines:="   , 12,
  "MinVoidArea:="       , "2mm2",
  "MinPadAreaToMesh:="  , "1mm2",
  "MinPlaneAreaToMesh:=" , "6.25e-6mm2",
  "SnapLengthThreshold:=" , "2.5um",
  "MeshAutoMatic:="     , True,
  "MeshFrequency:="     , "4GHz",
  "ReturnCurrentDistribution:=", False,
  "IncludeVISources:="   , False,

```



```
"IncludeInfGnd:="      , False,
"InfGndLocation:="     , "0mm",
"PerformERC:="         , False,
"IgnoreNonFunctionalPads:=", True
],
[
  "NAME:SIWSParmSettings",
  "UseStateSpace:="    , True,
  "Interpolation:="    , 1,
  "Extrapolation:="    , 3,
  "DCBehavior:="       , 3
],
[
  "NAME:SIWNearFieldSettings",
  "UseExternalSources:=" , False,
  "ExternalSourceFilePath:=", "",
  "InterpolateSpectrum:=" , True,
  "IncludeInfGnd:="      , False,
  "PosXOffset:="         , "1mm",
  "NegXOffset:="         , "1mm",
```

```
"PosYoffset:="      , "1mm",
"NegYoffset:="      , "1mm",
"PosZoffset:="      , "1mm",
"NegZoffset:="      , "1mm",
"MinAdaptPasses:="  , 1,
"MaxAdaptPasses:="  , 10,
"NewTrianglesPerPass:=" , 20,
"GlobalErrorTol:="  , 3,
"UseUserEdgeLength:=" , False,
"MeshEdgeLength:="  , "1mm",
"ExportData:="      , False,
"ExportDataPath:="   , ""
]
],
[
  "NAME:SweepDataList"
],
"SolveSetupType:="  , "SIwaveNearField"
])
```

```
oModule.Edit(<NAME:SIwaveNearField1>, (<NAME:Properties>, <Enable>), <SimSetupType>, (<NAME:SimulationSettings>, <Enabled>, <UseSplitPlaneCoupling>, <IncludeSplitPlaneCoupling>, <IncludeFringeCoupling>, <IncludeTraceCoupling>, <XtalkThreshold>, <MaxCoupledLines>, <IncludeVISources>, <IncludeInfGnd>, <InfGndLocation>, <PerformERC>, <IgnoreNonFunctionalPads>), (<NAME:SIWSPParamSettings>, <UseStateSpace>, <InterpolateSpectrum>, <IncludeInfGnd>, <PosXoffset>, <NegXoffset>, <PosYoffset>, <NegYoffset>, <PosZoffset>, <NegZoffset>, <ExportData>, <ExportDataPath>)), (<NAME:SweepDataList>), <SolveSetupType>)
```

```
V-
B
E-
x-
a-
m-
p-
l-
e
oModule.Edit Array("NAME:SIwaveNearField1", Array("NAME:Properties", "Enable:=", "true"), "SimSetupType:=",
"UseCustomSettings:=", False, "SISliderPos:=", 2, "PISliderPos:=", 0, Array("NAME:SIWAdvancedSettings", "
False, "IncludeFringeCoupling:=", False, "IncludeTraceCoupling:=", False, "XtalkThreshold:=", "-34", "Max
"6.25e-6mm2", "SnapLengthThreshold:=", "2.5um", "MeshAutoMatic:=", True, "MeshFrequency:=", "4GHz", "Retu
tion:=", "0mm", "PerformERC:=", False, "IgnoreNonFunctionalPads:=", True), Array("NAME:SIWDCSettings", "U
"DCSliderPos:=", 1), Array("NAME:SIWDCAdvancedSettings", "DcMinPlaneAreaToMesh:=", "0.25mm2", "DcMinVoidA
"MaxNumPasses:=", 5, "MinNumPasses:=", 1, "PercentLocalRefinement:=", 20, "EnergyError:=", 2, "MeshBws:=",
aSides:=", 8), Array("NAME:SIWSPParamSettings", "UseStateSpace:=", True, "Interpolation:=", 1, "Extrapolat
alSourceFilePath:=", "", "InterpolateSpectrum:=", True, "IncludeInfGnd:=", False, "PosXoffset:=", "1mm",
"NegZoffset:=", "1mm", "MinAdaptPasses:=", 1, "MaxAdaptPasses:=", 10, "NewTrianglesPerPass:=", 20, "Globa
"ExportDataPath:=", "")), Array("NAME:SweepDataList"), "SolveSetupType:=", "SIwaveNearField"))
```

EditCircuitExcitations (Layout Editor)

Use: Edits the properties of all the existing circuit excitations

Syntax: EditCircuitExcitations Array("NAME:NexximPostProcess", Array("NAME:portname", do-post-proc, "de-embedding distance", "re-normalization impedance")...))

Return Value: None

```
VB Example: oModule.EditCircuitExcitations Array("NAME:NexximPostProcess", Array("Port1", true, "50ohm + 0i ohm"), Array("Port5", true, "75ohm + 0i ohm"))
```

EditCircuitPort [Interface Port]

Use: Edit a circuit port

Command: None

Syntax: EditCircuitPort <Port Name> <Port Info> <Port Properties>

Return Value: None

Parameters: <Port Name> - Type: string

 <Port Info> - Type: array

 <Port Properties> - Type: array

VB Example:

```
oModule.EditCircuitPort "Port2", Array("NAME:Port2", "IIPortName:=", "Port2", "SymbolType:=", _  
0), Array(Array("NAME:Properties", Array("NAME:NewProps", Array("NAME:term", "PropType:=", _  
"TextProp", "OverridingDef:=", true, "Value:=", "Model1"))), Array("NAME:ChangedProps", Array  
("NAME:TerminationData", "Value:=", _  
"Zo"), Array("NAME:pnum", "Value:=", "2"), Array("NAME:noisetemp", "Value:=", "16.85")))
```

EditExcitations (Layout Editor)

Edits the properties of all existing excitations.

Note: GapSource should always be set.

UI Access	None.		
Parameters	Name	Type	Description
	<exciteArray>	Structured Array	Excitations block.
	<ppArray>	Structured Array	Post-processing block.
	<gapSource>	String	Gap source.
	<pushExParams>	Structured Array	PushExParams block. "" by default.
Return Value	None.		

Python Syntax	EditExcitations (<exciteArray>, <ppArray>, <gapSource>, <pushExParams>))
Python Example	<pre>oModule.EditExcitations(["NAME:Excitations", ["Port1", "0deg", "0.099V"], ["Port2", "0deg", "0.099V"]], ["NAME:PostProcess", ["Port1", True, True, "0mm", "50ohm + 0i ohm"], ["Port2", True, True, "0mm", "50ohm + 0i ohm"]], </pre>

	<code>"GapSource, IncidentVoltage", [])</code>

VB Syntax	<code>EditExcitations <exciteArray>, <ppArray>, <gapSource>, <pushExParams></code>
VB Example	<pre>oModule.EditExcitations Array("NAME:Excitations", Array("NAME:Port1", "0deg", "1V")), Array("NAME:PostProcess", Array("NAME:Port1", true, "1mm", "50ohm +0i ohm")), "GapSource", ""</pre>

GetAllBoundariesList (Layout Editor)

Use: Get a list of all boundaries.

Command: None

Syntax: GetAllBoundariesList <bounds>

Return Value: String array of list of all boundaries.

Parameters: <bounds>

Type: string

Example: `oEx = oDesign.GetModule('Excitations')`

`bounds = oEx.GetAllBoundariesList()`

GetAllPortsList (Layout Editor)

Use: Returns the names of all Ensemble ports

Syntax: GetAllPortsList

Return Value: Array of Strings

Parameters: None

Example: Dim portArr

```
portArr= oModule.GetAllPortsList
```

Rename [Interface Port]

Use: Renames an interface port

Syntax: Rename (STRING: Interface Port name)(STRING: New name)

VB Example: oModule.Rename "OldName" "NewName"

RenameSource [Interface Source]

Use: Renames an interface source

Syntax: RenameSource (STRING: Interface Source name)

VB Example: oModule.RenameSource "OldName" "NewName"

Rename (Layout Editor)

Use: Renames an excitation

Syntax: Rename(STRING oldname) // old excitation name
STRING newname) // new excitation name

Example: oModule. Rename <oldname> <newname>

RemoveRefPort (Layout Editor)

Use: Removes the reference for an existing port

Syntax: RemoveRefPort(

STRING portname) // name of the port

Example: oModule. RemoveRefPort portname

SelectInLayout (Layout Editor)

Use: Selects and highlights an excitation in the layout editor

Syntax: SelectInLayout(STRING excitation_name) // excitation name

Example: oModule. SelectInLayout <excitation_name>

This page intentionally
left blank.

23 - 2.5D Via Commands

The following commands are available in the LayoutVias module. These commands should be executed by the **oDesign** object. For example:

```
oDesign.GetModule("LayoutVias")  
oModule.CommandName <args>
```

The following commands are described in this section:

[ConvertPrimitives](#)

[Delete](#)

[Edit](#)

[MultipleEdit](#)

[Rename](#)

[SelectInLayout](#)

ConvertPrimitives (Layout Editor)

Use: Convert the selected primitives to 2.5D vias

Syntax: ConvertPrimitives(ARRAY primvia_data) // primitive data

Example: oModule.ConvertPrimitives primvia_data

Delete [Via HFSS 3D Layout]

Use: Delete a 2.5 via

Syntax: Delete(

STRING via) // 2.5 via name

Example: oModule. Delete via

Edit [Via HFSS 3D Layout]

Use: Edits a 2.5 via

Syntax: Edit(STRING via) // 2.5 via name

ARRAY via_data) // data for the 2.5 via

Example: oModule. Edit via via_data

MultipleEdit [multiple via HFSS 3D Layout]

Use: Edit the properties of a collection of 2.5D via

Syntax: MultipleEdit(ARRAY multiplevia_data) // multiple 2.5 via data

Example: oModule. MultipleEdit multiplevia_data

Rename [Via HFSS 3D Layout]

Use: Renames a 2.5 via

Syntax: Rename(STRING oldname) // old 2.5 via name

STRING newname) // new 2.5 via name

Example: oModule. Rename oldname newname

SelectInLayout [Via HFSS 3D Layout]

Use: Selects and highlights the 2.5 via in the layout editor

Syntax: SelectInLayout(String via) // 2.5 via name

Example: oModule. SelectInLayout via

This page intentionally
left blank.

24 - Cavity Commands

The following commands are available in the Cavities module. These commands should be executed by the **oDesign** object. For example:

```
oDesign.GetModule("Cavities")  
oModule.CommandName <args>
```

The following commands are described in this section:

[Add](#)

[Delete](#)

[Edit](#)

[Rename](#)

[SelectInLayout](#)

Add [Cavity HFSS 3D Layout]

Use: Add a cavity to the design

Syntax: Add(ARRAY cavity_data) // cavity data

Example: oModule. Add cavity_data

Delete [Cavity HFSS 3D Layout]

Use: Delete a cavity

Syntax: Delete(STRING cavity) // cavity name

Example: oModule. Delete cavity

Edit [Cavity HFSS 3D Layout]

Use: Edits a cavity

Syntax: Edit(STRING name) // cavity name

ARRAY cavity_data) // data for the cavity

Example: oModule. Edit name cavity_data

Rename [Cavity HFSS 3D Layout]

Use: Renames a cavity

Syntax: Rename(STRING oldname) // old cavity name

STRING newname) // new cavity name

Example: oModule. Rename oldname newname

SelectInLayout [Cavity HFSS 3D Layout]

Use: Selects and highlights the cavity in the layout editor

Syntax: SelectInLayout(STRING cavity) // cavity name

Example: oModule. SelectInLayout cavity

25 - Layout Scripting

The Layout scripting interface is a set of commands that can be executed by the *Layout Editor* interface of *Design* objects. Accessing the *Layout Editor* interface can be done by the *SetActiveEditor* method of a *Design* object, for example:

```
Dim oEditor  
oEditor = DesignObject.SetActiveEditor ("Layout")
```

The call to each scripting method contains three elements:

- Layout editor
- Method name
- Method arguments (if any)

For instance, in Visual Basic, a call might look like this:

```
oEditor.Delete ("rect_1")
```

The behavior of the scripting methods is similar to the corresponding commands in the Layout editor. This correspondence is particularly evident when recording a script in Layout Editor (**Tools>Record Script**).

Note: A subsection of the Layout Scripting commands are only accessible once **Electronics Desktop** has been placed in IC Editor mode (i.e., **IC** mode). These commands are all located within the [IC Editor Script Commands](#) directory. There are commands common to both the **General** mode and **IC** mode environments in every topic within Layout Scripting, but they are designated (**General Mode and IC Mode**) at the **Table of Contents** level. Unless otherwise noted, common commands function identically in either environment (e.g., **RemoveLayer** is divided into two sections for each mode since the command function is similar but not identical).

The topics for this section include:

[Object Identifiers and Script Recording](#)

[Create Primitives](#)

[Create Voids in Primitives](#)

[Other Creation Methods](#)

[Object Movement and Modification Methods](#)

[Activation and Deactivation Methods](#)

[Layout and Geometry Interrogation Methods](#)

[Boolean Operations on Primitives](#)

[Coordinate System Methods](#)

[IC Editor Script Commands \(IC Mode Only\)](#)

[Layers Methods](#)

[NetClass Methods](#)

[Port Methods](#)

[Miscellaneous Methods](#)

Object Identifiers and Script Recording

Many scripting methods apply to one or more *existing* objects. Objects in scripts are uniquely identified by their *names*, and the object names are typically referenced as:

`Array("name1", "name2")` // name1 and name2 are the names of the objects.

While working interactively in the Layout Editor, the user may observe (but not modify, except in some special cases) the *names* of the currently selected objects through the "Selected Object Properties" dialogs. These same names are the ones that the scripting methods use in order to identify the object the method applies to.

When particular objects are involved in interactive script recording, they are typically selected before the actual execution the particular interactive command. In this case, the *selected objects'* names would appear within the generated script method argument(s).

Create Primitives

The following create primitives are available.

[Create3DLine](#)

[CreateCircle](#)

[CreateLine](#)

[CreatePolygon](#)

[CreateRectangle](#)

[CreateText](#)

AddObject ("3D Line") (Layout Editor)

Adds a 3D line to the active design.

UI Access	From the Layout ribbon, select Draw 3D line .		
Parameters	Name	Type	Description
	<ObjectType>	String	Type of object to add (i.e., must have the value "3D Line").
	<Parameters>	Array	Structured array containing these parameters in order: • <NAME:Contents> • <3D LineGeometry>
	<NAME:Contents>	String	Identifies the contents of the array (i.e., "Contents").
	<3D LineGeometry>	Array	Structured array containing these parameters in order: • <Name> • <LayerName> • <lw> • <gm>
	<Name>	String	Name of the new object.

	<LayerName>	String	Name of the target layer.
	<lw>	Integer	Line width of the new object.
	<gm>	Array	Structured array containing these parameters in order: • <m> • <pnt>
	<m>	Integer	"Placement Mode" - species one of the following: 0 3D - points are placed according to user-specified XYZ coordinates. 1 Top - points are placed on top of the target layer. 2 Middle - points are placed in the middle of the target layer. 3 Bottom - points are placed on bottom of the target layer.
	<pnt>	Array	Structured array containing these parameters in order: • <lid> • <pos>
	<lid>	Integer	Identifies the layer identification number on which the 3D line's position will be affixed (e.g., "2").
	<pos>	Array	Structured array containing these parameters in order: • <X> • <Y> • <Z>
	<X>	Integer	X coordinate of the new object (e.g., -0.08mm).
	<Y>	Integer	X coordinate of the new object (e.g., 0.03mm).
	<Z>	Integer	Y coordinate of the new object (e.g., 0mm).
Return Value	Returns the name of the new 3D line.		

Python Syntax	AddObject(<ObjectType>, [<Parameters>])

Python Example

```

oEditor.AddObject("3D Line",
[
    "NAME:Contents",
    "3D LineGeometry:=",
    [
        "Name:=", "3D Line_12904",
        "LayerName:=", "Postprocessing",
        "lw:=" , "0",
        "gm:=",
        [
            "m:=", 1,
            "pnt:=", [ "lid:=", 2, "pos:=", [ "X:=", "-0.08mm", "Y:=", "0.03mm",
            "Z:=", "0"]],
            "pnt:=" , [ "lid:=" , 2, "pos:=" , [ "X:=" , "-0.07mm", "Y:=" , "0.03mm",
            "Z:=" , "0"]],
            "pnt:=" , [ "lid:=" , 2, "pos:=" , [ "X:=" , "-0.07mm", "Y:=" ,
            "0.021mm", "Z:=" , "0"]],
            "pnt:=" , [ "lid:=" , 2, "pos:=" , [ "X:=" , "-0.08mm", "Y:=" ,
            "0.02mm", "Z:=" , "0"]],
            "pnt:=" , [ "lid:=" , 2, "pos:=" , [ "X:=" , "-0.08mm", "Y:=" ,
            "0.03mm", "Z:=" , "0"]],
            "pnt:=" , [ "lid:=" , 2, "pos:=" , [ "X:=" , "-0.08mm", "Y:=" ,
            "0.03mm", "Z:=" , "0"]],
            "pnt:=" , [ "lid:=" , 2, "pos:=" , [ "X:=" , "-0.08mm", "Y:=" ,

```

	<pre> "0.03mm", "Z:=" , "0"]]]]])</pre>

VB Syntax	AddObject <ObjectType>, (<Parameters>)
VB Example	<pre>oEditor.AddObject "rect", Array("NAME:Contents", "rectGeometry:=", Array ("Name:=", "rect_5", "LayerName:=", "Trace", "lw:=", "0", "Ax:=", "-84mm", "Ay:=", "16mm", "Bx:=", "-77mm", "By:=", "12mm"]])</pre>

CreateCircle (Layout Editor)

Use: Creates a circle primitive object and adds it to the current layout. Returns the name of the newly created object.

Syntax: **CreateCircle**<circle_description>

Parameters: **<circle_description>:**

```
Array("NAME:Contents",  
      "circleGeometry:=", <circle_geometry>) // object description
```

<circle_geometry> :

```
Array("LayerName:=", <layer_name>, // name of the layer  
      "lw:=", <value>, // border line width  
      "x:=", <value>, // center x coordinate  
      "y:=", <value>, // center y coordinate  
      "r:=", <value>) // radius
```

<layer_id> :

integer; never used in scripting

<object_name> :

quoted string, uniquely identifying an object

<value> :

quoted_string_parseable_as_value (e.g. "0.111mm")

VB Example:

```
oEditor.CreateCircle
```

```
Array("NAME:Contents",  
      "circleGeometry:=",  
      Array("Layer:=", 6,  
            "Name:=", "circle_150",  
            "LayerName:=", "Top",  
            "lw:=", "0mm",  
            "x:=", "34mm",  
            "y:=", "-15mm",  
            "r:=", "8.60232526704263mm"))
```

CreateLine (Layout Editor)

Use: Creates a line primitive object.

Syntax: **CreateLine** <line_description>

Return Value: Returns the name of the newly created object.

Parameters: <line_description>:

```
      Array("NAME:Contents",  
            "lineGeometry:=", <line_geometry>)
```

<line_geometry> :

```
      Array("LayerName:=", <layer_name>,  
            "lw:=", <value>, // line width  
            "endstyle:=", <end_style>,
```

```
"joinstyle:=",<join_style>,  
<vertex_sequence> )
```

<end_style> : integer

FlatEnd = 0

ExtendedEnd = 1

RoundEnd = 2

<join_style> : integer

UnmiteredJoin = 0

MiteredJoin = 1

OptimallyMiteredJoin = 2

RadiallyMiteredJoin = 3

ChordMiteredJoin = 4

<vertex_sequence>:

"n:=", integer, // the total count of the vertices

"x0:=",<value>, "y0:=", <value>,

"x1:=",<value>, "y1:=", <value>,

etc. up to vertex count)

VB Example:


```
oEditor.CreateLine
Array("NAME:Contents",
"lineGeometry:=",
Array("Layer:=", 6,
"Name:=", "line_1",
"LayerName:=", "Top",
"lw:=", "2mil",
"endstyle:=", 0,
"joinstyle:=", 0,
"n:=", 4,
"x0:=", "-32mm", "y0:=", "2mm",
"x1:=", "-5mm", "y1:=", "12mm",
"x2:=", "1mm", "y2:=", "-4mm",
"x3:=", "19mm", "y3:=", "3mm"))
```

AddObject ("Partition Line") (IC Editor)

Create an additional partition line for a mesh region (i.e., CreateMeshRegion).

UI Access	From the Layout Editor , select a mesh region. Then right-click and select Mesh Region > Add a partition line .

Parameters	Name	Type	Description
	<ObjectType>	String	Type of object to add (i.e., must have the value "mesh region partition line").
	<Contents>	Array	Structured array containing these parameters in order: • <NAME:Contents> • <owner> • <mesh region partition lineGeometry> • <snap start> • <snap end>
	<NAME:Contents>	String	Identifies the contents of the array (i.e., "Contents").
	<owner>	String	Name of the mesh region in which to create the new partition line.
	<mesh region partition lineGeometry>	Array	Name of the array containing the geometry of the new partition line, containing these parameters in order: • <Name> • <LayerName> • <lw> • <endstyle> • <StartCap> • <EndCap> • <n> • <U> • <x> • <y> • <MR>
	<Name>	String	Name of the new partition line.

	<LayerName>	Integer	Name of the layer on which to create the new partition line (i.e., "MeshRegions").
	<lw>	Integer	Line width of the new partition line, including unit of measurement (i.e., "0.01mm").
	<endstyle>	Integer	Of the new partition line, all vertices (i.e., points) will be: 0 — round. 1 — corner (i.e., square). 2 — mitered (i.e., extended to a point where it meets a corner).
	<StartCap>	Integer	Of the new partition line, the starting point of the line will be: 0 — round. 1 — flat. 2 — extended (i.e., extended from the starting point beyond the actual beginning of its path).
	<Endcap>	Integer	Of the new partition line, the ending point of the line will be: 0 — round. 1 — flat. 2 — extended (i.e., extended beyond the actual end point). 3 — Same as Start.
	<n>	Integer	The number of vertices in the new partition line.
	<U>	String	The unit of measurement by which the new partition will be created.
	<x>	Integer	X coordinate of a point in the new partition line.
	<y>	Integer	Y coordinate of a point in the new partition line.
	<MR>	String	A number and unit of measurement representing the miter ratio (i.e., of the new partition line, assuming multiple segments, the size of the point where two segments meet).
	<snap start>	Boolean	True start of the new partition line will snap to a border of the mesh region.
	<snap end>	Boolean	True end of the new partition line will snap to a border of the mesh region.
Return Value	Returns the name of the new partition line.		

Python Syntax	AddObject(<ObjectType>, [<Contents>])
Python Example	<pre>oEditor.AddObject("mesh region partition line", ["NAME:Contents", "owner:=" , "meshregion__11", "mesh region partition lineGeometry:=", ["Name:=" , "line__18", "LayerName:=" , "MeshRegions", "lw:=" , "0.01mm", "endstyle:=" , 0, "StartCap:=" , 0, "EndCap:=" , 0, "n:=" , 3, "U:=" , "mm", "x:=" , 19, "y:=" , 9, "x:=" , 15, "y:=" , 4, "x:=" , 20, "y:=" , 1, "MR:=" , "600mm"] ,</pre>

	<pre>"snap start:=" , True, "snap end:=" , True])</pre>

VB Syntax	AddObject <ObjectType>, (<Contents>)
VB Example	<pre>oEditor.AddObject "mesh region partition line", Array("NAME:Contents", "owner:=", "meshregion__11", "mesh region partition lineGeometry:=", Array("Name:=", "line__ 18", "LayerName:=", "MeshRegions", "lw:=", "0.01mm", "endstyle:=", 0, "StartCap:=", 0, "EndCap:=", 0, "n:=", 3, "U:=", "mm", "x:=", 19, "y:=", 9, "x:=", 15, "y:=", 4, "x:=", 20, "y:=", 1, "MR:=", "600mm"), "snap start:=", True, "snap end:=", True)</pre>

CreatePolygon (Layout Editor)

Use: Creates a polygon primitive object.

Syntax: **CreatePolygon**<polygon_description>

Return Value: Returns the name of the newly created object.

Parameters: <polygon_description>:

```
Array("NAME:Contents",
"lineGeometry:=", <polygon_geometry>)
```

<polygon_geometry> :

```
Array("LayerName:=", <layer_name>, // layer  
"lw:=", <value>, // border line width  
<vertex_sequence> ) // polygon boundary
```

VB Example:

```
oEditor.CreatePolygon  
Array("NAME:Contents",  
"polyGeometry:=",  
Array("Layer:=", 6,  
"Name:=", "poly_5",  
"LayerName:=", "Top",  
"lw:=", "0mm",  
"n:=", 5,  
"x0:=", "-35mm", "y0:=", "17mm",  
"x1:=", "-37mm", "y1:=", "5mm",  
"x2:=", "-30mm", "y2:=", "8mm",  
"x3:=", "-30mm", "y3:=", "8mm",  
"x4:=", "-30mm", "y4:=", "8mm"))
```

CreateRectangle (Layout Editor)

Use: Creates a rectangle primitive object and adds it to the current layout.

Syntax: **CreateRectangle**<rectangle_description>

Return Value: Returns the name of the newly created object.

Syntax: <rectangle_description>:

```
Array("NAME:Contents",
    "rectGeometry:=", <rectangle_geometry>)
    <rectangle_geometry> :
    Array("LayerName:=", <layer_name>, // placement layer
        "lw:=", <value>, // line width
        "x:=", <value>, // center X coordinate
        "y:=", <value>, // center Y coordinate
        "w:=", <value>, // width (X-direction size)
        "h:=", <value>, // height (Y-direction size)
        "ang:=", <value>)) // rotation
```

VB Example:

```
oEditor.CreateRectangle
Array("NAME:Contents",
    "rectGeometry:=",
    Array("Layer:=", 6,
        "Name:=", "rect_4",
        "LayerName:=", "Top",
        "lw:=", "0mm",
```

```
"x:=", "29mm",  
"y:=", "9.5mm",  
"w:=", "24mm",  
"h:=", "15mm",  
"ang:=", "0deg"))
```

CreateText (Layout Editor)

Use: Creates a text primitive object and adds it to the current layout.

Syntax: **CreateText** <text_description>

Return Value: Returns the name of the newly created object.

Parameters: <text_description>:

```
Array("NAME:Contents",  
      "textGeometry:=", <text_geometry>)
```

<text_geometry> :

```
Array("LayerName:=", <layer_name>,  
      "x:=", <value>, // origin X  
      "y:=", <value>, // origin Y  
      "ang:=", <value>, // rotation  
      "isPlot:=", <bool>, // is it plotted or not  
      "font:=", <font_name>,  
      "size:=", <value>,
```



```
"weight:=", <text_weight>,  
"just:=", <text_justification>,  
"mirror:=", <bool>,  
"text:=", <quoted_string>) // text itself
```

<bool> : *true* | *false*

<font_name> : quoted string, a font name

<text_weight> : integer

```
Thin = 0,  
ExtraLight = 1  
Light = 2  
Normal = 3  
Medium = 4  
SemiBold = 5  
Bold = 6  
ExtraBold = 7  
Heavy = 8
```

<text_justification> : integer

LeftTop = 0

LeftBase = 1

LeftBottom = 2

CenterTop = 3

CenterBase= 4

CenterBottom = 5

RightTop = 6

RightBase = 7

RightBottom = 8

VB Example: oEditor.CreateText

Array("NAME:Contents",

"textGeometry:=",

Array("Layer:=", 6,

"Name:=", "text_6",

"LayerName:=", "Top",

"x:=", "-26mm",

"y:=", "-16mm",

"ang:=", "0deg",

"isPlot:=", true,

"font:=", "Roman",

```
"size:=", "0.508mm",  
"weight:=", 3,  
"just:=", 4,  
"mirror:=", false,  
"text:=", "Sample"))
```

Create Voids in Primitives

The following primitives are available.

[CreateCircleVoid](#)

[CreateLineVoid](#)

[CreatePolygonVoid](#)

[CreateRectangleVoid](#)

[GetPolygonVoids](#)

CreateCircleVoid (Layout Editor)

Use: Creates a circle void and adds it to a particular parent primitive.

Syntax: **CreateCircleVoid**<circle_void_description>

Return Value: Returns the name of the newly created object

Parameters: **<circle_void_description>:**

```
Array("NAME:Contents",
```

```
"owner:=", <object_name>, // parent primitive name
"circle voidGeometry:=", <circle_geometry>) // definition
VB Example:
oEditor.CreateCircleVoid
Array("NAME:Contents",
"owner:=", "rect_4",
"circle voidGeometry:=",
Array ("Layer:=", 6,
"Name:=", "circle void_10",
"LayerName:=", "Top",
"lw:=", "0mm",
"x:=", "26mm",
"y:=", "11mm",
"r:=", "1.41421356237309mm"))
```

CreateLineVoid (Layout Editor)

Use: Creates a line void and adds it to a specified as parameter parent primitive.

Syntax: **CreateLineVoid**<line_void_description>

Return Value: Returns the name of the newly created object.

Parameters: <line_void_description>:

```
Array("NAME:Contents",
```

```
"owner:=", <object_name>, // parent primitive name
"line voidGeometry:=", <line_geometry>) // definition
VB Example: oEditor.CreateLineVoid
Array("NAME:Contents",
"owner:=", "rect_4",
"line voidGeometry:=",
Array("Layer:=", 6,
"Name:=", "line void_14",
"LayerName:=", "Top",
"lw:=", "2mil",
"endstyle:=", 0,
"joinstyle:=", 0,
"n:=", 3,
"x0:=", "27mm", "y0:=", "5mm",
"x1:=", "35mm", "y1:=", "5mm",
"x2:=", "36mm", "y2:=", "9mm"))
```

CreatePolygonVoid (Layout Editor)

Use: Creates a polygon void and adds it to a specified as parameter parent primitive.

Syntax: **CreatePolygonVoid** <polygon_void_description>

Return Value: Returns the name of the newly created object.

Parameters: <polygon_void_description>:

```
Array("NAME:Contents",  
      "owner:=", <object_name>, // owner name  
      "poly voidGeometry:=", <polygon_geometry>) // definition
```

VB Example: oEditor.CreatePolygonVoid

```
Array("NAME:Contents",  
      "owner:=", "rect_4",  
      "poly voidGeometry:=",  
      Array("Layer:=", 6,  
            "Name:=", "poly void_18",  
            "LayerName:=", "Top",  
            "lw:=", "0mm",  
            "n:=", 5,  
            "x0:=", "21mm", "y0:=", "9mm",  
            "x1:=", "21mm", "y1:=", "5mm",  
            "x2:=", "24mm", "y2:=", "7mm",  
            "x3:=", "24mm", "y3:=", "7mm",  
            "x4:=", "24mm", "y4:=", "7mm")) )
```

CreateRectangleVoid (Layout Editor)

Use: Creates a rectangle void and adds it to a specified as parameter parent primitive.

Syntax: **CreateRectangleVoid**<rectangle_void_description>

Return Value: Returns the name of the newly created object.

Syntax: **<rectangle_void_description>:**

```
Array("NAME:Contents",  
      "owner:=", <object_name>, // owner_name  
      "rect voidGeometry:=", <rectangle_geometry>)
```

VB Example:

```
oEditor.CreateRectangleVoid  
Array("NAME:Contents",  
      "owner:=", "rect_4",  
      "rect voidGeometry:=",  
      Array("Layer:=", 6,  
            "Name:=", "rect void_16",  
            "LayerName:=", "Top",  
            "lw:=", "0mm",  
            "x:=", "34.5mm",  
            "y:=", "13mm",  
            "w:=", "3mm",
```

```
"h:=", "2mm",  
"ang:=", "0deg"))
```

GetPolygonVoids (Layout Editor)

Returns the names of the voids inside the selected polygon.

UI Access	N/A.		
Parameters	Name	Type	Description
	<PolyName>	String	target name of a polygon in the active design.
Return Value	Array of the names of the polygon voids inside the target polygon, separated by commas, or None if no voids exist.		

Python Syntax	GetPolygonVoids (<PolyName>)		
Python Example	<pre>oEditor.GetPolygonVoids('rect_0') ['circle void_1', 'circle void_2', 'circle_void 3']</pre>		

VB Syntax	GetPolygonVoids <PolyName>		
VB Example	<pre>oEditor.GetPolygonVoids 'rect_0' 'circle void_1', 'circle void_2', 'circle_void 3'</pre>		

Other Creation Methods

The following creation methods are available.

[AddPinGroupRefPort](#)

[CreateComponent](#)

[CreateCurrentSourcePort](#)

[CreateHole](#)

[CreateMeasure](#)

[CreatePin](#)

[CreatePinGroup](#)

[CreatePinGroupPort](#)

[CreatePinGroups](#)

[CreateTrace](#)

[CreateVia](#)

[CreateViaGroups](#)

[CreateVoltageProbePort](#)

[CreateVoltageSourcePort](#)

[DeletePinGroup](#)

AddPinGroupRefPort (Layout Editor)

Assigns a reference port to a pin group.

Note:

AddPinGroupRefPort is also called when the command "CreatePinGroupPort (Layout Editor) " on page 27-61 is used.

UI Access	From Layout , select Draw Ports/Sources .		
Parameters	Name	Type	Description
	<PinGroup>	Array	Array consisting of the name of a pingroup that requires a reference port (e.g., PinGroup_2).
	<PortRef>	Array	Array consisting of the name of a pingroup that acts as reference port (e.g., PinGroup_1).
Return Value	None		

Python Syntax	AddPinGroupRefPort ([<PinGroup>], [<PortRef>])
Python Example	<code>oEditor.AddPinGroupRefPort(["PinGroup_2"], ["PinGroup_1"])</code>

VB Syntax	AddPinGroupRefPort (<PinGroup>), (<PortRef>)
VB Example	<code>oEditor.AddPinGroupRefPort Array("PinGroup_2"), Array("PinGroup_1")</code>

CreateComponent (Layout Editor)

Use: Places a component.

Syntax: **CreateComponent** <Array>

Return Value: Returns the name of the newly created component.

Parameters: Array("NAME:Contents",
 "definition_name:=", <component name>,
 "placement:=", Array("x:=", <x position>, "y:=", <y position>),
 "layer:=", <placement layer name>,

```
"StackupLayers:=", Array("<footprint stackup layer>:<design stackup layer>", ...),  
"DrawLayers:=", Array("<footprint layer>:<design layer>", ...))  
VB Example: oEditor.CreateComponent Array("NAME:Contents",  
"definition_name:=", "MSTRL",  
"placement:=", Array("x:=", "-13mm", "y:=", "5mm"),  
"layer:=", "Top",  
"StackupLayers:=", Array("Top:Top", "0:Dielectric", "0:Ground"),  
"DrawLayers:=", Array("Measures:Measures", "Assembly Top:Assembly Top", "Silkscreen Top:Silk-  
screen Top", "Wirebonds:0"))
```

Notes:

Each Layer mapping is specified as a single text string in quotes, "<footprint layer>:<design layer>". If the layer does not map to anything, use a "0" for the layer. For example, you can use "Measures:0", if the footprint "Measures" layer does not map to a design layer. Use "0:Dielectric", if the design "Dielectric" layer does not map to a footprint layer.

Note, also, that the "StackupLayers" and the "DrawLayers" arguments may be omitted, in which case, the mappings are determined automatically.

CreateCurrentSourcePort (Layout Editor)

Create a current source port and enable placement on the **Layout Editor**.

UI Access	Draw > Port.

Parameters	Name	Type	Description
	<NAME:Location>	String	Required first parameter which designates the contents of the array (i.e., the Location of the new port).
	<PosLayer>	String	Name of the stackup layer on which to place the positive end of the new port (e.g., Trace).
	<X0>	Integer	X coordinate of the positive end of the new port.
	<Y0>	Integer	Y coordinate of the positive end of the new port.
	<NegLayer>	String	Name of the stackup layer on which to place the negative end of the new port.
	<X0>	Integer	X coordinate of the negative end of the new port.
	<Y0>	Integer	Y coordinate of the negative end of the new port.
Return Value	None		

Python Syntax	CreateCurrentSourcePort([<NAME:Location>, <PosLayer>, <X0>, <Y0>, <NegLayer>, <X1>, <Y1>])
Python Example	<pre>oEditor.CreateCurrentSourcePort (["NAME:Location", "PosLayer:=" , "Trace", "X0:=" , -0.011, "Y0:=" , 0.004, "NegLayer:=" , "Trace", "X1:=" , -0.011, "Y1:=" , 0.004])</pre>

VB Syntax	CreateCurrentSourcePort(<NAME:Location>, <PosLayer>, <X0>, <Y0>, <NegLayer>, <X1>, <Y1>)
VB Example	<pre>oEditor.CreateCurrentSourcePort Array("NAME:Location", "PosLayer:=" , "Trace", "X0:=" , -0.011, "Y0:=" , 0.004, "NegLayer:=" , "Trace", "X1:=" , -0.011, "Y1:=" , 0.004)</pre>

CreateHole (Layout Editor)

Use: Places a Hole object.

Syntax: **CreateHole** <Array>

Return Value: Returns the name of the newly created Hole

```
Parameters: Array("NAME:Contents",  
Array("NAME:full_definition",  
"type:=", "hole",  
Array("NAME:Properties",  
"VariableProp:=",  
Array("radius", "D", "", <value>), // radius  
"VariableProp:=",  
Array("sides", "D", "", integer)), //side count  
"from_layer:=", <layer_name>,  
"to_layer:=", <layer_name>),  
"placement:=", <vpoint>, // placement position  
"layer:=", <layer_name>) // placement layer
```

<vpoint> :

```
Array("x:=", <value>, // X coordinate  
"y:=", <value>) // Y coordinate
```

VB Example: oEditor.CreateHole

```
Array("NAME:Contents",  
Array("NAME:full_definition",  
"type:=", "hole",  
Array("NAME:Properties",  
"VariableProp:=",  
Array("radius", "D", "", "0.635mm"),  
"VariableProp:=",  
Array("sides", "D", "", "6"))),  
"from_layer:=", 6,  
"to_layer:=", 6),  
"placement:=", Array("x:=", "-36mm", "y:=", "-9mm"),  
"layer:=", "Top")
```

CreateMeasure (Layout Editor)

Use: Creates a measurement.

Syntax: CreateMeasure

Use: Returns the name of the created object.

```
Parameters: Array("NAME:Contents",  
"MeasurementGeometry:=",  
Array("LayerName:=", <layer_name>, // layer  
"lw:=", <value>, // line width
```

```
"sx:=", <value>, // start X coordinate
"sy:=", <value>, // start Y coordinate
"ex:=", <value>, // end X coordinate
"ey:=", <value>, // end Y coordinate
<text_style>)
```

<text_style> :

```
"name:=", <quoted string>, // its name
"isPlot:=", <bool>,
"font:=", <font_name>,
"size:=", double, // size in current units
"angle:=", <value>,
"weight:=", <text_weight>,
"just:=", <text_justification>,
"mirror:=", <bool>,
"scales:=", <bool>))
```

```
VB Example: oEditor.CreateMeasure
Array("NAME:Contents",
"MeasurementGeometry:=",
Array("Layer:=", 0,
```

```
"Name:=", "Measurement_2",  
"LayerName:=", "Measures",  
"lw:=", "0mm",  
"sx:=", "-32mm",  
"sy:=", "-13mm",  
"ex:=", "32mm",  
"ey:=", "-11mm",  
"name:=", "<DefaultAnnotation>",  
"isPlot:=", false,  
"font:=", "Arial",  
"size:=", 10,  
"angle:=", "0deg",  
"weight:=", 3,  
"just:=", 4,  
"mirror:=", false,  
"scales:=", false))
```

CreatePin (Layout Editor)

Use: Creates a pin.

Syntax: **CreatePin** <pin_description>

Return Value: Returns the name of the newly created pin.


```
Parameters: <pin_description>:
    Array("NAME:Contents",
    Array("NAME:Port", "Name:=", <object_name>), // not used
    "Rotation:=", Array(<value>), // rotation
    "Offset:=", <vpoint>, // position
    "Padstack:=", <quoted_string>) // padstack name
```

```
VB Example: oEditor.CreatePin
Array("NAME:Contents",
Array("NAME:Port", "Name:=", "Pin_1"),
"Rotation:=", Array("0deg"),
"Offset:=", Array("x:=", "-2mm", "y:=", "-1mm"),
"Padstack:=", "NoPad SMT East")
```

CreatePinGroup (Layout Editor)

Use: Create a pin group from a selection of pins.

Syntax: CreatePinGroup(**STRING** *pin-group-name*, [**"NAME:elements"**, **STRING** *pin*, **STRING** *pin*, ...])

```
VB Example: oEditor.CreatePinGroup("J1_GND_Group", [ "NAME:elements", "J1-1", "J1-2", "J1-35",
"J1-56" ])
```

CreatePinGroupPort (Layout Editor)

Use: Create a pin group port from two pin-groups; first is positive, second is negative.

Syntax: **CreatePinGroupPort**(["Name:elements", **STRING** *pos-pin-group*, **STRING** *neg-pin-group*])

VB Example: oEditor.CreatePinGroupPort(["NAME:elements", "J1_GND_Group", "J1_VCC_Group"])

CreatePinGroups (Layout Editor)

Creates pin groups from a selection of pins.

UI Access	After selecting two or more pins, right-click within the Layout Editor and select Port > Create Pin Group .		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array containing these parameters in order: • <NAME:PinGroupDatas> • <PinGroup>
	<NAME:PinGroupDatas>	String	Required first parameter which must have the specified value (i.e., "PinGroupDatas").
	<PinGroup>	Array	Structured array containing these parameters in order, followed by <PinGroup2>, <PinGroup3>, etcetera: • <NAME:> • <PinNames>
	<NAME:>	String	Required first parameter which must have the name of the pin group (e.g., "PinGroup_3").
	<PinNames>	String	Names of the pins to group, separated by commas.
Return Value	None.		

Python Syntax	CreatePinGroups([<NAME:PinGroupDatas>, [<NAME:>, <PinNames>]])
Python Example	<pre>oEditor.CreatePinGroups(["NAME:PinGroupDatas", ["NAME:PinGroup_3", "via_2", "via_1"]])</pre>

VB Syntax	CreatePinGroups(<NAME:PinGroupDatas>, (<NAME:>, <PinNames>))
VB Example	<pre>oEditor.CreatePinGroups("NAME:PinGroupDatas", ("NAME:PinGroup_3", "via_2", "via_1"))</pre>

CreateTrace (Layout Editor)

Use: Create a trace with a manually specified path between pins, ports, or selected edges.

Command: Draw > Route > Manual

Syntax: CreateTrace

```
Array("NAME:options", "Layer1:=", <"layer">, "Layer2:=", <"layer">),
```

```
Array("NAME:Lines", "Num:=", <n_lines>, "l0:=",
```

```
Array("Name:=", <"line">, "LayerName:=", <"layer">, "lw:=", <"width">,
```

```

"endstyle:=", <endstyle>, "joinstyle:=", <joinstyle>, "n:=", <n_vertex>,
"U:=", <"units">, "x:=", <value>, "y:=", <value>, ), ...),
Array("NAME:Vias", "Num:=", <n_vias>,
Array("NAME:v1", "name:=", <"via">, "ReferencedPadstack:=", <"padstack">,
"vposition:=", Array("x:=", <"vertex">, "y:=", <"vertex">), "vrotation:=",
Array(<"angle">), "overrides hole:=", <override>, "hole diameter:=",
Array(<"diameter">), "highest_layer:=", <"layer">, "lowest_layer:=", <"layer">), ...),
Array("NAME:elements", <"port">, ...),
Array("NAME:EdgeRefs", "edge:=", Array(<edge description>), ...)

```

Return Value: None

Parameters: <"layer">

Type: text

Description: layer name

<n_lines>

Type: integer

Description: number of line definitions following

<"line">

Type: text

Description: line name

<"width">

Type: text

Description: line width; value with units, e.g. "1mm"

<endstyle>

Type: integer

Description: end (cap) style value for the line.

<joinstyle>

Type: integer

Description: join style value.

<n_vertex>

Type: integer

Description: number of vertices in the line

<value>

Type: double

Description: simple value, e.g. 10

<n_vias>

Type: integer

Description: number of via definitions following

<"via">

Type: text

Description: via name

<"padstack">

Type: text

Description: padstack definition name.

<"angle">

Type: text

Description: via orientation; value with units, e.g. "180deg"

<override>

Type: boolean (true or false)

Description: if true, the diameter is used to override the via hole definition.

<"diameter">

Type: text

Description: via hole diameter override; a value with units, e.g. "1mm"

<"port">

Type: text

Description: a port or pin name.

<edge description>

for primitive edges

"et:=", "pe", "prim:=", <"prim">, "edge:=", <edge#>

<"prim">

Type: text

Description: primitive name

<edge#>

Type: integer

Description: 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)


```
VB Example: oEditor.CreateTrace Array("NAME:options", "Layer1:=", "Top",
"Layer2:=", "Ground"), Array("NAME:elements"), Array("NAME:EdgeRefs",
"edge:=", Array("et:=", "pe", "prim:=", "rect_4", "edge:=", 3), "edge:=",
Array("et:=", "pse", "sel:=", "via_3", "layer:=", 8, "sx:=", -0.0015, "sy:=", 0.0015,
"ex:=", -0.0015, "ey:=", -0.0015, "h:=", 0, "rad:=", 0))
```

CreateVia (Layout Editor)

Use: Creates a new via.

Syntax: **CreateVia** <via_description>

Return Value: Returns the name of the created via

Parameters: <via_description>:

```
Array("NAME:Contents",
"name:=", <object_name>, // not used
"vposition:=", <vpoint>, // position
"rotation:=", double, // rotation in radians
"overrides hole:=", <bool>, // overrides or not padstack hole
"hole diameter:=", Array(<value>), // via diameter
"ReferencedPadstack:=", <quoted_string>, // padstack name
"highest_layer:=", <layer_name>,
"lowest_layer:=", <layer_name>)
```

```

VB Example: oEditor.CreateVia
Array("NAME:Contents",
"name:=", "",
"vposition:=", Array("x:=", "-22mm", "y:=", "5mm"),
"rotation:=", 0,
"overrides hole:=", false,
"hole diameter:=", Array("1mm"),
"ReferencedPadstack:=", "Round 1mm/0.5mm",
"highest_layer:=", "Top",
"lowest_layer:=", "Bottom")

```

CreateViaGroups (Layout Editor)

Creates via groups on the target layer.

UI Access	After selecting two or more vias, navigate to Draw > Via Groups > Create > Persistent/Non-Persistent .		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array containing these parameters in order: • <NAME:vias> • <TargetObjects> • <GroupName> • <Enable>
	<NAME:vias>	String	Required first parameter which must have the specified value (i.e., "vias").

	<TargetObjects>	String	Names of the target primitives to add to a via group, separated by commas.
	<GroupName>	String	Names of the new via group.
	<Enable>	Boolean	True creates persistent via group. False creates non-persistent via group.
Return Value	None.		

Python Syntax	CreateViaGroups([<NAME:vias>, <TargetObjects>, <GroupName>], <Enable>)
Python Example	<pre>oEditor.CreateViaGroups(["NAME:vias", "circle_8651", "rect_825", "ViaGroup_1"], False)</pre>

VB Syntax	CreateViaGroups(<NAME:vias>, <TargetObjects>, <GroupName>), <Enable>
VB Example	<pre>oEditor.CreateViaGroups Array("NAME:vias", "circle_8651", "rect_825", "ViaGroup_1"), False</pre>

CreateVoltageProbePort (Layout Editor)

Create a voltage probe port and enable placement on the **Layout Editor**.

UI Access	Draw > Port.		
Parameters	Name	Type	Description
	<NAME:Location>	String	Required first parameter which designates the contents of the array (i.e., the Location of the new port).
	<PosLayer>	String	Name of the stackup layer on which to place the positive end of the new port (e.g., Trace).
	<X0>	Integer	X coordinate of the positive end of the new port.
	<Y0>	Integer	Y coordinate of the positive end of the new port.
	<NegLayer>	String	Name of the stackup layer on which to place the negative end of the new port.
	<X0>	Integer	X coordinate of the negative end of the new port.
	<Y0>	Integer	Y coordinate of the negative end of the new port.
Return Value	String PortID, the identifier that can be used for this port in script operations involving this schematic		

Python Syntax	CreateVoltageProbePort([<NAME:Location>, <PosLayer>, <X0>, <Y0>, <NegLayer>, <X1>, <Y1>])
Python Example	<pre>oEditor.CreateVoltageProbePort (["NAME:Location", "PosLayer:=" , "Trace", "X0:=" , -0.011, "Y0:=" , 0.004, "NegLayer:=" , "Trace", "X1:=" , -0.011, "Y1:=" , 0.004])</pre>

VB Syntax	CreateVoltageProbePort Array(<NAME:Location>, <PosLayer>, <X0>, <Y0>, <NegLayer>, <X1>, <Y1>)
VB Example	oEditor.CreateVoltageProbePort

	<pre>Array("NAME:Location", "PosLayer:=" , "Trace", "X0:=" , -0.011, "Y0:=" , 0.004, "NegLayer:=" , "Trace", "X1:=" , -0.011, "Y1:=" , 0.004)</pre>

CreateVoltageSourcePort (Layout Editor)

Create a voltage source port and enable placement on the **Layout Editor**.

UI Access	Draw > Port.		
Parameters	Name	Type	Description
	<NAME:Location>	String	Required first parameter which designates the contents of the array (i.e., the Location of the new port).
	<PosLayer>	String	Name of the stackup layer on which to place the positive end of the new port (e.g., Trace).
	<X0>	Integer	X coordinate of the positive end of the new port.
	<Y0>	Integer	Y coordinate of the positive end of the new port.
	<NegLayer>	String	Name of the stackup layer on which to place the negative end of the new port.
	<X0>	Integer	X coordinate of the negative end of the new port.
	<Y0>	Integer	Y coordinate of the negative end of the new port.
Return Value	None		

Python Syntax	CreateVoltageSourcePort([<NAME:Location>, <PosLayer>, <X0>, <Y0>, <NegLayer>, <X1>, <Y1>])
Python Example	<pre>oEditor.CreateVoltageSourcePort (["NAME:Location", "PosLayer:=" , "Trace", "X0:=" , -0.011, "Y0:=" , 0.004, "NegLayer:=" , "Trace", "X1:=" , -0.011, "Y1:=" , 0.004)</pre>

VB Syntax	CreateVoltageSourcePort(<NAME:Location>, <PosLayer>, <X0>, <Y0>, <NegLayer>, <X1>, <Y1>)
VB Example	<pre>oEditor.VoltageSourcePort Array("NAME:GlobalPortProps", "Name:=", "G_0", "Id:=", 1),_ Array("NAME:Attributes","Page:=", 1, "X:=", -0.02286, "Y:=", 0.00254, "Angle:=", 0, "Flip:=", false)</pre>

DeletePinGroup (Layout Editor)

Use: Deletes a pin group.

Syntax: **DeletePinGroup**(**STRING** *pin-group-name*)

VB Example: oEditor.DeletePinGroup("J1_GND_Group")

Object Movement and Modification Methods

The documented object movement methods are available.

[Connect \(Layout Editor\)](#)

[Copy \(Layout Editor\)](#)

[Delete \(Layout Editor\)](#)

[Disconnect \(Layout Editor\)](#)

[Edit \(Layout Editor\)](#)

[Expand \(Layout Editor\)](#)

[ExpandWithPorts \(Layout Editor\)](#)

[FlipHorizontal \(Layout Editor\)](#)

[FlipVertical \(Layout Editor\)](#)

[Move \(Layout Editor\)](#)

[Paste \(Layout Editor\)](#)

[Rotate \(Layout Editor\)](#)

Connect (Layout Editor)

Use: Causes connecting of the referenced object. After the command the objects will be in the same electrical net.

Syntax: **Connect** Array ("NAME: *elements*",

<object_name>, // 1st object

<object_name>, // 2nd object, if any

...) // etc

VB Example:

```
oEditor.Connect Array("NAME:elements", "2:n2", "circle_4")
```

Copy (Layout Editor)

Use: Copies the referenced objects to the clipboard.

Syntax: **Copy** Array (<object_name>, // 1st object

<object_name>, // 2nd object

Return Value: Returns the name of the new net.

VB Example:

```
oEditor.Copy Array("circle_0", "rect_2")
```

Delete (Layout Editor)

Use: Deletes one or more objects.

Syntax: **Delete** <object_name_list>// names of the objects to be deleted

<object_name_list>:

```
Array ("NAME:elements", <object_name>, <object_name>, ...)
```

VB Example:

```
oEditor.Delete Array("NAME:elements", "circle_0", "rect_2")
```

Disconnect (Layout Editor)

Use: Removes the specified object(s) from their current electrical net(s) (if any).

Syntax: **Disconnect** Array ("NAME: elements",

<object_name>, // 1st object

<object_name>, // 2nd object, if any

...) // etc

VB Example: oEditor.Disonnect Array("NAME:elements", "2:n2")

Edit (Layout Editor)

Use: Causes modification of one or more existing object(s).

Syntax: **Edit** <Array("NAME:items".....)>,

Parameters: <Array("NAME:items"
<edit_object_info>, // one object info
<edit_object_info>, // another one
...) // etc

<edit_object_info>:

Array("NAME:item",
"name:=", <object_name>, // name of the object
<object_description>) // 'new' object state, type should be consistent with <object_name>:

<object_description>:

<circle_description> |

<rectangle_description> |

<line_description> |

<polyon_description> |

<text_description> |

<circle_void_description> |

<rectangle_void_description> |

<line_void_description> |

<polyon_void_description> |

<component_description> |

```
<pin_description> |
```

```
<via_description>
```

VB Example:

```
oEditor.Edit Array("NAME:items", Array("NAME:item",
"name:=", "circle_0", Array("NAME:contents",
"circleGeometry:=", Array("Layer:=", 6,
"Name:=", "circle_0", "LayerName:=", "Top",
"lw:=", "0mm", "x:=", "-0.008meter",
"y:=", "13.2924281984334mm", "r:=", 10.2924281984334mm)))
```

Expand (Layout Editor)

Scales an existing circle, square, or polygon. To expand an object while preserving ports, use [ExpandWithPorts](#).

UI Access	Draw > Expand.		
Parameters	Name	Type	Description
	<expansion>	String	Expansion amount, including unit. Value can be negative to shrink an object.
	<join>	String	Corner style. One of: "ROUND" "MITER" "CORNER"
	<replace>	Boolean	True to replace original object; False to create new object.
	<items>	Array	Structured array containing object to expand: <pre>Array("NAME:elements",</pre>

			"<Name of Object to Expand>")
Return Value	None.		

Python Syntax	Expand (<expansion>, <join>, <replace>, <items>)		
Python Example	oEditor.Expand("2mm", "ROUND", True, ["NAME:elements", "circle_0"])		

VB Syntax	Expand <expansion>, <join>, <replace>, <items>		
VB Example	oEditor.Expand "2mm", "ROUND", true, Array("NAME:elements", "poly_0")		

ExpandWithPorts (Layout Editor)

Scales an existing circle, square, or polygon and preserves ports. To expand an object without preserving ports, use [Expand](#).

UI Access	Draw > Expand.		
Parameters	Name	Type	Description
	<expansion>	String	Expansion amount, including unit. Value can be negative to shrink an object.
	<join>	String	Corner style. One of: "ROUND" "MITER" "CORNER"
	<replace>	Boolean	True to replace original object; False to create new object.
	<items>	Array	Structured array containing object to expand:

			Array("NAME:elements", "<Name of Object to Expand>")
Return Value	None.		

Python Syntax	ExpandWithPorts (<expansion>, <join>, <replace>, <items>)
Python Example	oEditor.ExpandWithPorts("2mm", "ROUND", True, ["NAME:elements", "circle_0"])

VB Syntax	ExpandWithPorts <expansion>, <join>, <replace>, <items>
VB Example	oEditor.ExpandWithPorts "2mm", "ROUND", true, Array("NAME:elements", "poly_0")

FlipHorizontal (Layout Editor)

Use: Causes horizontal flipping of one or more specified objects (mirroring about a vertical axis given its X coordinate).

Syntax: **FlipHorizontal** <object_name_list>, //objects to be flipped <position_or_center> // (X of flip_axis / not used)

VB Example:

```
oEditor.FlipHorizontal Array("NAME:elements", "circle_0", "rect_2"),  
Array(0.01, -0.001)
```

FlipVertical (Layout Editor)

Use: Causes vertical flipping of one or more specified objects (mirroring about a horizontal axis given its Y coordinate).

Syntax: **FlipVertical** <object_name_list>, //objects to be flipped <position_or_center> // not used, flip axis

VB Example: oEditor.FlipVertical Array("NAME:elements", "circle_0", "rect_2"), Array(0.01, -0.001)

Move (Layout Editor)

Use: Translate this Polygon by the vector specified in the provided Point. Return this Polygon.

Syntax: oPolygon.Move(<oPoint>)

Return Value: This Polygon object.

VB Example: oPolygon = oPolygon.Move(oLayout.Point().Set(1,1))

Paste (Layout Editor)

Use: Pastes the objects currently in clipboard; the argument specifies the offset of the new (copy) object(s) from their initial position(s).

Syntax: **Paste** Array("NAME:offset",

"xy:=",

Array(double, double)) // X/Y offset

VB Example: oEditor.Paste Array("NAME:offset", "xy:=", Array(0.034, 0.013))

Rotate (Layout Editor)

Use: Rotate this Polygon counter clockwise (CCW) about a center specified by the provided Point. Angle is in radians. Return this Polygon.

Syntax: oPolygon.Rotate(<AngRad>, <oPoint>)

Return Value: This Polygon object.

VB Example: `oPolygon = oPolygon.Rotate(pi/2, oLayout.Point())`

Activation and Deactivation Methods

The following activation/deactivation methods are available.

[Activate](#)

[DeactivateOpen](#)

[DeactivateShort](#)

[Delete](#)

Activate (Layout Editor)

Use: Causes activation of one or more components.

Syntax: `Activate Array("NAME:components",
<object_name>, // 1st component name
<object_name>, // 2nd component, if any
...) // etc`

VB Example:

```
oEditor.Activate Array("NAME:components", "2", "3")
```

DeactivateOpen (Layout Editor)

Use: Causes deactivation of one or more components to *open circuit* state.

Syntax: **ActivateOpen** Array ("NAME:components",

```
<object_name>, // 1st component name  
<object_name>, // 2nd component, if any  
...)// etc
```

VB Example:

```
oEditor.DeactivateShort Array("NAME:components", "2", "3")
```

DeactivateShort (Layout Editor)

Use: Causes deactivation of one or more components to *short circuit* state.

Syntax: **ActivateShort** Array("NAME:components",

```
<object_name>, // 1st component name  
<object_name>, // 2nd component, if any  
...)// etc
```

VB Example:

```
oEditor.DeactivateShort Array("NAME:components", "2", "3")
```

Delete (Layout Editor)

Use: Deletes one or more objects.

Syntax: **Delete** <object_name_list>// names of the objects to be deleted

```
<object_name_list>:  
Array ("NAME:elements", <object_name>, <object_name>, ...)
```

VB Example:

```
oEditor.Delete Array("NAME:elements", "circle_0", "rect_2")
```

Layout and Geometry Interrogation Methods

From the Layout scripting object, the following methods are available for layout and geometry interrogation.

The topics for this section include:

[Layout Interrogation](#)

[Point Object](#)

[Polygon Object](#)

Layout Interrogation (Layout Editor)

Methods employing Point and Polygon objects are specified in SI units (e.g. meters).

The following methods are available.

[CreateLineFromPolygon \(Layout Editor\)](#)

[CreateObjectFromPolygon \(Layout Editor\)](#)

[FindObjects \(Layout Editor\)](#)

[FilterObjectList \(Layout Editor\)](#)

[FindObjectsByPoint \(Layout Editor\)](#)

[FindObjectsByPolygon \(Layout Editor\)](#)

[GetBBox \(Layout Editor\)](#)

[GetCSObjects \(Layout Editor\)](#)

[GetPolygon \(Layout Editor\)](#)

[GetPolygonDef \(Layout Editor\)](#)

[Point \(Layout Editor\)](#)

[Polygon \(Layout Editor\)](#)

CreateLineFromPolygon (Layout Editor)

Use: Create a line/trace object on the specified layer and net from the provided Polygon object. Net is optional.

Syntax: oLayout.CreateLineFromPolygon(<oPolygon>, <width>, <bendType>, <startCapType>, <endCapType>, <Layer>, <Net>)

Parameters: width - line width, in meters

bendType - {'corner', 'round'}

startCapType, endCapType - {'flat', 'extended', 'round'}

VB Example:

```
newobj = oLayout.CreateLineFromPolygon(oPolygon, 0.1e-3, 'round', 'flat', 'round', 'L1', 'DQ0')
```

CreateObjectFromPolygon (Layout Editor)

Use: Create a polygon on the specified layer and net from the provided Polygon object. Return the name of the new object. Net is optional.

Syntax: oLayout.CreateObjectFromPolygon(<oPolygon>, <Layer>, <Net>)

Return Value: Object name.

VB Example:

```
newobj = oLayout.CreateObjectFromPolygon(oPolygon, 'L1', 'GND')
```

FindObjects (Layout Editor)

Returns a list of object names using a key/value pair.

UI Access	N/A.		
Parameters	Name	Type	Description
	<key>	String	One of: • "Name" to search by object name. • "Type" to search by object type. Valid <value> strings for this type include: 'pin', 'via', 'rect', 'arc', 'line', 'poly', 'plg', 'circle void', 'line void', 'rect void', 'poly void', 'plg void', 'text', 'cell', 'Measurement', 'Port', 'Port Instance', 'Port Instance Port', 'Edge Port', 'component', 'CS', 'S3D', 'ViaGroup' • "Layer" to search by layer. An asterisk (*) in the <value> string matches all layers. Vias and pins match using their defined layer range. 'Multi' matches (non via/pin) multi-layer objects. • "Net" to search by net assignment.
	<value>	String	Specific query. Queries use a basic string match. For instance, an asterisk (*) matches anything. All matching is case insensitive.
Return Value	List of object names.		

Python Syntax	FindObjects (<key>, <value>)
Python Example	oLayout.FindObjects ('Type', 'Via')

VB Syntax	FindObjects <key>, <value>
VB Example	Set vias = oLayout.FindObjects "Type", "Via"

FilterObjectList (Layout Editor)

Filters a list of object names using a key/value pair. See: [FindObjects](#).

UI Access	N/A.		
Parameters	Name	Type	Description
	<key>	String	One of: • "Name" to search by object name. • "Type" to search by object type. Valid <value> strings for this type include: 'pin', 'via', 'rect', 'arc', 'line', 'poly', 'plg', 'circle void', 'line void', 'rect void', 'poly void', 'plg void', 'text', 'cell', 'Measurement', 'Port', 'Port Instance', 'Port Instance Port', 'Edge Port', 'component', 'CS', 'S3D', 'ViaGroup' • "Layer" to search by layer. An asterisk (*) in the <value> string matches all layers. Vias and pins match using their defined layer range. 'Multi' matches (non via/pin) multi-layer objects. • "Net" to search by net assignment.
	<value>	String	Specific query. Queries use a basic string match. For instance, an asterisk (*) matches anything. All matching is case insensitive.
Return Value	Filtered list of object names.		

Python Syntax	FilterObjectList (<key>, <value>)
Python Example	<pre>RFIN_pins = oLayout.FilterObjectList('Type', 'Pin', oLayout.FindObjects('Net','RFIN'))</pre>

VB Syntax	FilterObjectList <key>, <value>
VB Example	<pre>Set objectlist = oEditor.FindObjects("Net", "RFIN") Set RFIN_pins = oEditor.FilterObjectList("Type", "Pin", objectlist)</pre>

FindObjectsByPoint (Layout Editor)

Use: Get a list of all objects intersected by the specified Point object on a given layer.

Syntax: oLayout.FindObjectsByPoint(<oPoint>, <Layer>)

Return Value: Object list.

VB Example:

```
objs = oLayout.FindObjectsByPoint(oLayout.Point().Set(-1.149e-3, 3.465e-3), 'L3')
```

FindObjectsByPolygon (Layout Editor)

Use: Get a list of all objects intersecting the specified Polygon object on the given layer.

Syntax: oLayout.FindObjectsByPolygon(<oPolygon>, <Layer>)

Return Value: Object list.

VB Example:

```
# Find all objects intersecting box = {(0,0), (1e-3,1e-3)}
```

```
p0 = oLayout.Point().Set(0, 0)
p1 = oLayout.Point().Set(1e-3, 0)
p2 = oLayout.Point().Set(1e-3, 1e-3)
p3 = oLayout.Point().Set(0, 1e-3)
box = oLayout.Polygon().AddPoint(p0).AddPoint(p1).AddPoint(p2).AddPoint(p3).SetClosed(True)
objs = oLayout.FindObjectsByPolygon(box, '*')
```

GetBBox (Layout Editor)

Use: Get a [Polygon Object](#) defining the bounding box for the specified object name, or None if the object does not exist.

Syntax: oLayout.GetBBox(<Name>)

Return Value: Polygon object.

VB Example: oPolygon = oLayout.GetBBox('rect_10')

GetCSObjects (Layout Editor)

Use: Given a coordinate system name, returns a list of the objects inside. Coordinate system names can be found using FindObjects ('Type', 'CS'). See [Find Objects](#).

Syntax: oLayout.GetCSObjects(<CS_name>)

Return Value: List of objects in the coordinate system.

VB Example: CS_objects = oLayout.GetCSObjects("CS_1")

GetPolygon (Layout Editor)

Use: Get a [Polygon Object](#) for the specified object name, or None if the object does not exist. Returned polygon is as rendered, in SI units

Syntax: oLayout.GetPolygon(<Name>)

Return Value: Polygon object.

VB Example: oPolygon = oLayout.GetPolygon('line_37')

GetPolygonDef (Layout Editor)

Use: Get a [Polygon Object](#) for the specified object name, or None if the object does not exist. Returned polygon is as rendered, in SI units. For trace types, this corresponds to the center line.

Syntax: oLayout.GetPolygonDef(<Name>)

Return Value: Polygon object.

VB Example: oPolygon = oLayout.GetPolygonDef('line_37')

Point (Layout Editor)

Use: Create and return a [Point Object](#).

Syntax: oLayout.Point()

Return Value: Point object.

VB Example: oPoint = oLayout.Point()

Polygon (Layout Editor)

Use: Create and return a [Polygon Object](#).

Syntax: oLayout.Polygon()

Return Value: Polygon object.

VB Example: oPolygon = oLayout.Polygon()

Point Object (Layout Editor)

The Point object provides a point container in a 2D Cartesian coordinate system. Point objects have no link to specific Layout primitives. All values and interfaces are specified in meters.

A Layout object is used to create a Point as:

```
oPoint = oLayout.Point()
```

The following methods are available.

[Set](#)

[SetX](#)

[GetX](#)

[SetY](#)

[GetY](#)

[SetArc](#)

[IsArc](#)

[IsEqual](#)

[Mag](#)

[Distance](#)

[Cross](#)

[Move](#)

[Rotate](#)

[Normalize](#)

[DistanceFromLine](#)

[ClosestPointOnLine](#)

Set (Layout Editor)

Use: Set both the x and y -coordinates of the Point, in meters. Return *this* Point.

Syntax: oPoint.Set(<x>, <y>)

Return Value: This Point object.

VB Example: oPoint = oPoint.Set(1e-3, 1e-3)

SetX (Layout Editor)

Use: Set the x-coordinate of the Point, in meters. Return *this* Point.

Syntax: oPoint.SetX(<x>)

Return Value: This Point object.

VB Example: oPoint = oPoint.SetX(1e-3)

GetX (Layout Editor)

Use: Get the x-coordinate of the Point, in meters.

Syntax: oPoint.GetX()

Return Value: X-coordinate.

VB Example: x = oPoint.GetX()

SetY (Layout Editor)

Use: Set the y-coordinate of the Point, in meters. Return *this* Point.

Syntax: oPoint.SetY(<y>)

Return Value: This Point object.

VB Example: oPoint = oPoint.SetY(1e-3)

GetY (Layout Editor)

Use: Get the y-coordinate of the Point, in meters.

Syntax: oPoint.GetY()

Return Value: Y-coordinate.

VB Example: y = oPoint.GetY()

SetArc (Layout Editor)

Use: Set whether the given point is an arc. For arc-points, the x-coordinate is used to specify signed height. Return *this* Point.

Syntax: oPoint.SetArc(<isArc>)

Return Value: This Point object.

```
VB Example: oPoint = oPoint.SetArc(True)
```

IsArc (Layout Editor)

Use: Check whether this Polygon is defined by an arc-segment.

Syntax: oPolygon.IsArc()

```
VB Example: isArc = oPolygon.IsArc()
```

IsEqual (Layout Editor)

Use: Test whether this point is equal to another. If tolerance is not specified, a default 1e-9 is used.

Syntax: oPoint.IsEqual(<oPoint>, [<Tolerance>])

VB Example:

```
isEqual1 = oPointThis.IsEqual(oPointOther)
```

```
isEqual2 = oPointThis.IsEqual(oPointOther, 1e-6)
```

Mag (Layout Editor)

Use: Get the magnitude of the given Point

Syntax: oPoint.Mag()

```
VB Example: mag = oPoint.Mag()
```

Distance (Layout Editor)

Use: Get the distance from this point to another.

Syntax: oPointThis.Distance(oPointOther)

VB Example: dist = oPointThis.Distance(oPointOther)

Cross (Layout Editor)

Use: Get the cross product of this point with another. In 2D, this is simply the determinant of [x1 x2; y1 y2].

Syntax: oPointThis.Cross(oPointOther)

VB Example: cross = oPointThis.Cross(oPointOther)

Move (Layout Editor)

Use: Translate this Polygon by the vector specified in the provided Point. Return this Polygon.

Syntax: oPolygon.Move(<oPoint>)

Return Value: This Polygon object.

VB Example: oPolygon = oPolygon.Move(oLayout.Point().Set(1,1))

Rotate (Layout Editor)

Use: Rotate this Polygon counter clockwise (CCW) about a center specified by the provided Point. Angle is in radians. Return this Polygon.

Syntax: oPolygon.Rotate(<AngRad>, <oPoint>)

Return Value: This Polygon object.

VB Example: oPolygon = oPolygon.Rotate(pi/2, oLayout.Point())

Normalize (Layout Editor)

Use: Normalize this point. Return this point.

Syntax: oPoint.Normalize()

Return Value: This Point object.

VB Example: oPoint = oPoint.Normalize()

DistanceFromLine (Layout Editor)

Use: Get the distance from this point to a line defined by two additional points.

Syntax: oPoint.DistanceFromLine(<oPoint>, <oPoint>)

VB Example: dist = oPoint.DistanceFromLine(oLayout.Point().Set(0,0), oLayout.Point().Set(1,0))

ClosestPointOnLine (Layout Editor)

Use: Get the closest point to this on a line defined by the two input points.

Syntax: oPoint.ClosestPointOnLine(<oPoint>, <oPoint>)

Return Value: The closest point, as a Point Object.

VB Example: oPointClosest = oPoint.ClosestPointOnLine(oLayout.Point().Set(0,0), oLayout.Point().Set(1,0))

Polygon Object (Layout Editor)

The Polygon object provides a polygon container in a 2D Cartesian coordinate system. Objects have no link to specific Layout primitives. All values and interfaces are specified in meters.

A Layout object can be used to create an empty, and open, Polygon as:

```
oPolygon = oLayout.Polygon()
```

The following methods are available.

[AddPoint](#)

[SetClosed](#)

[IsClosed](#)

[Move](#)

[Rotate](#)

[Scale](#)

[MirrorX](#)

[GetPoints](#)

[AddHole](#)

[HasHoles](#)

[GetHoles](#)

[HasArcs](#)

[HasSelfIntersections](#)

[BBoxLL](#)

[BBoxUR](#)

[IsParametric](#)

[IsConvex](#)

[IsPoint](#)

[IsSegment](#)

[IsArc](#)

[IsBox](#)

[IsCircle](#)

[GetBoundingCircleCenter](#)

[GetBoundingCircleRadius](#)

[Area](#)

[PointInPolygon](#)

[CircleIntersectsPolygon](#)

[GetIntersectionType](#)

[GetClosestPoint](#)

[GetClosestPoints](#)

[Unite](#)

[Intersect](#)

[Subtract](#)

[Xor](#)

AddPoint (Layout Editor)

Command: Add a point to the polygon. Return this Polygon.

Syntax: oPolygon.AddPoint(<oPoint>)

Return Value: This Polygon object.

VB Example: oPolygon=oPolygon.AddPoint(oLayout.Point().Set(0,1))

SetClosed (Layout Editor)

Use: Set whether a polygon is open or closed. Return this Polygon.

Syntax: oPolygon.SetClosed(isClosed)

Return Value: This Polygon object.

VB Example: oPolygon = oPolygon.SetClosed(True)

IsClosed (Layout Editor)

Use: Get whether a polygon is open or closed.

Syntax: oPolygon.IsClosed()

VB Example: isClosed = oPolygon.IsClosed()

Move (Layout Editor)

Use: Translate this Polygon by the vector specified in the provided Point. Return this Polygon.

Syntax: oPolygon.Move(<oPoint>)

Return Value: This Polygon object.

VB Example: oPolygon = oPolygon.Move(oLayout.Point().Set(1,1))

Rotate (Layout Editor)

Use: Rotate this Polygon counter clockwise (CCW) about a center specified by the provided Point. Angle is in radians. Return this Polygon.

Syntax: oPolygon.Rotate(<AngRad>, <oPoint>)

Return Value: This Polygon object.

VB Example: oPolygon = oPolygon.Rotate(pi/2, oLayout.Point())

Scale (Layout Editor)

Scales the polygon by the specified factor.

UI Access	N/A.		
Parameters	Name	Type	Description
	<factor>	Integer	Scaling factor.
	<centerPoint>	Object	Any oPoint object. See: Point Object .
Return Value	Polygon object.		

Python Syntax	Scale(<factor>, <centerPoint>)
Python Example	oPolygon.Scale(2, oPoint)

VB Syntax	Scale <factor>, <centerPoint>
VB Example	oPolygon.Scale 2, oPoint

MirrorX (Layout Editor)

Use: Mirror this Polygon about the line defined by the provided X-coordinate. Return this Polygon.

Syntax: oPolygon.MirrorX(x-coord)

Return Value: This Polygon object.

VB Example: oPolygon = oPolygon.MirrorX(0.0)

GetPoints (Layout Editor)

Use: Get a list of Point objects defining this Polygon.

Syntax: oPolygon.GetPoints()

Return Value: List of Point objects.

VB Example: ptList = oPolygon.GetPoints()

AddHole (Layout Editor)

Use: Add a hole to a Polygon. The hole is defined by the input Polygon object. Return this Polygon.

Syntax: oPolygon.AddHole(oPolygon)

Return Value: This Polygon object.

VB Example: oPolygon = oPolygon.AddHole(oPolygonHole)

HasHoles (Layout Editor)

Use: Get whether a Polygon has holes.

Syntax: oPolygon.HasHoles()

VB Example: HasHoles = oPolygon.HasHoles()

GetHoles (Layout Editor)

Use: Get a list of holes owned by this Polygon.

Syntax: oPolygon.GetHoles()

Return Value: List of holes as Polygon objects.

VB Example: HoleList = oPolygon.GetHoles()

HasArcs (Layout Editor)

Use: Get whether this Polygon has any arc-segments.

Syntax: oPolygon.HasArcs()

VB Example: HasArcs = oPolygon.HasArcs()

HasSelfIntersections (Layout Editor)

Use: Check whether this Polygon has any self-intersections. Default tolerance is 1e-9, or it may be explicitly provided.

Syntax: oPolygon.HasSelfIntersections(<tolerance>)

VB Example: hasSelfInt1 = oPolygon.HasSelfIntersections() # default tol=1e-9

hasSelfInt2 = oPolygon.HasSelfIntersections(1e-6)

BBoxLL (Layout Editor)

Use: Get the lower-left point for the bounding-box region of this Polygon.

Syntax: oPolygon.BBoxLL()

Return Value: A Point object defining the lower-left coordinate.

VB Example: oPoint = oPolygon.BBoxLL()

BBoxUR (Layout Editor)

Use: Get the upper-right point for the bounding-box region of this Polygon.

Syntax: oPolygon.BBoxUR()

Return Value: A Point object defining the upper-right coordinate.

VB Example: oPoint = oPolygon.BBoxUR()

IsParametric (Layout Editor)

Use: Check whether this Polygon is defined parametrically.

Syntax: oPolygon.IsParametric()

VB Example: isParametric = oPolygon.IsParametric()

IsConvex (Layout Editor)

Use: Check whether this Polygon is convex.

Syntax: oPolygon.IsConvex()

VB Example: isConvex = oPolygon.IsConvex()

IsPoint (Layout Editor)

Use: Check whether this Polygon is defined by a point.

Syntax: oPolygon.IsPoint()

VB Example: isPt = oPolygon.IsPoint()

IsSegment (Layout Editor)

Use: Check whether this Polygon is defined by a segment.

Syntax: oPolygon.IsSegment

VB Example: isSeg = oPolygon.IsSegment

IsArc (Layout Editor)

Use: Check whether this Polygon is defined by an arc-segment.

Syntax: oPolygon.IsArc()

VB Example: isArc = oPolygon.IsArc()

IsBox (Layout Editor)

Use: Check whether this Polygon defines a box.

Syntax: oPolygon.IsBox()

VB Example: isBox = oPolygon.IsBox()

IsCircle (Layout Editor)

Use: Check whether this Polygon defines a circle.

Syntax: oPolygon.IsCircle()

VB Example: isCircle = oPolygon.IsCircle()

GetBoundingCircleCenter (Layout Editor)

Use: Get a Point defining the center of a bounding circle for this Polygon.

Syntax: oPolygon.GetBoundingCircleCenter()

Return Value: A Point object defining the bounding circle's center coordinate.

VB Example: oPoint = oPolygon.GetBoundingCircleCenter()

GetBoundingCircleRadius (Layout Editor)

Use: Get the radius of a bounding circle for this Polygon.

Syntax: oPolygon.GetBoundingCircleRadius()

Return Value: The bounding circle's radius.

VB Example: rad = oPolygon.GetBoundingCircleRadius()

Area (Layout Editor)

Use: Get the area of the Polygon, in square meters.

Syntax: oPolygon.Area()

VB Example: `area = oPolygon.Area()`

PointInPolygon (Layout Editor)

Use: Test whether the provided Point is in this Polygon. Points on the Polygon's boundary are considered in the Polygon.

Syntax: `oPolygon.PointInPolygon(<oPoint>)`

VB Example: `hit = oPolygon.PointInPolygon(oPoint)`

CircleIntersectsPolygon (Layout Editor)

Use: Test whether the circle defined by the input Point and radius hit this Polygon. Circles tangent to the Polygon's boundary are considered intersecting.

Syntax: `oPolygon.CircleIntersectsPolygon(<oPointCenter>, <radius>)`

VB Example: `hit = oPolygon.CircleIntersectsPolygon(oPointCenter, 1e-3)`

GetIntersectionType (Layout Editor)

Use: Get the intersection type of this Polygon with another. A tolerance can be optionally specified, or the default 1e-9 will be used.

Syntax: `oPolygon.GetIntersectionType(<oPolygon>)`

Return Value: Integer:

- 0 - No intersection
- 1 - This fully inside other
- 2 - Other fully inside this
- 3 - Common contour points
- 4 - Undefined intersection

```
VB Example: typ1 = oPolygon.GetIntersectionType(oPolygonOther) # default tol=1e-9
            typ2 = oPolygon.GetIntersectionType(oPolygonOther, 1e-6)
```

GetClosestPoint (Layout Editor)

Use: Find the closest Point on this Polygon's boundary to the Point provided.

Syntax: oPolygon.GetClosestPoint(<oPoint>)

Return Value: A Point object defining the closest location on this Polygon's boundary.

```
VB Example: oPointClosest = oPolygon.GetClosestPoint(oLayout.Point().Set(1.0,0))
```

GetClosestPoints (Layout Editor)

Use: Given another Polygon, find the closest Points on each. The closest Point on this Polygon is returned as the first Point in the list. The closest Point on the other Polygon is returned as the last (or second) Point in the list.

Syntax: oPolygon.GetClosestPoints(oPolygon)

Return Value: A list of Point objects.

```
VB Example: closestPointList = oPolygon.GetClosestPoints(oPolygonOther)
```

Unite (Layout Editor)

Use: Causes Boolean uniting of 2 or more *primitive* (polygons, rectangles, lines, or circles) objects.

Syntax: **Unite** Array ("NAME: *primitives*",

<object_name>, // 1st primitive name

<object_name>, // 2nd primitive, if any

```
...) // etc
```

VB Example:

```
oEditor.Unite Array("NAME:primitives", "circle_0", "rect_2")
```

Intersect (Layout Editor)

Use: Causes Boolean intersecting of 2 or more *primitive* (polygons, rectangles, lines, or circles) objects.

Syntax: **Intersect** Array ("NAME: *primitives*",

<object_name>, // 1st primitive name

<object_name>, // 2nd primitive, if any

...)// etc

VB Example:

```
oEditor.Intersect Array("NAME:primitives", "circle_0", "rect_2")
```

Subtract (Layout Editor)

Use: Causes boolean subtracting of one or more *primitive* (polygons, rectangles, lines, or circles) object(s) from another one.

Syntax: **Subtract** Array ("NAME: *primitives*",

<object_name>, // Primitive to subtract from

<object_name>, // 1nd primitive to subtract, if any

...) // etc

VB Example:

```
oEditor.Intersect Subtract ("NAME:primitives", "circle_0", "rect_2")
```


Xor (Layout Editor)

Use: Exclusive or (XOR) this Polygon with another and return as a list of Polygons.

Syntax: oPolygon.Xor(oPolygon)

Return Value: Xor as a list of Polygon objects.

VB Example:

```
resList = oPolygon.Xor(oPolygonOther)
```

Boolean Operations on Primitives

The following boolean operations on primitives are available.

[ConvertPrimitives \(Layout Editor\)](#)

[ConvertPrimitivesToVias \(Layout Editor\)](#)

[Unite \(Layout Editor\)](#)

[Intersect \(Layout Editor\)](#)

[SplitRegion \(Layout Editor\)](#)

[Subtract \(Layout Editor\)](#)

[ToggleViaPin \(Symbol Editor\)](#)

ConvertPrimitives (Layout Editor)

Use: Convert the selected primitives to 2.5D vias

Syntax: ConvertPrimitives(ARRAY primvia_data) // primitive data

Example: oModule.ConvertPrimitives primvia_data

ConvertPrimitivesToVias (Layout Editor)

Converts primitives in the active design to vias.

UI Access	After selecting one or more primitives, navigate to Draw > Convert > Primitives to Vias .		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array containing these parameters in order: • <NAME:elements> • <TargetPrimitives>
	<NAME:elements>	String	Required first parameter which must have the specified value (i.e., "elements").
	<TargetPrimitives>	String	Names of the target primitives to convert to vias, separated by commas.
Return Value	None.		

Python Syntax	ConvertPrimitivesToVias([<NAME:elements>, <TargetPrimitives>])
Python Example	<pre>oEditor.ConvertPrimitivesToVias (["NAME:elements", "rect_395"])</pre>

VB Syntax	ConvertPrimitivesToVias(<NAME:elements>, <TargetPrimitives>)

VB Example	<code>oEditor.ConvertPrimitivesToVias Array("NAME:elements", "rect_395")</code>

Unite (Layout Editor)

Use: Causes Boolean uniting of 2 or more *primitive* (polygons, rectangles, lines, or circles) objects.

Syntax: **Unite** Array ("NAME: *primitives*",

<object_name>, // 1st primitive name

<object_name>, // 2nd primitive, if any

...) // etc

VB Example:

```
oEditor.Unite Array("NAME:primitives", "circle_0", "rect_2")
```

Intersect (Layout Editor)

Use: Causes Boolean intersecting of 2 or more *primitive* (polygons, rectangles, lines, or circles) objects.

Syntax: **Intersect** Array ("NAME: *primitives*",

<object_name>, // 1st primitive name

<object_name>, // 2nd primitive, if any

...)// etc

VB Example:

```
oEditor.Intersect Array("NAME:primitives", "circle_0", "rect_2")
```

SplitRegion (Layout Editor)

Split geometry against a polygon region.

UI Access	After the polygon is created and the design and polygon are selected, right-click and select Draw > Split Polygon Region .
Parameters	None.
Return Value	"Name:Poly_#"

Python Syntax	SplitRegion
Python Example	<pre>oEditor.SplitRegion(["NAME:ExampleName", "Poly_1"])</pre>

VB Syntax	SplitRegion
VB Example	<pre>oEditor.SplitRegion "NAME:ExampleName", "Poly_1"</pre>

Subtract (Layout Editor)

Use: Causes boolean subtracting of one or more *primitive* (polygons, rectangles, lines, or circles) object(s) from another one.

Syntax: **Subtract** Array ("NAME: *primitives*",

<object_name>, // Primitive to subtract from

<object_name>, // 1nd primitive to subtract, if any

...) // etc

VB Example:

```
oEditor.Intersect Subtract ("NAME:primitives", "circle_0", "rect_2")
```

ToggleViaPin (Symbol Editor)

Selected pins are changed to vias. Selected vias are changed to pins.

Syntax: ToggleViaPin <NAME:elements", <object_name> ... // objects to be toggled

Return Value: None.

Parameters: <object_name>

Type: <String>

VB Example: oEditor.ToggleViaPin Array("NAME:elements", "via_195")

Coordinate System Methods

The following scripts are available for use with the Designer Coordinate System.

[CreateCS \(Layout Editor\)](#)

[ClearRelative \(Layout Editor\)](#)

[Create3DStructure \(Layout Editor\)](#)

[GetCSObjects \(Layout Editor\)](#)

[Group \(Layout Editor\)](#)

[CreateGroupSelected \(Layout Editor\)](#)

[PositionRelative \(Layout Editor\)](#)

[SetCS \(Layout Editor\)](#)

[Ungroup \(Layout Editor\)](#)

CreateCS (Layout Editor)

Use: Inserts a new coordinate system (CS) into the currently active Layout design.

Command: Draw > Coordinate System > Create.

Syntax: There are two forms available:

```
CreateCS  
Array("NAME:Contents",  
Array("NAME:full_definition", "type:=", "CS", "N:=", <"CS name">),  
"placement:=", Array("x:=", <"x_coord">, "y:=", <"y_coord">),  
"Clf:=", <color flag>, "Col:=", <color> )
```

```
CreateCS  
Array("NAME:Contents",  
"Name:=", <"CS name">),  
"RelPos:=", Array(edge description))
```

Return Value: Text containing the name of the created relative coordinate system.

Parameters: <"CS name">

Text with the requested name for the relative coordinate system

<x_coord>

Text that is the X location for the relative coordinate system origin

<y_coord>

Text that is the Y location for the relative coordinate system origin

// the following arguments are optional

<color flag>

True if a color is being specified by the next parameter

<color>

Integer specifying the color for the relative coordinate system.

Only used if color flag is true;

<edge description> for primitive edges

```
"t:=", "pe", "from:=", <"prim">, "pos:=", <edge position>, "et:=", "pe", "prim:=",  
<"prim">, "edge:=", <edge#>
```

<"prim">: text that is the primitive name

<edge position>

double between 0 and 1, inclusive

0 indicates the start of the edge

1 indicates the end of the edge

values in between are positions along the edge

<edge#>: integer that is the edge number on the primitive

<edge description> for via edges

```
"t:=", "pe", "from:=", <"via">, "pos:=", <edge position>, "et:=", "pse", "sel:=", <"via">, "lay-  
er:=", <layer id>, "sx:=", <start X location>, "sy:=", <start Y location>, "ex:=", <end X loc-  
ation>, "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)

```
VB Example: oEditor.CreateCS Array("NAME:Contents", Array("NAME:full_definition",  
"type:=", "CS", "N:=", "CS_232"), "placement:=", Array("x:=", "-15mm", "y:=",  
"15mm"), "Clf:=", false, "Col:=", 8421504)
```

```
oEditor.CreateCS Array("NAME:Contents", "Name:=", "CS_15", "RelPos:=",
```

```
Array("t:=", "pe", "from:=", "poly_1", "pos:=", 0, "et:=", "pe", "prim:=", "poly_1",  
"edge:=", 1))
```

```
oEditor.CreateCS Array("NAME:Contents", "Name:=", "CS_196", "RelPos:=",  
Array("t:=", "pe", "from:=", "via_0", "pos:=", 0, "et:=", "pse", "sel:=", "via_0",  
"layer:=", 10, "sx:=", -0.0015, "sy:=", -0.0015, "ex:=", 0.0015, "ey:=", -0.0015,  
"h:=", 0, "rad:=", 0))
```

ClearRelative (Layout Editor)

Use: Clear the relative positioning between a coordinate system (CS) and another object.

Command: None.

Syntax: oEditor.ClearRelative Array("NAME:elements", <CS1>, <CS2>, ...)

Return Value: None.

VB Example: oEditor.ClearRelative Array("NAME:elements", "CS_13")

Create3DStructure (Layout Editor)

Use: Place a set of primitives into a 3D structure coordinate system (i.e. a 3D via or cross-layer plate).

Command: None.

Syntax: oEditor.Create3DStructure <3D structure name>, Array("NAME:elements", <element1>, <element2>, ...)

Return Value: None.

VB Example: oEditor.Create3DStructure "S3D_11", Array("NAME:elements", "circle_10", "circle_9")

GetCSObjects (Layout Editor)

Use: Given a coordinate system name, returns a list of the objects inside. Coordinate system names can be found using FindObjects ('Type', 'CS'). See [Find Objects](#).

Syntax: oLayout.GetCSObjects(<CS_name>)

Return Value: List of objects in the coordinate system.

VB Example: CS_objects = oLayout.GetCSObjects("CS_1")

Group (Layout Editor)

Use: Groups a set of primitives into a coordinate system.

Command: None.

Syntax: oEditor.Group <CS name>, Array("NAME:elements", <element1>, <element2>, ...)

Return Value: None.

VB Example: oEditor.Group "CS_7", Array("NAME:elements", "rect_4", "rect_6")

CreateGroupSelected (Layout Editor)

Use: Groups a set of primitives into a subdesign.

Command: Layout > Group into Subdesign

Syntax: oEditor.CreateGroupSelected Array("NAME:elements", <selectable name>, ...)

Return Value: None.

VB Example: oEditor.CreateGroupSelected Array("NAME:elements", "1", "2", "3", "4", "5", "6")

PositionRelative (Layout Editor)

Use: Position a coordinate system (CS), including 3D structures and components, relative to another object.

Command: Draw > Position Relative

Syntax: There are two forms available:

PositionRelative

```
Array("NAME:Contents",  
"RelPos:=", Array(<edge description>)),  
Array("NAME:elements", <coordinate system name>, < coordinate system name >, ...)
```

```
PositionRelative Array("NAME:Contents",  
"RelPos:=", Array("t:=", "pr", "from:=", <edge port or pin name>)),  
Array("NAME:elements", <coordinate system name>, <coordinate system name>, ...)
```

Return Value: None.

Parameters:

```
Array("NAME:elements", "CS_23")
```

<edge description> for primitive edges

"t:=", "pe", "from:=", <"prim">, "pos:=", <edge position>, "et:=", "pe", "prim:=", <"prim">, "edge:=", <edge#>

<"prim">: text that is the primitive name

<edge position>

double between 0 and 1, inclusive

0 indicates the start of the edge

1 indicates the end of the edge

values in between are positions along the edge

<edge#>: integer that is the edge number on the primitive

<edge description> for via edges

"t:=", "pe", "from:=", <"via">, "pos:=", <edge position>, "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)

< coordinate system name >

Text that contains the name of the coordinate system

<edge port or pin name>

Text that contains the name of the edge port or via

VB Example:

```
oEditor.PositionRelative Array("NAME:Contents", "RelPos:=", Array("t:=", "pe",  
"from:=", "poly_0", "pos:=", 0, "et:=", "pe", "prim:=", "poly_0", "edge:=", 3)),  
Array("NAME:elements", "CS_23")
```

```
oEditor.PositionRelative Array("NAME:Contents", "RelPos:=", Array("t:=", "pe",  
"from:=", "via_0", "pos:=", 0, "et:=", "pse", "sel:=", "via_0", "layer:=", 10, "sx:=",  
0.0015, "sy:=", 0.0015, "ex:=", -0.0015, "ey:=", 0.0015, "h:=", 0, "rad:=", 0)),  
Array("NAME:elements", "CS_201")
```

SetCS (Layout Editor)

Use: Activate a coordinate system (CS).

Command: None.

Syntax: oEditor.SetCS <CS name or blank>

Return Value: None.

```
VB Example: oEditor.SetCS "CS_7" // activate CS_7
oEditor.SetCS "" // activate the 'global CS', i.e. no CS is active
```

Ungroup (Layout Editor)

Use: Reverses a group operation; deletes a coordinate system but leaves the primitives.

Command: None.

Syntax: oEditor.Ungroup Array("NAME:elements", <CS1>, <CS2>, ...)

Return Value: None.

```
VB Example: oEditor.Ungroup Array("NAME:elements", "CS_7")
```

IC Editor Script Commands (IC Mode Only)

IC Editor scripting commands are accessible once **Electronics Desktop** has been placed in IC Editor mode (i.e., **IC** mode).

Enter the following commands to place **Electronics Desktop** in **IC** mode and access the *IC Editor* interface by executing the **SetActiveEditor** command.

```
oDesign = oProject.SetActiveDesign("Design_Example1")
```

```
oDesign.SetDesignMode("IC")  
oEditor = oDesign.SetActiveEditor("ICEditor")
```

The call to each scripting method contains three elements:

- Layout editor
- Method name
- Method argument(s) (if any)

For example, in Visual Basic:

```
oEditor.Delete ("rect_1")
```

The behavior of the scripting methods is similar to the corresponding commands in the **Layout Editor**. This correspondence is evident when recording a script in Layout Editor (i.e., **Tools > Record Script**).

The following commands can only be executed from the **IC** mode environment.

[ApplyTechnology \(IC Editor\)](#)

[ChangeActiveLayout \(IC Editor\)](#)

[CreateViaGroupsOnLayer \(IC Editor\)](#)

[ReconstructArcs \(IC Editor\)](#)

[ReconstructArcsOnLayer \(IC Editor\)](#)

[RemoveSmall Holes \(IC Editor\)](#)

[RemoveSmallLines \(IC Editor\)](#)

[RemoveSmallMetalIslands \(IC Editor\)](#)

[ResetTechnology \(IC Editor\)](#)

[SnapPrimitives \(IC Editor\)](#)

[SnapPrimitivesOnLayer \(IC Editor\)](#)

[SnapVias \(IC Editor\)](#)

[SnapViasOnLayer \(IC Editor\)](#)

[UnitePrimitivesOnLayer \(IC Editor\)](#)

[WrapGeometries \(IC Editor\)](#)

[WrapGeometry \(IC Editor\)](#)

[WrapGeometry \(IC Editor\)](#)

Common to General Mode

The following commands are common to **General** mode and **IC** mode environment. Unless otherwise noted, the commands function identically in either environment (e.g., **RemoveLayer** is divided into two sections for each mode since the command function is similar but not identical).

[AddObject \(IC Editor and Layout Editor\)](#)

[AddPinGroupRefPort \(Layout Editor\)](#)

[AddPortsToNet \(Layout Editor\)](#)

[AddRefPort \(Layout Editor\)](#)

[AssignRefPort \(Layout Editor\)](#)

[ChangeLayer \(Layout Editor\)](#)

[ChangeLayers \(Layout Editor\)](#)

[ChangeOptions \(Layout Editor\)](#)

[ChangeProperty \(Schematic Editor and Layout Editor\)](#)

[ClearRefPort \(Layout Editor\)](#)

[ClipPlane \(Layout Editor\)](#)

[ConvertPrimitives \(Layout Editor\)](#)

[ConvertPrimitivesToVias \(Layout Editor\)](#)

[Copy \(Layout Editor\)](#)

[CreateCircle \(Layout Editor\)](#)

[CreateCircuitPort \(Layout Editor\)](#)

[CreateComponent \(Layout Editor\)](#)

[CreateDifferentialPair \(Layout Editor\)](#)

[CreateEdgePort \(Layout Editor\)](#)

[CreateLine \(Layout Editor\)](#)

[CreateLineFromPolygon \(Layout Editor\)](#)

[CreateNetClass \(Layout Editor\)](#)

[CreateObjectFromPolygon \(Layout Editor\)](#)

[CreatePinGroup \(Layout Editor\)](#)

[CreatePinGroupPort \(Layout Editor\)](#)

[CreatePinGroups \(Layout Editor\)](#)

[CreatePolygon \(Layout Editor\)](#)

[CreatePortsOnComponents \(Layout Editor\)](#)

[CreatePortsOnComponentsByNet \(Layout Editor\)](#)

[CreateRectangle \(Layout Editor\)](#)

[CreateViaGroups \(Layout Editor\)](#)

[CutOutSubDesign \(Layout Editor\)](#)

[Delete \(Layout Editor\)](#)

[DeleteNets \(Layout Editor\)](#)

[DeletePinGroup \(Layout Editor\)](#)

[DelNetClass \(Layout Editor\)](#)

[DissolveComponents \(Layout Editor\)](#)

[Duplicate \(Layout Editor\)](#)

[DuplicateAcrossLys \(Layout Editor\)](#)

[Edit \(Layout Editor\)](#)

[EditComponent \(Layout Editor\)](#)

[EditPinGroup \(Layout Editor\)](#)

[EnableComponents \(Layout Editor\)](#)

[ExportStackupXML \(Layout Editor\)](#)

[FilterObjectList \(Layout Editor\)](#)

[FindObjects \(Layout Editor\)](#)

[FindObjectsByPoint \(Layout Editor\)](#)

[FindObjectsByPolygon \(Layout Editor\)](#)

[GetActiveUnits \(Layout Editor\)](#)

[GetAllLayerNames \(Layout Editor\)](#)

[GetBBox \(Layout Editor\)](#)

[GetCSObjects \(Layout Editor\)](#)

[GetEditorName \(Layout Editor\)](#)

[GetLayerInfo \(Layout Editor\)](#)

[GetMaterialList \(Layout Editor\)](#)

[GetNetClasses \(Layout Editor\)](#)

[GetNetClassNets \(Layout Editor\)](#)

[GetNets \(Layout Editor\)](#)

[GetPolygon \(Layout Editor\)](#)

[GetPolygonDef \(Layout Editor\)](#)

[GetPolygonVoids \(Layout Editor\)](#)

[GetPortInfo \(Layout Editor\)](#)

[GetProperties \(Layout Editor\)](#)

[GetPropertyValue \(Layout Editor\)](#)

[GetSelections \(Layout Editor\)](#)

[GetStackupLayerNames \(Layout Editor\)](#)

[HighlightNet \(Layout Editor\)](#)

[ImportStackupXML \(Layout Editor\)](#)

[Intersect \(Layout Editor\)](#)

[ModifyDifferentialPair \(Layout Editor\)](#)

[ModifyNetClass \(Layout Editor\)](#)

[Move \(Layout Editor\)](#)

[Paste \(Layout Editor\)](#)

[Point \(Layout Editor\)](#)

[Polygon \(Layout Editor\)](#)

[RemoveLayer \(Layout Editor\)](#)

[RemovePort \(Symbol Editor\)](#)

[RemovePortsFromAllNets \(Layout Editor\)](#)

[RemovePortsOnComponents \(Layout Editor\)](#)

[ResetDifferentialPairs \(Layout Editor\)](#)

[Select \(Layout Editor\)](#)

[SelectAll \(Layout Editor\)](#)

[SelectNetConnected \(Layout Editor\)](#)

[SelectPhysicallyConnected \(Layout Editor\)](#)

[SetActiveUnits \(Layout Editor\)](#)

[SetCS \(Layout Editor\)](#)

[SetHfssExtentsVisible \(Layout Editor\)](#)

[SetHfssPortsVisible \(Layout Editor\)](#)

[SetNetColor \(Layout Editor\)](#)

[SetNetVisible \(Layout Editor\)](#)

[SetPropertyValue \(Layout Editor\)](#)

[Subtract \(Layout Editor\)](#)

[ToggleNetHighlight \(Layout Editor\)](#)

[ToggleViaPin \(Symbol Editor\)](#)

[Ungroup \(Layout Editor\)](#)

[Unite \(Layout Editor\)](#)

[UnselectAll \(Layout Editor\)](#)

ApplyTechnology (IC Editor)

Applies the specified technology to the active design's stackup.

UI Access	From the Layout ribbon, select Apply Technology from the Technology drop-down menu.		
Parameters	Name	Type	Description
	<DefName>	String	Name of technology to apply to the active design's stackup.
Return Value	None		

Python Syntax	ApplyTechnology(<DefName>)
Python Example	<code>oEditor.ApplyTechnology("Model.vlc.tech")</code>

VB Syntax	ApplyTechnology <DefName>
VB Example	<code>oEditor.ApplyTechnology "Model.vlc.tech"</code>

ChangeActiveLayout (IC Editor)

Switches the active layout.

UI Access	Navigate to the Cells window > Hierarchical tab. Then double-click the appropriate instance in the hierarchical tree.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array containing these parameters in order: • <NAME:CellInstances> • <SelectedLayout>
	<NAME:CellInstances>	String	Required first parameter which must have the specified value (i.e., "CellInstances").
	<SelectedLayout>	Integer	A comma-separated list of two integers, where the first integer always represents the top of the hierarchical tree (i.e., "0") and the second identifies the layer to make active (i.e., "1", "2", "3", etcetera).
Return Value	None		

Python Syntax	ChangeActiveLayout([<NAME:CellInstances>, <SelectedLayout>])
Python Example	<pre>oEditor.ChangeActiveLayout (["NAME:CellInstances", "0", "1"])</pre>

VB Syntax	<code>ChangeActiveLayout <NAME:CellInstances>, <SelectedLayout></code>
VB Example	<pre>oEditor.ChangeActiveLayout Array("NAME:CellInstances", "0", "1")</pre>

CreateMeshRegion (IC Editor)

Designate a mesh region in a design and create partitions for simulation.

UI Access	From the Layout Editor , select a component. Then right-click and select Mesh Region > Create .		
Parameters	Name	Type	Description
	<MeshRegionSettings>	Array	Structured array containing these parameters in order: • <NAME:MeshRegionSettings> • <NumPrimaryPartitions> • <NumSecondaryPartitions> • <XY> • <PosZ> • <NegZ> • <UseActiveNets> • <InclRef> • <ExtentType>
	<NAME:MeshRegionSettings>	String	Identifies the contents of the array (i.e., "MeshRegionSettings").
	<NumPrimaryPartitions>	Integer	Number of primary partitions to create in the new

			mesh region.
	<NumSecondaryPartitions>	Integer	Number of secondary partitions to create in the new mesh region.
	<XY>	Integer	Horizontal padding on both sides of the new mesh.
	<PosZ>	Integer	Vertical padding above the new mesh region.
	<NegZ>	Integer	Vertical padding below the new mesh region.
	<UseActiveNets>	Boolean	True will create a new mesh region defined by the active nets in the design. False will create a new mesh region defined by the dielectric extents in the design.
	<InclRef>	Boolean	True will include both positive nets and reference nets in the definition of the new mesh region, if <UseActiveNets> is True . False will not include both positive nets and reference nets.
	<ExtentType>	Integer	Define the new mesh region within a: 0 — bounding box. 1 — conformal coat. 2 — convex hull.
Return Value	None.		

Python Syntax	CreatMeshRegion([<NAME:MeshRegionSettings>, <NumPrimaryPartitions>, <NumSecondaryPartitions>, <XY>, <PosZ>, <NegZ>, <UseActiveNets>, <InclRef>, <ExtentType>])
Python Example	<pre>oEditor.CreateMeshRegion(["NAME:MeshRegionSettings",</pre>

	<pre> "NumPrimaryPartitions:=", 6, "NumSecondaryPartitions:=", 1, "XY:=", "0.005", "PosZ:=", "0.1", "NegZ:=", "0.1", "UseActiveNets:=", True, "InclRef:=", False, "ExtentType:=", 0]) </pre>

V-B Syntax	CreatMeshRegion Array(<NAME:MeshRegionSettings>, <NumPrimaryPartitions>, <NumSecondaryPartitions>, <XY>, <PosZ>, <NegZ>, <UseActiveNets>, <InclRef>, <ExtentType>)
V-B Example	<pre> oEditor.CreateMeshRegion Array("NAME:MeshRegionSettings", "NumPrimaryPartitions:=", 6, "NumSecondaryPartitions:=", 1, "XY:=", "0.005", "PosZ:=", "0.1", "NegZ:=", "0.1", "UseActiveNets:=", True, "InclRef:=", False, "ExtentType:=", 0) </pre>

CreateMeshRegionPartitionLine (IC Editor)

Create an additional partition line for a mesh region (i.e., CreateMeshRegion).

UI Access	From the Layout Editor , select a mesh region. Then right-click and select Mesh Region > Add a partition line .		
Parameters	Name	Type	Description
	<Contents>	Array	Structured array containing these parameters in order: • <NAME:Contents> • <owner> • <mesh region partition lineGeometry> • <snap start> • <snap end>
	<NAME:Contents>	String	Identifies the contents of the array (i.e., "Contents").
	<owner>	String	Name of the mesh region in which to create the new partition line.
	<mesh region partition lineGeometry>	Array	Name of the array containing the geometry of the new partition line, containing these parameters in order: • <Name> • <LayerName> • <lw> • <endstyle> • <StartCap> • <EndCap> • <n> • <U> • <x>

		• <y> • <MR>
<Name>	String	Name of the new partition line.
<LayerName>	Integer	Name of the layer on which to create the new partition line (i.e., "MeshRegions").
<lw>	Integer	Line width of the new partition line, including unit of measurement (i.e., "0.01mm").
<endstyle>	Integer	Of the new partition line, all vertices (i.e., points) will be: 0 — round. 1 — corner (i.e., square). 2 — mitered (i.e., extended to a point where it meets a corner).
<StartCap>	Integer	Of the new partition line, the starting point of the line will be: 0 — round. 1 — flat. 2 — extended (i.e., extended from the starting point beyond the actual beginning of its path).
<Endcap>	Integer	Of the new partition line, the ending point of the line will be: 0 — round. 1 — flat. 2 — extended (i.e., extended beyond the actual end point). 3 — Same as Start.
<n>	Integer	The number of vertices in the new partition line.
<U>	String	The unit of measurement by which the new partition will be created.
<x>	Integer	X coordinate of a point in the new partition line.
<y>	Integer	Y coordinate of a point in the new partition line.
<MR>	String	A number and unit of measurement representing the miter ratio (i.e., of the new partition line, assuming multiple segments, the size of the point where two segments meet).
<snap start>	Boolean	True start of the new partition line will snap to a border of the mesh region.
<snap end>	Boolean	True end of the new partition line will snap to a border of the

	<table><tr><td></td><td></td><td>mesh region.</td></tr></table>			mesh region.
		mesh region.		
Return Value	Returns the name of the new partition line.			

Python Syntax	<code>CreatMeshRegionPartitionLine([<NAME:Contents>, <owner>, <mesh region partition lineGeometry>, [<Name>, <LayerName>, <lw>, <endstyle>, <StartCap>, <EndCap>, <n>, <U>, <x>, <y>, <MR>], <snap start>, <snap end>])</code>
Python Example	<pre>oEditor.CreateMeshRegionPartitionLine(["NAME:Contents", "owner:=" , "meshregion__11", "mesh region partition lineGeometry:=", ["Name:=" , "line__18", "LayerName:=" , "MeshRegions", "lw:=" , "0.01mm", "endstyle:=" , 0, "StartCap:=" , 0, "EndCap:=" , 0, "n:=" , 3, "U:=" , "mm", "x:=" , 19,</pre>

```

        "y:=" , 9,
        "x:=" , 15,
        "y:=" , 4,
        "x:=" , 20,
        "y:=" , 1,
        "MR:=" , "600mm"],
    "snap start:=" , True,
    "snap end:=" , True
  )

```

V B S y n t a x	Cre a t M e s h R e g i o n P a r t i t i o n A r r a y (< N A M E : C o n t e n t s > , < o w n e r > , < m e s h r e g i o n p a r t i t i o n l i n e G e o m e t r y > , A r r a y (< N a m e > , < L a y e r N a m e > , < l w > , < e n d s t y l e > , < S t a r t C a p > , < E n d C a p > , < n > , < U > , < x > , < y > , < M R > , < s n a p s t a r t > , < s n a p e n d >)
V B E x - a - m	o E d i t o r . C r e a t M e s h R e g i o n P a r t i t i o n A r r a y (" N A M E : C o n t e n t s " , " o w n e r := " , " m e s h r e g i o n _ 1 " , " m e s h r e g i o n p a r t i t i o n l i n e G e o m e t r y := " , A r r a y (" N a m e := " , " l i n e _ 1 8 " , " L a y e r N a m e := " , " M e s h R e g i o n s " , " l w := " , " 0 . 0 1 m m " , " e n d s t y l e := " , 0 " S t a r t C a p := " , 0 " E n d C a p := " , 0 " n := " , 3 " U := " , " m m " , " x := " , 1 9 " y := " , 9 " x := " , 1 5 " y := " , 4 " x := " , 2 0 " y := " , 1 " M R := " , " 6 0 0 m m ") , " s n a p s t a r t := " , T r u e " , " s n a p e n d := " , T r u e)

p-
l-
e**CreateViaGroupsOnLayer (IC Editor)**

Creates a via group on the target layer.

UI Access	From the Layers window, right-click the target layer and select Group Vias > Persistent or Non-Persistent .		
Parameters	Name	Type	Description
	<NAME>	String	Name of the target layer.
	<Parameters>	Array	Structured array containing these parameters in order: • <NAME:Setting Contents> • <Group_By_Proximity> • <Via_Grouping_Tolerance> • <Check_Containment> • <Via_Persistence>
	<NAME:Setting Contents>	String	Required first parameter which must have the specified value (i.e., "Setting Contents").
	<Group_By_Proximity>	Boolean	True groups via by proximity. False groups by range.
	<Via_Grouping_Tolerance>	Integer	When <Group_By_Proximity> = False , vias at or within this tolerance are grouped.
	<Check_Containment>	Boolean	True analyzes the placement of via primitives. When the possibility of an electrical short is detected, <Check_Containment> attempts to prevent the short. False deactivates the check.
	<Via_Persistence>	Boolean	True maintains the new via group. False selects the group of vias but does not maintain a new via

	group.
Return Value	None.

Python Syntax	CreateViaGroupsOnLayer(<NAME>, [<NAME:Setting Contents>, <Group_By_Proximity>, <Via_Grouping_Tolerance>, <Check_Containment>, <Via_Persistence>])
Python Example	<pre>oEditor.CreateViaGroupsOnLayer(["Trace", "NAME:Setting Contents", "group_by_proximity:=", "True", "via_grouping_tolerance:=", "100um", "check_containment:=", "True", "via_persistence:=", "False"]) </pre>

VB Syntax	CreateViaGroupsOnLayer <NAME>, (<NAME:Setting Contents>, <Group_By_Proximity>, <Via_Grouping_Tolerance>, <Check_Containment>, <Via_Persistence>)
VB Example	<pre>oEditor.CreateViaGroupsOnLayer "Trace", Array("NAME:Setting Contents", "group_by_proximity:=", true, "via_grouping_tolerance:=", "100um", "check_containment:=", true, "via_persistence:=", false) </pre>

ReconstructArcs (IC Editor)

Fits arcs and circles to the selected polygons.

UI Access	After choosing polygons, select Reconstruct Arcs from the Layout ribbon.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array containing these parameters in order: • <NAME:polygons> • <Objects> • <Arcs_Tolerance>
	<NAME:polygons>	String	Required first parameter which must have the specified value (i.e., "polygons").
	<Objects>	List	A comma-separated list of integers, where each integer represents a selected polygon (i.e., "poly_30", "poly_31", "poly_32", etcetera)
Return Value	<Arcs_Tolerance>	Integer	Arcs and circles at or within this tolerance are fitted.
	None.		

Python Syntax	ReconstructArcs([<NAME:polygons>, <Objects>], <Arcs_Tolerance>)
Python Example	<pre>oEditor.ReconstructArcs(["NAME:polygons", "poly_30", "poly_31", "poly_32"</pre>

	<code>], 0.00254)</code>

VB Syntax	<code>ReconstructArcs (<NAME:polygons>, <Objects>), <Arcs_Tolerance></code>
VB Example	<code>oEditor.ReconstructArcs Array("NAME:polygons", "poly_30", "poly_31", "poly_32"), 0.00254</code>

ReconstructArcsOnLayer (IC Editor)

Fits arcs and circles to the selected polygons on a target layer.

UI Access	After choosing polygons, navigate to the Layers window. Then right-click the target layer and select Reconstruct Arcs .		
Parameters	Name	Type	Description
	<code><NAME></code>	String	Name of the target layer.
	<code><Arcs_Tolerance></code>	Integer	Arcs and circles at or within this tolerance are fitted.
Return Value	None.		

Python Syntax	<code>ReconstructArcsOnLayer(<NAME>, <Arcs_Tolerance>)</code>
Python Example	<code>oEditor.ReconstructArcsOnLayer("Trace", 0.00254)</code>

VB Syntax	ReconstructArcsOnLayer <NAME>, <Arcs_Tolerance>
VB Example	<code>oEditor.ReconstructArcs "Trace", 0.00254</code>

RemoveSmallHoles (IC Editor)

Removes small holes at or within tolerance from the target layer.

UI Access	Select Remove Holes from the Layout ribbon.		
Parameters	Name	Type	Description
	<Max_Area_Tolerance>	Integer	Holes at or within this tolerance are removed.
	<Exclusion_Tolerance>	Integer	Distance around nets to exclude from removal.
	<Parameters>	Array	Structured array containing these parameters in order: • <NAME:Nets> • <SelectedLayout>
	<NAME:Nets>	String	Required first parameter which must have the specified value (i.e., "Nets").
	<NetName>	String	Name of the target net.
Return Value	None		

Python Syntax	AddObject(<ObjectType>, <LayerName>)
Python Example	<pre>oEditor.RemoveSmallHoles("1000um2", "1um", ["NAME:Nets",</pre>

	<pre>"net1"])</pre>

VB Syntax	AddObject <ObjectType>, <LayerName>
VB Example	oEditor.RemoveSmallHoles "1000um2", "1um", Array("NAME:Nets", "net1")

RemoveSmallLines (IC Editor)

Removes small lines at or within tolerance from the target layer.

UI Access	From the Layers window, right-click the target layer and select Remove Small Lines .		
Parameters	Name	Type	Description
	<NAME>	String	Name of the target layer.
	<Max_Length_Tolerance>	Integer	Lines at or within this tolerance are removed.
Return Value	None		

Python Syntax	RemoveSmallLines(<NAME>, <Max_Length_Tolerance>)
Python Example	oEditor.RemoveSmallLines("Trace", "1mm")

VB Syntax	RemoveSmallLines <NAME>, <Max_Length_Tolerance>
VB Example	<code>oEditor.RemoveSmallLines "Trace", "1mm"</code>

RemoveSmallMetallIslands (IC Editor)

Removes primitives and vias at or within tolerance from the target layer.

UI Access	After choosing primitives, select Remove Islands from the Layout Ribbon .		
Parameters	Name	Type	Description
	<NAME>	String	Name of the target layer.
	<Max_Area_Tolerance>	Integer	Primitives and vias at or within this tolerance are removed.
Return Value	None		

Python Syntax	RemoveSmallMetallIslands(<NAME>, <Max_Area_Tolerance>)
Python Example	<code>oEditor.RemoveSmallMetalIslands("Trace", "1mm")</code>

VB Syntax	RemoveSmallMetallIslands <NAME>, <Max_Area_Tolerance>
VB Example	<code>oEditor.RemoveSmallMetalIslands "Trace", "1mm"</code>

ResetTechnology (IC Editor)

Disassociates the applied technology from the active design's stackup.

UI Access	From the Layout ribbon, select Reset Technology from the Technology drop-down menu.
Parameters	None
Return Value	None

Python Syntax	ResetTechnology()
Python Example	<code>oEditor.ResetTechnology()</code>

VB Syntax	ResetTechnology
VB Example	<code>oEditor.ResetTechnology</code>

SnapPrimitives (IC Editor)

Aligns the edge of the selected primitives.

UI Access	After choosing primitives, select Snap Primitives from the Layout ribbon.		
Parameters	Name	Type	Description
	<LAYER_NAME>	String	Name of the target layer.
	<Parameters>	Array	Structured array containing these parameters in order: • <NAME:primitives> • <Objects> • <Snap_Dis_Tolerance>

			• <CheckConnectivity>
	<NAME:primitives>	String	Required first parameter which must have the specified value (i.e., "primitives").
	<Objects>	List	A comma-separated list of integers, where each integer represents a selected primitive (i.e., "rect_0", "rect_1", "rect_2", etcetera)
	<Snap_Dis_Tolerance>	Integer	Primitives at or within this tolerance are snapped.
	<CheckConnectivity>	Boolean	True analyzes the placement of primitives to ensure snapping does not violate the connectivity of the original objects. False deactivates the check.
Return Value	None.		

Python Syntax	SnapPrimitives(<LAYER_NAME> [<NAME:primitives>, <Objects>], <Snap_Dis_Tolerance>, <Check-Connectivity>)
Python Example	<pre>oEditor.SnapPrimitives(["Trace", "NAME:primitives", "rect_0:=", "rect_1", "rect_2",], "5mm", True)</pre>

VB Syn-	SnapPrimitives <LAYER_NAME> (<NAME:primitives>, <Objects>), <Snap_Dis_Tolerance>, <CheckConnectivity>

tax	
VB Example	<code>oEditor.SnapPrimitives "Trace", Array("NAME:primitives", "rect_0", "rect_1", "rect_2"), "5mm", True</code>

SnapPrimitivesOnLayer (IC Editor)

Aligns the edge of the selected primitives on the target layer.

UI Access	From the Layers window, right-click the target layer and select Snap Primitives .		
Parameters	Name	Type	Description
	<LAYER_NAME>	String	Name of the target layer.
	<Snap_Dis_Tolerance>	Integer	Primitives at or within this tolerance are snapped.
	<CheckConnectivity>	Boolean	True analyzes the placement of primitives to ensure snapping does not violate the connectivity of the original objects. False deactivates the check.
Return Value	None.		

Python Syntax	<code>SnapPrimitivesOnLayer(<LAYER_NAME>, <Snap_Dis_Tolerance>, <CheckConnectivity>)</code>
Python Example	<code>oEditor.SnapPrimitivesOnLayer("Trace", "5mm", True)</code>

VB Syntax	<code>SnapPrimitivesOnLayer <LAYER_NAME>, <Snap_Dis_Tolerance>, <CheckConnectivity></code>
VB Example	<code>oEditor.SnapPrimitivesOnLayer "Trace", "1mm", True</code>

SnapVias (IC Editor)

Forces the edges of selected vias to snap to alignment.

UI Access	After choosing vias, select Snap Vias from the Layout ribbon.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array containing these parameters in order: • <NAME:vias> • <Objects>
	<NAME:vias>	String	Required first parameter which must have the specified value (i.e., "vias").
	<Objects>	Integer	A comma-separated list of integers, where each integer represents a selected via (i.e., "via_0", "via_1", "via_2", etcetera)
	<Parameters2>	Array	Structured array containing these parameters in order: • <NAME:Setting Contents> • <IsFactorBased> • <SnapTol> • <AlignTol> • <RemoveDanglingVias>
	<NAME:Setting Contents>	String	Required first parameter which must have the specified value (i.e., "Setting Contents").
	<IsFactorBased>	Boolean	True selected via groups are snapped by a maximum area equal to the tolerance (i.e., <Snaptol> and <AlignTol>) times the surface area of the via group. False selected via groups are snapped by a maximum distance of the tolerance.
	<SnapTol>	Integer	Vias at or within this tolerance are snapped.
	<AlignTol>	Integer	Vias at or within this tolerance are aligned.
	<RemoveDanglingVias>	Boolean	True removes vias from selection that have no connectivity or have connectivity only to a single layer. False deactivates this check.

Return Value	None.

Python Syntax	SnapVias([<NAME:vias>, <Objects>], [<NAME:Setting Contents>, <IsFactorBased>, <SnapTol>, <AlignTol>, <RemoveDanglingVias>])
Python Example	<pre>oEditor.SnapVias(["NAME:vias", "via_0", "via_1", "via_2"], ["NAME:Setting Contents", "IsFactorBased:=", True, "snapTol:=", "3", "AlignTol:=", "300um", "removeDanglingVias:=", True])</pre>

VB Syn-	SnapVias (<NAME:vias>, <Objects>), (<NAME:Setting Contents>, <IsFactorBased>, <SnapTol>, <AlignTol>, <RemoveDanglingVias>)

tax	
VB Example	<pre>oEditor.SnapVias Array("Name:vias", "via_0", "via_1", "via_2"), Array(NAME:Setting Contents, "IsFactorBased:=", True, "snapTol:=", "3", "AlignTol:=", "300um", "removeDanglingVias:=", True)</pre>

SnapViasOnLayer (IC Editor)

Forces the edges of selected vias to snap to alignment on the target layer.

UI Access	From the Layers window, right-click the target layer and select Snap Vias .		
Parameters	Name	Type	Description
	<Layer_Name>	String	Name of the target layer.
	<Parameters>	Array	Structured array containing these parameters in order: • <NAME:Setting Contents> • <IsFactorBased> • <SnapTol> • <AlignTol> • <RemoveDanglingVias>
	NAME:Setting Contents	Array	Required first parameter which must have the specified value (i.e., "Setting Contents").
	IsFactorBased	Array	True selected via groups are snapped by a maximum area equal to the tolerance (i.e., <Snaptol> and <AlignTol>) times the surface area of the via group. False selected via groups are snapped by a maximum distance of the tolerance.
	SnapTol	Array	Vias at or within this tolerance are snapped.
	AlignTol	Array	Vias at or within this tolerance are aligned.

	<i>RemoveDanglingVias</i>	Array	True removes vias from selection that have no connectivity or have connectivity only to a single layer. False deactivates this check.
Return Value	None.		

Python Syntax	SnapVias(<Layer_Name>, [<NAME:Setting Contents>, <IsFactorBased>, <SnapTol>, <AlignTol>, <RemoveDanglingVias>])
Python Example	<pre>oEditor.SnapViasOnLayer(["Trace", "NAME:Setting Contents", "IsFactorBased:=", False, "snapTol:=", "5mm", "AlignTol:=", "5mm", "removeDanglingVias:=", True]) </pre>

VB Syntax	SnapVias <Layer_Name>, (<NAME:Setting Contents>, <IsFactorBased>, <SnapTol>, <AlignTol>, <RemoveDanglingVias>)
VB Example	oEditor.SnapViasOnLayer "Trace", Array("NAME:Setting Contents", "IsFactorBased:=", False, "snapTol:=", "5mm", "AlignTol:=", "5mm", "removeDanglingVias:=", True)

UnitePrimitivesOnLayer (IC Editor)

Unites primitives on the target layer.

UI Access	From the Layers window, right-click the target layer and select Unite Primitives .		
Parameters	Name	Type	Description
	<NAME>	String	Name of the target layer.
Return Value	None.		

Python Syntax	UnitePrimitivesOnLayer(<NAME>)
Python Example	<code>oEditor.UnitePrimitivesOnLayer("Trace")</code>

VB Syntax	UnitePrimitivesOnLayer<NAME>
VB Example	<code>oEditor.UnitePrimitivesOnLayer "Trace"</code>

WrapGeometries (IC Editor)

Wraps the selected polygons.

UI Access	After choosing the appropriate geometries, select Wrap Geometry from the Layout ribbon.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array containing these parameters in order: • <NAME:Contents>

			• <Elements> • <Objects>
	<NAME:Contents>	String	Required first parameter which must have the specified value (i.e., "Contents").
	<Elements>	String	Required first parameter which must have the specified value (i.e., "Elements").
	<Objects>	Array	A structured array containing a comma-separated list of integers, where each integer represents a selected polygon(i.e., "rect_2577", "rect_2578", "rect_2579", etcetera)
Return Value	None.		

Python Syntax	WrapGeometries ([<NAME:Contents>, Elements, Objects])
Python Example	<pre>oEditor.WrapGeometries(["NAME:Contents", "elements:=", ["rect_2577", "rect_2578", "rect_2579"]])</pre>

VB Syntax	WrapGeometries (<NAME:Contents>, Elements, Objects)
VB Example	<pre>oEditor.WrapGeometries Array("NAME:Contents", "elements:=", Array("rect_2577", "rect_2578", "rect_2579"))</pre>

WrapGeometry (IC Editor)

Sets a tolerance and wraps polygons beginning with the selected polygon.

UI Access	From the Layout ribbon, select Wrap Geometry . Right-click the selected geometry within the design and Adjust the tolerance slider as necessary. Then click Wrap .		
Parameters	Name	Type	Description
	<SEED_OBJECT_NAME>	String	Name of the selected polygon.
	<Snap_Dis_Tolerance>	Integer	Polygons at or within this range are wrapped.
Return Value	None.		

Python Syntax	WrapGeometry (<SEED_OBJECT_NAME>, <Snap_Dis_Tolerance>)
Python Example	oEditor.WrapGeometry ("rect_1855", "5um")

VB Syntax	WrapGeometry <SEED_OBJECT_NAME> <Snap_Dis_Tolerance>
VB Example	oEditor.WrapGeometry "rect_1855", "5um"

Layers Methods

The following methods are available to manipulate layers:

[AddLayer \(Layout Editor\)](#)

[AddObject \(IC Editor and Layout Editor\)](#)

[AddStackupLayer \(Layout Editor\)](#)

[ChangeLayer \(Layout Editor\)](#)

[ChangeLayers \(Layout Editor\)](#)

[ClearLayerMappings \(Layout Editor\)](#)

[CreateBoardBend \(Layout Editor\)](#)

[DuplicateAcrossLyrs \(Layout Editor\)](#)

[ExportStackupXML \(Layout Editor\)](#)

[GetAllLayerNames \(Layout Editor\)](#)

[GetLayerInfo \(Layout Editor\)](#)

[GetStackupLayerNames \(Layout Editor\)](#)

[ImportStackupXML \(Layout Editor\)](#)

[ProcessBentModelCmd \(Layout Editor\)](#)

[RemoveLayer \(Layout Editor\)](#)

[SetLayerMapping \(Layout Editor\)](#)

[SyncRigidFlexSandbox \(Layout Editor\)](#)

AddObject (Layout Editor and IC Editor)

Adds a user-selected object to the active design.

UI Access	Not strictly applicable, although objects <i>can</i> be drawn by making the appropriate selection from the Layout ribbon (i.e., Draw polygon , Draw line , Draw rectangle , Draw arc , Draw circle).

Parameters	Name	Type	Description
	<ObjectType>	String	Type of object to add (i.e., rect (i.e., rectangle), circle, poly (i.e., polygon), line, or 3D line).
	<Parameters>	Array	Structured array containing the parameters required for the type of object selected. The parameters differ for each object. Refer to the following pages: • "CreateRectangle (Layout Editor) " on page 27-66 • "CreateCircle (Layout Editor) " on page 27-36 • "CreatePolygon (Layout Editor) " on page 27-63 • "CreateLine (Layout Editor) " on page 27-50 • "AddObject ("3D Line") (Layout Editor) " on page 25-3 • CreatePartitionLine (IC Editor)
Return Value	Returns the name of the newly created object (IC Mode only).		

Python Syntax	AddObject(<ObjectType>, [<Parameters>])
Python Example	<pre>oEditor.AddObject("rect", ["NAME:Contents", "rectGeometry:=", ["Name:=", "rect_5", "LayerName:=", "Trace", "lw:=", "0",</pre>

	<pre> Ax:=", "-84mm", Ay:=", "16mm", Bx:=", "-77mm", By:=", "12mm"]])</pre>

VB Syntax	AddObject(<ObjectType>, [<Parameters>])
VB Example	<pre> oEditor.AddObject "rect",Array("NAME:Contents", "rectGeometry:=", Array("Name:=", "rect_5", "LayerName:=", "Trace", "lw:=", "0", "Ax:=", "-84mm", "Ay:=", "16mm", "Bx:=", "-77mm", "By:=", "12mm"]])</pre>

AddLayer (Layout Editor)

Adds a layer in a layout definition. This command can take all parameters available in [ChangeLayers \(Layout Editor\)](#) but cannot be recorded.

Note As with other Layout scripting interface commands that modify the layout, this command is not intended for use within scripts that define footprints. The command behavior from within such a script is undefined and may be unexpected. Use the LayoutHost scripting interface commands within scripts that define footprints.

UI Access	None		
Parameters	Name	Type	Description
	<LayerArray>	Array	An array that contains these parameters in order: • <Layername> • <Type> • <TopBottom> • <Color> • <Transparency> • <Pattern> • <VisFlag> • <Locked> • <Draw>
	<LayerName>	String	Name of layer.
	<Type>	String	Type of layer, such as "user", "soldermask", "solderpaste", "silkscreen", "assembly", "wirebond", "glue", "user", "Postprocessing", or "outline".
	<TopBottom>	String	Optional. Indicates whether the layer is the top or bottom. Acceptable values are "top", "bottom", or "neither".
	<Color>	Integer	Optional. A code to indicate the color of the layer. It is an RGB code in the hex format <i>bbggrr</i> . For example red (#0000ff) is 255, green (#00ff00) is 65280, and blue (#ff0000) is 16711680.
	<Transparency>	Integer	Optional. A percentage that indicates the transparency of the layer between 0 (opaque) and 100 (completely transparent and not visible).
	<Pattern>	Integer	Optional. A code for the pattern of the layer: 0 is hollow, 1 is solid, and the other

			patterns 3-8 are various hatch patterns.
	<VisFlag>	Integer	Sets visibility for all objects on the layer using the following additive code: • 1 makes primitives (rectangles, circles, polygons) visible. • 2 makes lines (also known as paths or traces) visible. • 4 makes pads (from padstacks, aka vias) visible. • 8 makes holes (or vias) visible. • 16 makes components visible. • 32 makes the mesh overlay visible if they are plotted. • 64 makes the mesh for the background material visible. For example, a value of 25 (1+8+16) makes primitives, vias, and components visible. Use 127 to show everything and 0 to hide all.
	<Locked>	Boolean	Optional. Whether the layer is locked.
	<Draw>	Integer	Optional. Allows drawing on the layer as wireframe, outlines, or solid. Acceptable values are -1 for wireframe, 0 for normal, and 1 for solid.
Return Value	None		

Python Syntax	<pre>oEditor.AddLayer(["NAME:layers", "Name:=", "<LayerName>", "Type:=", "<Type>", "Top Bottom:=", "<TopBottom>", "Color:=", "<Color>",</pre>

	<pre>"Transparency:=", <Transparency>, "Pattern:=", <Pattern>, "VisFlag:=", <VisFlag>, "Locked:=", <Locked>, "DrawOverride:=", <Draw>,])</pre>
Python Example	<pre>oEditor = oDesign.SetActiveEditor("Layout") oEditor.AddLayer(["NAME:layers", "Name:=", "signal", "Type:=", "assembly", "Top Bottom:=", "neither", "Color:=", 16711680, "Transparency:=", 0, "Pattern:=", 1, "VisFlag:=", 127, "Locked:=", False, "DrawOverride:=", 0,])</pre>

VB Syntax	<code>oEditor.AddLayer Array("NAME:stackup layer", "Name:=", "<LayerName>", "Type:=", "<Type>", "Top Bottom:=", "<TopBottom>", "Color:=", <Color>, "Transparency:=", <Transparency>, "Pattern:=", <Pattern>, "VisFlag:=", <VisFlag>, "Locked:=", <Locked>, "DrawOverride:=", <Draw>,)</code>
VB Example	<pre>Set oEditor = oDesign.SetActiveEditor("Layout") oEditor.AddLayer Array("NAME:stackup layer", "Name:=", _ "NewLayer", "Type:=", "assembly", "Top Bottom:=", "neither", "Color:=", _ 3361318, "Transparency:=", 60, "Pattern:=", 1, "VisFlag:=", 127, "Locked:=", _ false, "DrawOverride:=", 0,)</pre>

AddStackupLayer (Layout Editor)

Adds a stackup layer in the Layout editor. This command can take all parameters available in [ChangeLayers \(Layout Editor\)](#) but cannot be recorded.

UI Access	None		
Parameters	Name	Type	Description
	<LayerArray>	Array	An array that contains these parameters in order: • <Layername> • <Type> • <TopBottom> • <Color> • <Transparency> • <Pattern>

		• <VisFlag> • <Locked> • <Draw> • and an array with the parameters: • <Thickness> • <LowerElevation> • <Material>
<LayerName>	String	Name of layer.
<Type>	String	Type of layer, either "signal" or "dielectric".
<TopBottom>	String	Optional. Indicates whether the layer is the top or bottom. Acceptable values are "top", "bottom", or "neither".
<Color>	Integer	Optional. A code to indicate the color of the layer. It is an RGB code in the hex format <i>bbggrr</i> . For example red (#0000ff) is 255, green (#00ff00) is 65280, and blue (#ff0000) is 16711680.
<Transparency>	Integer	Optional. A percentage that indicates the transparency of the layer between 0 (opaque) and 100 (completely transparent and not visible).
<Pattern>	Integer	Optional. A code for the pattern of the layer: 0 is hollow, 1 is solid, and the other patterns 3-8 are various hatch patterns.
<VisFlag>	Integer	Sets visibility for all objects on the layer using the following additive code: • 1 makes primitives (rectangles, circles, polygons) visible. • 2 makes lines (also known as paths or traces) visible. • 4 makes pads (from padstacks, aka vias) visible. • 8 makes holes (or vias) visible. • 16 makes components visible. • 32 makes the mesh overlay visible if they are plotted. • 64 makes the mesh for the background material visible.

			For example, a value of 25 (1+8+16) makes primitives, vias, and components visible. Use 127 to show everything and 0 to hide all.
	<Locked>	Boolean	Optional. Whether the layer is locked.
	<Draw>	Integer	Optional. Allows drawing on the layer as wireframe, outlines, or solid. Acceptable values are -1 for wireframe, 0 for normal, and 1 for solid.
	<Thickness>	String	Thickness of the layer, contains the measurement and units.
	<LowerElevation>	String	The elevation of the lower edge of the layer, contains the measurement and units.
	<Material>	String	The material the layer is made of.
Return Value	None		

Python Syntax	<pre> oEditor.AddStackupLayers(["NAME:stackup layer", "Name:=", "<LayerName>", "Type:=", "<Type>", "Top Bottom:=", "<TopBottom>", "Color:=", "<Color>", "Transparency:=", "<Transparency>", "Pattern:=", "<Pattern>", "VisFlag:=", "<VisFlag>", "Locked:=", "<Locked>", "DrawOverride:=", "<Draw>", </pre>		

	<pre>["NAME:sublayer", "Thickness:=", "<Thickness>", "LowerElevation:=", "<LowerElevation>", "Material:=", "<Material>",]])</pre>
Python Example	<pre>oEditor = oDesign.SetActiveEditor("Layout") oEditor.AddStackupLayer(["NAME:stackup layer", "Name:=", "Cover", "Type:=", "signal", "Top Bottom:=", "neither", "Color:=", 16711680, "Transparency:=", 0, "Pattern:=", 1, "VisFlag:=", 127, "Locked:=", False, "DrawOverride:=", 0,</pre>

	<pre> ["NAME:Sublayer", "Thickness:=", "0mil", "LowerElevation:=", "20mil", "Material:=", "gold",]]) </pre>

VB Syntax	<pre> oEditor.AddStackupLayers Array("NAME:stackup layer" "Name:=", "<LayerName>", "Type:=", "<Type>", "Top Bottom:=", "<TopBottom>", "Color:=", <Color>, "Transparency:=", <Transparency>, "Pattern:=", <Pattern>, "VisFlag:=", <VisFlag>, "Locked:=", <Locked>, "DrawOverride:=", <Draw>, Array("NAME:Sublayer", "Thick- ness:=", "<Thickness>", "LowerElevation:=", "<LowerElevation>", "Material:=", <Material>")) </pre>
VB Example	<pre> Set oEditor = oDesign.SetActiveEditor("Layout") oEditor.AddStackupLayers Array("NAME:stackup layer", "Name:=", _ "NewLayer", "Type:=", "dielectric", "Top Bottom:=", "neither", "Color:=", _ 3361318, "Transparency:=", 60, "Pattern:=", 1, "VisFlag:=", 127, "Locked:=", _ false, "DrawOverride:=", 0, Array("NAME:Sublayer", "Thickness:=", _ "1.6mm", "LowerElevation:=", "0mm", "Material:=", "FR4_epoxy")) </pre>

ChangeLayer (Layout Editor)

Changes one or more attributes or parameters of a single layer.

Note: To execute the command, all of the following parameters must be entered in the command line even if the user only intends to change one attribute.

UI Access	Limited to showing/hiding element and object visibility in the Layers window.		
Parameters	Name	Type	Description
	<LayerArray>	Array	An array that contains these parameters in order: • <mode> • <Layername> • <ID> • <Type> • <TopBottom> • <Color> • <Transparency> • <Pattern> • <VisFlag> • <Locked> • <Draw> • <Subarray>
	<mode>	String	Type of stackup. Can be "Laminate", "Multizone", or "Overlap".
	<LayerName>	String	Name of layer.
	<ID>	Integer	A numerical identification for the layer.
	<Type>	String	Type of layer, such as "user", "signal", "dielectric", "soldermask", "solderpaste", "silkscreen", "assembly", "wirebond", "dielectric", "glue", "user", "Post-processing", or "outline".
	<TopBottom>	String	<i>Optional</i> . Indicates whether the layer is the top or bottom. Acceptable values

		are "top", "bottom", or "neither".
<Color>	Integer	<i>Optional.</i> A code to indicate the color of the layer. It is an RGB code in the hex format <i>bbggrr</i> . For example red (#0000ff) is 255, green (#00ff00) is 65280, and blue (#ff0000) is 16711680.
<Transparency>	Integer	<i>Optional.</i> A percentage that indicates the transparency of the layer between 0 (opaque) and 100 (completely transparent and not visible).
<Pattern>	Integer	<i>Optional.</i> A code for the pattern of the layer: 0 is hollow, 1 is solid, and the other patterns 3-8 are various hatch patterns.
<VisFlag>	Integer	Sets visibility for all objects on the layer using the following additive code: • 1 makes primitives (rectangles, circles, polygons) visible. • 2 makes lines (also known as paths or traces) visible. • 4 makes pads (from padstacks, aka vias) visible. • 8 makes holes (or vias) visible. • 16 makes components visible. • 32 makes the mesh overlay visible if they are plotted. • 64 makes the mesh for the background material visible. For example, a value of 25 (1+8+16) makes primitives, vias, and components visible. Use 127 to show everything and 0 to hide all.
<Locked>	Boolean	<i>Optional.</i> Whether the layer is locked.
<DrawOverride>	Integer	<i>Optional.</i> Allows drawing on the layer as wireframe, outlines, or solid. Acceptable values are -1 for wireframe, 0 for normal, and 1 for solid.
<Zones>		
<Subarray>	Array	<i>Optional.</i> An array that contains these parameters in order: • <Thickness> • <LowerElevation> • <Roughness> • <BotRoughness> • <SideRoughness>

			• <Material> • <FillMaterial> This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.
	<Thickness>	String	Thickness of the layer, contains the measurement and units.
	<LowerElevation>	String	The elevation of the lower edge of the layer, contains the measurement and units.
	<Roughness>	String	The roughness of the top of the layer, contains the measurement and units.
	<BotRoughness>	String	The roughness of the bottom of the layer, contains the measurement and units.
	<SideRoughness>	String	The roughness of the side of the layer, contains the measurement and units.
	<Material>	String	The material the layer is made of.
	<FillMaterial>	String	The material to use to fill in around geometry on the layer.
Return Value	None		

Python Syntax	<code>oEditor.ChangeLayer(<LayerArray>)</code>
Python Example	<pre>oEditor = oDesign.SetActiveEditor("Layout") oEditor.ChangeLayer(["NAME:stackup layer", "Name:=" , "MQ", "ID:=" , 20, "Type:=" , "signal", "Top Bottom:=" , "neither",</pre>

```
"Color:="          , 6136303,  
"Transparency:="   , 60,  
"Pattern:="        , 1,  
"VisFlag:="        , 96,  
"Locked:="         , False,  
"DrawOverride:="    , 0,  
"Zones:="          , [],  
  ["NAME:Sublayer",  
    "Thickness:="    , "0.745um",  
    "LowerElevation:=" , "737.155um",  
    "Roughness:="    , "0um",  
    "BotRoughness:=" , "0um",  
    "SideRoughness:=" , "0um",  
    "Material:="     , "MQ",  
    "FillMaterial:="  , "FR4_epoxy"  
  ]  
]  
)
```

VB Syntax	<code>oEditor.ChangeLayer <LayerArray></code>
VB Example	<code>Set oEditor = oDesign.SetActiveEditor("Layout")</code>

	<pre>oEditor.ChangeLayer Array("NAME:stackup layer","Name:=" , "MQ", "ID:=" , 20, "Type:=" , "signal", "Top Bottom:=" , "neither", "Color:=" , 6136303, "Transparency:=" , 60, "Pattern:=" , 1, "VisFlag:=" , 96, "Locked:=" , False, "DrawOverride:=" , 0, "Zones:=" , Array(), Array("NAME:Sublayer", "Thickness:=" , "0.745um", "LowerElevation:=" , "737.155um", "Roughness:=" , "0um", "BotRoughness:=" , "0um", "SideRoughness:=" , "0um", "Material:=" , "MQ", "FillMaterial:=" , "FR4_epoxy"))</pre>

ClearLayerMappings (Layout Editor)

Use: Clear layer mappings.

Command: None.

Syntax: ClearLayerMappings <component name>

Return Value: None.

Parameters: <component name> is a string that specifies the name of the component.

VB Example: oEditor.ClearLayerMappings "2"

DuplicateAcrossLyrs (Layout Editor)

Use: Duplicate selected objects (layout and footprint) to other layers.

Command: Draw > Duplicate > Across Layers

Syntax: DuplicateAcrossLyrs Array("NAME:elements", "element-name",...), Array("NAME:layers", "layer-name",...)

Return Value: None

Parameters: <element-name> // The name of the element to be duplicated.

<layer-name> // The name of the layer to duplicate elements to.

VB Example: oEditor.DuplicateAcrossLyrs Array("NAME:elements", "poly_550"), Array("NAME:layers", "Top")

ExportGDSII (Layout Editor)

Exports results to GDSII.

UI Access	File > Export > GDSII.		
Parameters	Name	Type	Description
	<OptionParameters>	Array	Structured array. Array("NAME:options", "FileName:=", <string, GDSII export file name.>, "NumVertices:=", <integer, maximum number of vertices allowed in a polygon (specify zero if unlimited).>, "ArcTol:=", <double, arc tolerance (in meters); the value used to discretize arcs. Smaller the value the greater the number of edges exported. For 0.1mm use 0.0001.>, "LayerMap:=", Array("entry:=", Array(

	<pre> "layer:=", <string, name of a layer to export.>, "id:=", <integer, GDSII layer ID to assign the exported layer.>, "include:=", <boolean, if true, the layer is exported.>)) </pre>
Return Value	None.

Python Syntax	ExportGDSII(<OptionParameters>)
Python Example	<pre>oEditor.ExportGDSII [("NAME:options", "FileName:=", "C:/Projects/ design.gds", "NumVertices:=", 0, "ArcTol:=", 2E-006, "LayerMap:=", ["entry:=", ["layer:=", "Bottom", "id:=", 2, "include:=", true], "entry:=", [</pre>

	<pre> "layer:=", "Top", "id:=", 1, "include:=", true]]]</pre>

VB Syntax	ExportGDSII <OptionParameters>
VB Example	<pre> oEditor.ExportGDSII Array("NAME:options", "FileName:=", "C:/Projects/ design.gds", "NumVertices:=", 0, "ArcTol:=", 2E-006, "LayerMap:=", Array("entry:=", Array("layer:=", "Bottom", "id:=", 2, "include:=", true), "entry:=", Array("layer:=", "Top", "id:=", 1, "include:=", true)))</pre>

ExportStackupXML (Layout Editor)

Exports the stackup in XML format.

UI Access	From the Edit Layers window, select Stackup > Export XML to open an explorer window. Navigate to an appropriate directory, choose a File name , and click Save .		
Parameters	Name	Type	Description
	<SaveAs>	String	User-specified directory in which to save the new stackup XML file.
Return Value	None.		

Python Syntax	ExportStackupXML(<SaveAs>)
Python Example	<code>oEditor.ExportStackupXML("C:\\Users\\USERNAME\\Downloads\\test.xml")</code>

VB Syntax	ExportStackupXML <SaveAs>
VB Example	<code>oEditor.ExportStackupXML "C:\\Users\\USERNAME\\Downloads\\test.xml"</code>

GetAllLayerNames (Layout Editor)

Use: Informational.

Command: None.

Syntax: GetAllLayerNames

Return Value: Array of strings which are the names of all layers in the layout, blackbox, or footprint.

VB Example: None

GetLayerInfo (Layout Editor)

Retrieves information on a specified layer in the Layout editor.

UI Access	None		
Parameters	Name	Type	Description
	<LayerName>	String	Name of the layer
Return Value	An array of strings, as follows: - Type: type of name - TopBottomAssociation: "Top" "Neither" "Bottom" "Template" "Invalid" - Color: integer [representing rgb in hex] - IsVisible: "true" "false". This is true if any these types of objects are visible: - IsVisibleShape: "true" "false"[for stackup layer only] - IsVisiblePath: "true" "false"[for stackup layer only] - IsVisiblePad: "true" "false"[for stackup layer only] - IsVisibleHole: "true" "false"[for stackup layer only] - IsVisibleComponent: "true" "false"[for stackup layer only] - IsLocked: "true" "false" - LayerId: integer [the ID for the layer] The following are also in the array if the layer is a stackup layer: - Index: integer [the stackup order index] - LayerThickness: double [total layer thickness, in meters] - EtchFactor: double [This will not appear if the layer has no etch factor defined.] - IsIgnored: "true" "false" - NumberOfSublayers: 1 [This is always 1.]		

- Material0: material name
- FillMaterial0: material name
- Thickness0: expression with units
- LowerElevation0: expression with units
- If roughness is defined, some or all of the following appear:
 - Roughness0Type: Groiss | Huray
 - Roughness0: value and units or surface ratio value for Huray
 - BottomRoughness0 Type: Groiss | Huray
 - BottomRoughness0: value and units or surface ratio value for Huray
 - SideRoughness0 Type: Groiss | Huray
 - SideRoughness0: value and units or surface ratio value for Huray

For example:

```
[ 'Type: signal', 'TopBottomAssociation: Neither', 'Color: 32512d', 'IsVisible:
true', ' IsVisibleShape: true', ' IsVisiblePath: true', ' IsVisiblePad: true', '
IsVisibleHole: true', ' IsVisibleComponent: true', 'IsLocked: false', 'LayerId:
7', 'Index: 0', 'LayerThickness: 0', 'IsIgnored: false', 'NumberOfSublayers: 1',
'Material0: copper', 'FillMaterial0: FR4_epoxy', 'Thickness0: 0mil', 'Lower-
Elevation0: 62mil', 'Roughness0 Type: Huray', 'Roughness0: 0mil, 2.9', 'Bot-
tomRoughness0 Type: Huray', 'BottomRoughness0: 0mil, 2.9', 'SideRoughness0 Type:
Huray', 'SideRoughness0: 0mil, 2.9']
```

Python Syntax	GetLayerInfo(<layer_name>)
Python Example	<pre>oEditor = oDesign.SetActiveEditor("Layout") oEditor.GetLayerInfo("TopLayer")</pre>

VB Syntax	GetLayerInfo <layer_name>
VB Example	<pre>Set oModule = oDesign.GetModule("Layout") oModule.GetLayerInfo "TopLayer"</pre>

GetStackupLayerNames (Layout Editor)

Use: Informational.

Command: None.

Syntax: GetStackupLayerNames

Return Value: array of strings which are the names of all layers in the layout, blackbox, or footprint.

Parameters: None

ImportStackupXML (Layout Editor)

Imports an XML file into an active project.

UI Access	From the Edit Layers window, select Stackup > Import XML .		
Parameters	Name	Type	Description
	<XMLFileName>	String	Full path to XML file.
Return Value	None.		

Python Syntax	ImportStackupXML(<XMLFileName>)

Python Example	<pre>oProject = oDesktop.SetActiveProject("Project_Name") oDesign = oProject.SetActiveDesign("DesignName") oEditor = oDesign.SetActiveEditor("Layout") oEditor.ImportStackupXML("C:/Files/example.xml")</pre>

VB Syntax	ImportStackupXML <XMLFileName>
VB Example	<pre>oProject = oDesktop.SetActiveProject "Project_Name" oDesign = oProject.SetActiveDesign "DesignName" oEditor = oDesign.SetActiveEditor "Layout" oEditor.ImportStackupXML "C:/Files/example.xml"</pre>

RemoveLayer (Layout Editor)

UI Access	From the Layers window, right-click the target layer and select Remove .

General Mode

Removes the target layer or target stackup layer.

Parameters			
	Name	Type	Description

	<code><LayerName></code>	String	Name of the target layer.
Return Value	None.		

Python Syntax	<code>RemoveLayer(<LayerName>)</code>		
Python Example	<code>oEditor.RemoveLayer("Ground")</code>		

VB Syntax	<code>RemoveLayer <LayerName></code>		
VB Example	<code>oEditor.RemoveLayer "Ground"</code>		

IC Mode (IC Editor)

Removes all objects from the target layer or remove all objects *and* the target layer.

Parameters	Name	Type	Description
	<code><Parameters></code>	Array	Structured array containing these parameters in order: • <code><NAME:Contents></code> • <code><NAME></code> • <code><RemoveLayer></code>
	<code><NAME:Contents></code>	String	Required first parameter which must have the specified value (i.e., "Contents").
	<code><LayerName></code>	String	Name of the target layer.
	<code><RemoveLayer></code>	Boolean	True removes the layer and all objects on the target layer. False removes only all objects.

Return Value	None.

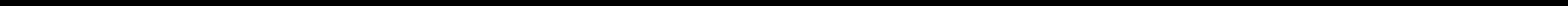
Python Syntax	RemoveLayer([<NAME:Contents>, <LayerName>], <RemoveLayer>)
Python Example	<pre>oEditor.RemoveLayer(["NAME:Contents", "Ground"], False)</pre>

VB Syntax	RemoveLayer(<NAME:Contents>, <LayerName>), <RemoveLayer>
VB Example	<pre>oEditor.RemoveLayer Array("NAME:Contents", "Ground",) , False</pre>

SetLayerMapping (Layout Editor)

Use: Set layer mapping.

Command: None.



Syntax: SetLayerMapping <component name>, <design layer>, <footprint layer>

Return Value: None.

Parameters: <component name> is a string that specifies the name of the component

<design layer> is a string that specifies the name of the design layer

<footprint layer> is a string that specifies the name of the footprint layer

VB Example: oEditor.SetLayerMapping "2", "Bottom Signal", "Top"

SyncRigidFlexSandbox (Layout Editor)

Create or modify geometry and define zones and rigid-flex bends in a design. Existing elements not included in the argument are deleted.

UI Access	None		
Parameters	Name	Type	Description
	<Contents>	Array	Structured Array containing: Array("Name:contents", < Geometry1Array <div>,<geometry2array>,<geometry3array>...< div=""> Note: The number and order of geometries and bends will differ for each design and at the user's discretion. The order of zones in the array implies the Zone Order parameter in Electronics Desktop. </div>,<geometry2array>,<geometry3array>...<>

	<Geometry>	Array	Structured Array containing: <pre>"Name:=", <string zone# name>, "LayerName:=", "Outline",</pre> Note: Depending on the type of geometry ((i.e., polygon, rect-angle, circle, line, or board bend), additional parameters may be applicable. For example, adding "Is Fixed:=true within a <Geometry> array will assign the Fixed status to the primitive. The primitive will then become a fixed reference position during the bending process. Fixing more than one primitive in a design is outside the scope of this function and results are therefore unpredictable.
Return Value	None.		
Python Syntax	SyncRigidFlexSandbox (<Contents>, <Geometry>...)		
Python Example			
VB Syntax	SyncRigidFlexSandbox <Contents>, <Geometry>...		
VB Example			

Net and NetClass Methods

The following net and net class primitives are available:

[CreateDifferentialPair](#)

[CreateNetClass](#)[DeleteNets](#)[DelNetClass](#)[GetNetClasses](#)[GetNetClassNets](#)[GetNetConnections](#)[GetNets](#)[HighlightNet](#)[ModifyDifferentialPair](#)[ModifyNetClass](#)[ResetDifferentialPairs](#)[SelectNetConnected](#)[SelectNetConnected](#)[SetNetColor](#)[SetNetVisible](#)[ToggleNetHighlight](#)

CreateDifferentialPair (Layout Editor)

Creates a differential pair from two target nets.

UI Access	From the Nets window, Ctrl+click two nets from the list. Then right-click within the Nets window and select Create Differential Pair .

Parameters	Name	Type	Description
	<DiffPairName>	String	Name of the new differential pair being created (e.g., DiffPair1)
	<Description>	String	Redundant parameter, intentionally left blank.
	<Parameters>	Array	Array containing two target net names: • <Net1> • <Net2>
	<Net1>	String	One of two target nets used to create the new differential pair (e.g., GND).
	<Net2>	String	One of two target nets used to create the new differential pair (e.g., Port1).
Return Value	None		

Python Syntax	CreateDifferentialPair(<DiffPairName>, <Description>, [<Net1>, <Net2>])
Python Example	oEditor.CreateDifferentialPair("DiffPair1", "", ["GND", "Port1"])

VB Syntax	CreateDifferentialPair <DiffPairName>, <Description>, (<Net1>, <Net2>)
VB Example	oEditor.CreateDifferentialPair "DiffPair1", "", Array("GND", "Port1")

CreateNetClass

*Use:*Create a new net class into a layout.

*Command:*CreateNetClass.

*Syntax:*CreateNetClass (<name> <description> <nets name list>)

*Return Value:*None

*VB Example:*oEditor.CreateNetClass "power", "power nets", Array("A_D2D_VDD_0", "A_D2D_VDD_1", "A_D2D_VSS_0")

DeleteNets (Layout Editor)

Removes specified nets from the project.

UI Access	None.		
Parameters	Name	Type	Description
	<netName>	String	Name of unwanted net.
Return Value	None.		

Python Syntax	DeleteNets ([<netName>, ...])
Python Example	<pre>oEditor.DeleteNets (["NET_1", "NET_2", "NET_3"])</pre>

VB Syntax	DeleteNets Array (<netName>, ...)

VB Example	<pre>oEditor.DeleteNets Array("NET_1") oEditor.DeleteNets Array("NET_2", "NET_3", "NET_4")</pre>

DelNetClass

*Use:*Delete a net class in a layout.

*Command:*DelNetClass

*Syntax:*DelNetClass (<names>)

*Return Value:*None

*VB Example:*oEditor.DelNetClass Array ("power")

GetNetClasses

*Use:*Gets all the net classes in a layout.

*Command:*GetNetClasses

*Syntax:*GetNetClasses ()

*Return Value:*None

*VB Example:*oEditor. GetNetClasses

GetNetClassNets

*Use:*Gets the list of nets in a net class.

Command: GetNetClassNets

Syntax: GetNetClassNets (<name>)

Return Value: None

VB Example: oEditor. GetNetClassNets Array ("power")

GetNetConnections (Layout Editor)

Use: Informational.

Command: None.

Syntax: GetNetConnections(<netName>)

Return Value: Array of strings containing the connections for the net identified by the

netName argument. The strings in the array are in one of four formats:

"ComponentPin compID pinname x y EdgePort"

"ComponentPin compID pinname x y PinPadstack: padstackName"

"InterfacePort portname portID x y Padstack: padstackName"

"EdgePort portname portID EdgeInfo: Primitive id, edge index[Primitive id, edge index]"

where *compID* is the component instance identifier, *pinname* is the name of the connected pin, *x* and *y* are the connection point, *padstackName* is the name of the padstack, *portname* is the name of the port, *portID* is the identifier for the interface port, and *id* and *index* identify edges involved in an edgeport.

Parameters: <netName>

Type: String

The name of the net.

VB Example: `netArray = oEditor.GetNetConnections("net_1")`

GetNets (Layout Editor)

Returns the names of all nets in the active design.

UI Access	N/A.
Parameters	None.
Return Value	Array of the names of the nets in the active design, separated by commas, or None if no nets exist.

Python Syntax	GetNets()
Python Example	<code>oEditor.GetNets() ['gnd', 'net1', 'net2']</code>

VB Syntax	GetNets
VB Example	<code>oEditor.GetNets 'gnd', 'net1', 'net2'</code>

HighlightNet (Layout Editor)

Highlights or removes the highlight from all elements in a net.

UI Access	None.		
Parameters	Name	Type	Description
	<requiredParameterKeyword>	String	Required first parameter which must have the value "Name:Args".
	<requiredNetKeyword>	String	Required second parameter which must have the value "Name:=". Must be the first parameter for each net.
	<netName>	String	Name of the net.
	<requiredFlagKeyword>	String	Required fourth parameter which must have the value "Hi:=". Must be the third parameter for each net.
	<highlightFlag>	Boolean	True if the net should be highlighted. False if it should not.
Return Value	None.		

Python Syntax	HighlightNet ("Name:Args", "Name:=", " <netName>", "Hi:=", <highlightFlag>, ...)
Python Example	<pre>oEditor.HighlightNet (["NAME:Args", "Name:=", "net_0", "Hi:=", True])</pre>

VB Syntax	HighlightNet Array("Name:Args", "Name:=", " <netName>", "Hi:=", <highlightFlag>, ...)
VB Example	<pre>Set oDesign = oProject.SetActiveDesign("HFSS3D1")</pre>

```
Set oEditor = oDesign.SetActiveEditor("Layout")  
oEditor.HighlightNet Array("NAME:Args", "Name:=", "net_0", "Hi:=", false)
```

ModifyDifferentialPair (Layout Editor)

Modifies a differential pair from two target nets.

UI Access	From the Nets window, select the Differential tab. Then right-click a differential pair and select Swap .		
Parameters	Name	Type	Description
	<DiffPairName>	String	Name of the target differential pair (i.e., Sig_0).
	<NewName>	String	New name of the target differential pair (i.e., since it will be unchanged, Sig_0).
	<Description>	String	Redundant parameter, intentionally left blank.
	<Parameters>	Array	Array containing the nets in the differential pair in their new swapped order: • <Net1> • <Net2>
	<Net1>	String	A net in the differential pair, presumably now in the swapped position (e.g., Sig_P).
	<Net2>	String	A net in the differential pair, presumably now in the swapped position (e.g., Sig_N).
Return Value	None		

Python Syntax	ModifyDifferentialPair(<DiffPairName>, <NewName>, <Description>, [<Net1>, <Net2>])
Python Example	oEditor.ModifyDifferentialPair("Sig_0", "Sig_0", "", ["Sig_P", "Sig_N"])

VB Syntax	ModifyDifferentialPair <DiffPairName>, <NewName>, <Description>, (<Net1>, <Net2>)
VB Example	<code>oEditor.ModifyDifferentialPair "DiffPair1", "", Array("GND", "Port1")</code>

ModifyNetClass

Use: Modify an existing net class in a layout.

Command: ModifyNetClass

Syntax: ModifyNetClass (<name> <new name> <new description> <new nets name list>)

Return Value: None

VB Example: `oEditor.ModifyNetClass "power", "power", "new power nets", Array("A_D2D_VDD_0", "A_D2D_VDD_1", "A_D2D_VSS_0", "VDD_DIG_0", "VDD_DIG_1")`

ResetDifferentialpairs (Layout Editor)

Resets a differential pair to its original state.

UI Access	From the Nets window, select the Differential tab. Then right-click a differential pair and select Edit .		
Parameters	Name	Type	Description
	<Parameters1>	Array	Structured array containing these parameters in order: • <NAME:New Differential Pairs> • <DiffPair1> • <DiffPair2>
	<NAME:New Differential Pairs>	String	Required first parameter which must have the specified value (i.e., "New Differential Pairs").
	<DiffPair1>	Array	Array containing the following parameters for the target differential pair in order:

			• <DiffPairName> • <Description1> • <DiffPair1Nets>
	<DiffPair1Name>	String	Name of the target differential pair (i.e., 1).
	<Description1>	String	Redundant parameter, intentionally left blank.
	<DiffPair1Nets>	Array	Array containing the nets in the target differential pair: • <Net1> • <Net2>
	<Net1>	String	The first net in the target differential pair (e.g., Port1).
	<Net2>	String	The second net in the target differential pair (e.g., Sig_P).
	<DiffPair2>	Array	Array containing the following parameters for the target differential pair in order: • <DiffPairName> • <Description2> • <DiffPair2Nets>
	<DiffPair2Name>	String	Name of the target differential pair (i.e., 2).
	<Description2>	String	Redundant parameter, intentionally left blank.
	<DiffPair2Nets>	Array	Array containing the nets in the target differential pair: • <Net3> • <Net4>
	<Net3>	String	The first net in the target differential pair (e.g., Sig_N).
	<Net4>	String	The second net in the target differential pair (e.g., Port2).
Return Value	None		

Python Syntax	ResetDifferentialPairs([<NAME:New Differential Pairs>, [<DiffPair1>, <Description1>, [<Net1>, <Net2>]],

	[<DiffPair2>, <Description2>, [<Net3>, <Net4>]]])
Python Example	<pre> oEditor.ResetDifferentialPairs(["NAME:New Differential Pairs", ["1", "", ["Port1", "Sig_P"]], ["2", "", ["Sig_N", "Port2"]]]) </pre>

VB Syntax	ResetDifferentialPairs (<NAME:New Differential Pairs>, (<DiffPair1>, <Description1>, (<Net1>, <Net2>)), (<DiffPair2>, <Description2>, (<Net3>, <Net4>)))
VB Example	oEditor.ResetDifferentialPairs Array("NAME:New Differential Pairs", Array("1", "", ("Port1", "Sig_P")), Array("2", "", Array("Sig_N", "Port2")))

SelectNetConnected (Layout Editor)

Select all elements in the active design connected to the target net.

UI Access	From the Layout tab, right-click an object in the active design and select Nets > Select Net Connected .		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array containing these parameters in order: • <NAME:elements> • <TargetNet>
	<NAME:elements>	String	Required first parameter which must have the specified value (i.e., "elements").
	<TargetNet>	String	Name of the target net.
Return Value	None.		

Python Syntax	SelectNetConnected([<NAME:elements>, <TargetNet>])
Python Example	<pre>oEditor.SelectNetConnected(["NAME:elements", "circle_1288",])</pre>

VB Syntax	SelectNetConnected(<NAME:elements>, <TargetNet>)
VB Example	<code>oEditor.SelectNetConnected Array("NAME:elements", "circle_1288")</code>

SelectPhysicallyConnected (Layout Editor)

Selects all elements physically connected to the target object.

UI Access	From the Layout tab, right-click an object in the active design and select Nets > Select Physically Connected .		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array containing these parameters in order: • <NAME:elements> • <TargetObject>
	<NAME:elements>	String	Required first parameter which must have the specified value (i.e., "elements").
	<TargetObject>	String	Name of the target object.
Return Value	None.		

Python Syntax	SelectPhysicallyConnected([<NAME:elements>, <TargetObject>])
Python Example	<pre>oEditor.SelectPhysicallyConnected(["NAME:elements", "circle_1288",</pre>

	])

VB Syntax	SelectPhysicallyConnected(<NAME:elements>, <TargetObject>)
VB Example	oEditor.SelectPhysicallyConnected Array("NAME:elements", "circle_1288")

SetNetColor (Layout Editor)

Sets the color of a target net.

UI Access	From the Nets window, right-click a net and select List to open the Design List window. From any row, double-click the bar in the Color column to open the Color window. Select a new color, then click OK to change the net's color.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array containing these parameters in order: • <NAME:Args> • <Name> • <Color>
	<NAME:Args>	String	Required first parameter which must have the specified value (i.e., "Args").
	<Name>	String	Name of the target net.
	<Color>	Array	Structured array containing the RGB (i.e., Red, Green, Blue) color code, which the target net will be set to: • <R> • <G>

			• • <Check_Containment> • <Via_Persistence>
	<R>	Integer	Value of red color.
	<G>	Integer	Value of green color.
		Integer	Value of blue color.
Return Value	None.		

Python Syntax	SetNetColor([<NAME:Args>, <Name>, <Color>, [<R>, <G>,]])
Python Example	<pre>oEditor.SetNetColor(["NAME:Args", "Name:=", "Net_1", "Color:=", ["R:=", 0, "G:=", 0, "B:=", 160]]) </pre>

VB Syntax	SetNetColor(<Name:Args>, <Name>, <Color>, (<R>, <G>,))
VB Example	<pre>oEditor.SetNetColor Array("NAME:Args", "Name:=", "Net_1", "Color:=", Array("R:=", 0, "G:=", 0, "B:=", 160)) </pre>

SetNetVisible (Layout Editor)

Shows and highlights or hides specified nets in the Layout Editor. Multiple nets can be highlighted or hidden at once.

UI Access	On the Nets tab of the Nets window, highlight and right click on one or more nets. Click Hide or any of the Show options.		
Parameters	Name	Type	Description
	<requiredParameterKeyword>	String	Required first parameter which must have the value "Name:Args".
	<requiredNameKeyword>	String	Required second parameter which must have the value "Name:=". Must be the first parameter for each net.
	<netName>	String	Name of the net
	<requiredFlagKeyword>	String	Required fourth parameter which must have the value "Vis:=". Must be the third parameter for each net.
	<visibilityFlag>	Boolean	True if the net should be visible. False if it should be hidden.
Return Value	None.		

Python Syntax	SetNetVisible(["Name:Args", "Name:=", "<netName>", "Vis:=", <highlightFlag>, ...])
Python Example	<pre>oEditor.SetNetVisible(["NAME:Args", "Name:=", "Net_1", "Vis:=", True</pre>

	<pre>]) oEditor.SetNetVisible(["NAME:Args", "Name:=", "<NO-NET>", "Vis:=", False, "Name:=", "Ground", "Vis:=", False, "Name:=", "Chassis", "Vis:=", True])])</pre>

VB Syntax	SetNetVisible Array ("Name:Args", "Name:=", "<netName>", "Vis:=", <visibilityFlag>, ...)
VB Example	<pre>oEditor.SetNetVisible Array("NAME:Args", "Name:=", "NET_1", "Vis:=", true) oEditor.SetNetVisible Array("NAME:Args", "Name:=", "<NO-NET>", "Vis:=", false, "Name:=", "Ground", "Vis:=", false, "Name:=", "Chassis", "Vis:=", true)</pre>

ToggleNetHighlight (Layout Editor)

Highlights or removes the highlight from all objects in the nets of the target elements.

UI Access	After selecting one or more elements within the Layout Editor , navigate to Layout > Nets > Toggle Net Highlight .		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array containing these parameters in order: • <NAME:elements> • <Names>
	<NAME:elements>	String	Required first parameter which must have the specified value (i.e., "elements").
	<Names>	String	Names of the target elements to highlight, separated by commas.
Return Value	None.		

Python Syntax	ToggleNetHighlight([<NAME:elements>, <Names>])
Python Example	<pre>oEditor.ToggleNetHighlight(["NAME:elements", "rect_0", "rect_2", "circle_1" "R96-2"])</pre>

VB Syntax	ToggleNetHighlight(<NAME:elements>, <Names>)
VB Example	oEditor.ToggleNetHighlight("NAME:elements", "rect_0", "rect_2", "circle_1" "R96-2")

Port Methods

The following port methods are available:

[AddCircuitRefPort](#)

[AddPortsToAllNets](#)

[AddPortsToNets](#)

[AddRefPort](#)

[AlignPorts](#)

[AssignCircuitRefPort](#)

[AssignRefPort](#)

[ClearRefPort](#)

[CreateCircuitPort](#)

[CreateEdgePort](#)

[CreateNPort](#)

[CreatePortInstancePorts](#)

[CreatePortsOnComponents](#)

[CreatePortsOnComponentsByNet](#)

[GetPortInfo](#)

[RemovePort](#)

[RemovePortsFromAllNets](#)

[RemovePortsFromNet](#)

[RemovePortsOnComponents](#)

[SetHfssExtentsVisible](#)

[SetHfssPortsVisible](#)

AddCircuitRefPort (Layout Editor)

Use: Assign an edge as a reference to an existing port; model as a circuit port.

Command: AddCircuitRefPort

Syntax: AddCircuitRefPort

```
Array(<"port name">, <"port name">, ...),  
Array("NAME:Contents",  
"edge:=", Array(<edge description>), "edge:=", Array(<edge description>), ...)
```

Return Value: None.

Parameters: <edge description> for primitive edges

```
"et:=", "pe", "prim:=", <"prim">, "edge:=", <edge#>
```

```
<"prim">:
```

text that is the primitive name

<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)

```
VB Example: oEditor.AddCircuitRefPort Array("Port3"), Array("NAME:Contents", "edge:=", _  
Array("et:=", "pe", "prim:=", "rect_2", "edge:=", 1))
```

AddPortsToAllNets (Layout Editor)

Use: Adds ports to all the pins in all the nets.

Command: Layout tab under Nets right-click and click **Create Ports**.

Syntax: AddPortsToAllNets

Return Value: None

Parameters: None.

VB Example:

```
oEditor. AddPortsToAllNets
```

AddPortsToNet (Layout Editor)

Use: Add ports to all the pins on the designated nets.

Command: In Layout right-click and click **Port > Create Ports on Net**.

Layout tab under Nets, right-click and click **Create Ports**.

Syntax: AddPortsToNet Array("NAME:Nets", "net-name", ...)

Return Value: None

Parameters: net-name the name of a net; all pins on this particular net receive ports.

VB Example:

```
oEditor.AddPortsToNet Array("NAME:Nets", "CB1", "CB5")
```

AddRefPort (Layout Editor)

Use: Create a reference port from edges and associate with each of the named ports.

Command: (When more than 2 edges are selected.)

Draw > Port > Create

Right-click > Port > Create

Also available through toolbar icon.

Syntax: AddRefPort

```
Array(<"port name">, <"port name">, ...),
```

```
Array("NAME:Contents",
```

```
"edge:=", Array(<edge description>), "edge:=", Array(<edge description>), ...)
```

Return Value: None.

Parameters: <edge description> for primitive edges

```
"et:=", "pe", "prim:=", <"prim">, "edge:=", <edge#>
```

```
<"prim">:
```

text that is the primitive name

<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)

```
VB Example: oEditor.AddRefPort Array("Port3"), Array("NAME:Contents", "edge=",  
Array("et=", "pe", "prim=", "line_998", "edge=", 0))  
oEditor.AddRefPort Array("Port1"), Array("NAME:Contents", "edge=",  
Array("et=", "pse", "sel=", "via_5", "layer=", 10, "sx=", 0.0015, "sy=", 0.0015,  
"ex=", -0.0015, "ey=", 0.0015, "h=", 0, "rad=", 0))
```

AlignPorts (Layout Editor)

Use: Causes automatic alignment of the microwave ports of the components, EdgePorts, and Pins referenced in the argument. In case the list is empty, aligns ALL the microwave ports in the layout.

Command: None.

Syntax: **AlignPorts** Array ("NAME: *elements*",

<object_name>, // 1st object

<object_name>, // 2nd object, if any

...) // etc

Return Value: None.

Parameters: The LayoutComp's ID

```
VB Example: oEditor.Paste Array ("NAME:elements", "1", "2")  
oEditor.Paste Array ("NAME:elements")
```

AssignCircuitRefPort (Layout Editor)

Use: Assign the internal port as a reference to an existing port; model as a circuit port.

Command: AssignCircuitRefPort

Syntax: AssignCircuitRefPort

```
Array(<"port name">, <"port name">, ...),  
Array("NAME:Contents",  
"edge:=", Array(<edge description>), "edge:=", Array(<edge description>), ...)
```

Return Value: None

Parameters: <"layer">

Type: text

Description: layer name.

<"port">

Type: text

Description: a port or pin name.

<edge description> for primitive edges

"et:=", "pe", "prim:=", <"prim">, "edge:=", <edge#>

<"prim">

Type: text

Description: primitive name

<edge#>

Type: integer

Description: 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)

VB Example: oEditor.AssignCircuitRefPort Array("pin_1"), "pin_2"

AssignRefPort (Layout Editor)

Use: Assign the named internal port as a reference for each of the named ports.

Command: AssignRefPort

Syntax: AssignRefPort

Array(<"port name">, <"port name">, ...),

Array("NAME:Contents",

"edge:=", Array(<edge description>), "edge:=", Array(<edge description>), ...)

Return Value: None

Parameters: <"layer">

Type: text

Description: layer name.

<"port">

Type: text

Description: a port or pin name.

<edge description>

for primitive edges

"et:=", "pe", "prim:=", <"prim">, "edge:=", <edge#>

<"prim">

Type: text

Description: primitive name

<edge#>

Type: integer

Description: 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)

VB Example: `oEditor.AssignRefPort Array("pin_1"), "pin_2"`

ClearRefPort (Layout Editor)

Use: Clear references (or referencing) ports from each of the named ports.

Command: Draw > Clear reference Port

Syntax: **ClearRefPort** Array("Port1", ...)

Return Value: None.

VB Example: oEditor.ClearRefPort Array("Port1", ...)

CreateCircuitPort (Layout Editor)

Use: Create a circuit port between two points.

Command: Draw > Create Circuit Ports

Syntax: CreateCircuitPort Array("NAME:Location", "PosLayer:=", "layer name", "X0:=", _
x-value, "Y0:=", y-value, "NegLayer:=", "layer name", "X1:=", x-value, "Y1:=", y-value)

Return Value: None

Parameters: <"layer">

Type: text

Description: layer name.

<"port">

Type: text

Description: a port or pin name.

<edge description>

for primitive edges

```
"et:=", "pe", "prim:=", <"prim">, "edge:=", <edge#>
```

```
<"prim">
```

Type: text

Description: primitive name

```
<edge#>
```

Type: integer

Description: 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)

VB Example: oEditor.CreateCircuitPort Array("NAME:Location", "PosLayer:=", "Top", "X0:=", _
-0.003, "Y0:=", 0.003, "NegLayer:=", "Top", "X1:=", 0.002, "Y1:=", 0.003)

CreateEdgePort (Layout Editor)

Creates an edge port using the specified edges.

UI Access	Draw > Port > Create or Right-click > Port > Create.		
Parameters	Name	Type	Description
	<EdgePortArray>	Array	Identifies the contents of the array (i.e., "Contents") containing the fol-

			lowing arguments: <NAME:Contents> <Edge> <External>
	<NAME:Contents>	String	Identifies the contents of the array (i.e., "Contents").
	<Edge>	Array	Array containing separate arrays for primitive edges and via edges. For primitive edges, array contains: <Et> <Prim> <Edge> For via edges, array contains: <Et> <Sel> <Layers> <Sx> <Sy> <Ex> <Ey> <H> <Rad>

			Note: Array can repeat in the same command.
	<Et>	String	Type of new edge port (i.e., "pe" for primitive edge or "pse" for pad-stack edge or via).
	<Prim>	String	Name of the target primitive.
	<Edge#>	Integer	Edge number of the target primitive.
	<Sel>	String	Name of the target via.
	<Layers>	Integer	Layer ID of the pad of the target via.
	<Sx>	Integer	Starting X coordinate for the arc of the new edge port.
	<Sy>	Integer	Starting Y coordinate for the arc of the new edge port.
	<Ex>	Integer	Ending X coordinate for the arc of the new edge port.
	<Ey>	Integer	Ending Y coordinate for the arc of the new edge port.
	<H>	Integer	Arc height of the new edge port (e.g., enter 0 for a straight edge).
	<rad>	String	Arc size of the new edge port (e.g., enter 0 for a straight edge).
	<External>	Boolean	True — the port is external. False — the port is internal.
	Return Value	Text containing the name of the created edge port. (Returns an empty name if the edge port is not created.)	

Python Syntax	CreateEdgePort(<EdgePortArray>)
Python Example	Primitive edge port: <pre>oEditor.CreateEdgePort ("NAME:Contents", "edge:=", ("et:=", "pe", "prim:=", "rect_167", "edge:=", 0), "edge:=", ("et:=", "pe", "prim:=", "rect_167", "edge:=", 3), "external:=", true))</pre> Via edge port:

	<pre>oEditor.CreateEdgePort ("NAME:Contents", "edge:=", ("et:=", "pse", "sel:=", "via_0", "layer:=", 10, "sx:=", -0.0015, "sy:=", -0.0015, "ex:=", 0.0015, "ey:=", -0.0015, "h:=", 0, "rad:=", 0), "external:=", true))</pre>

VB Syntax	CreateEdgePort<EdgePortArray>
VB Example	Primitive edge port: <pre>oEditor.CreateEdgePort Array("NAME:Contents", "edge:=", Array("et:=", "pe", "prim:=", "rect_167", "edge:=", 0), "edge:=", Array("et:=", "pe", "prim:=", "rect_167", "edge:=", 3), "external:=", true)</pre> Via edge port: <pre>oEditor.CreateEdgePort Array("NAME:Contents", "edge:=", Array("et:=", "pse", "sel:=", "via_0", "layer:=", 10, "sx:=", -0.0015, "sy:=", -0.0015, "ex:=", 0.0015, "ey:=", -0.0015, "h:=", 0, "rad:=", 0), "external:=", true)</pre>

CreateNPort (Layout Editor)

Use: Creates an N-Port definition and component and adds them to the current project, layout, and schematic.

Syntax: CreateNPort<Array>

Return Value: Returns the name of the newly created component.

```
Parameters:  Array("NAME:Contents",  
"definition_name:=", <nport_definition_name>,  
"placement:=", <component_placement>,  
"layer:=", <placement_layer_name>,  
<nport_data_definition>)
```

<nport_definition_name>:

quoted string (name of the component definition)

<component_placement>:

Array("x:=", <value>, // x coordinate

"y:=", <value>, // y coordinate

"scaling:=", <value>) // double

<nport_data_definition> - see the I/O format in TODO

VB Example: oEditor.CreateNPort

```
Array("NAME:Contents",  
"definition_name:=", "NetworkData3",  
"placement:=",
```



```
Array("x:=", "-9mm",  
      "y:=", "-5mm",  
      "scaling:=", "2"),  
      "layer:=", "Symbols",  
      Array("NAME:NPortData",  
            Array("NAME:NetworkData2",  
                  "filelocation:=", "UsePath",  
                  "filename:=", "",  
                  "domain:=", "frequency",  
                  "numberofports:=", 2,  
                  "datamode:=", "Import",  
                  "devicename:=", "", "  
ImpedanceTab:=", 1,  
"NoiseDataTab:=", 1,  
"DCBehaviorTab:=", 1,  
"SolutionName:=", "",  
"dcbehavior:=", "DCOpen",  
"displayformat:=", "MagnitudePhase",  
"datatype:=", "SMatrix",  
"interptype:=", "Linear",
```

```
"extraptype:=", "Same as interpolation",  
"ShowRefPin:=", 0,  
"RefNodeCheckbox:=", 1)))
```

CreatePortInstancePorts (Layout Editor)

Use: Create port on port instances.

Command: Right-Click-Menu > Port > Create

Syntax: CreatePortInstancePorts Array("NAME:elements", "element-name",...)

Return Value: None

Parameters: <element-name>

Type: string

// Name of a port instance on which ports will be created.

VB Example:

```
oEditor.CreatePortInstancePorts Array("NAME:elements", "0:106")
```

CreatePortsOnComponents (Layout Editor)

Use: Create port on port instances of selected components.

Command: Right-Click-Menu > Port > Create Ports on Component

Syntax: CreatePortsOnComponentsArray("NAME:elements", "element-name",...)

Return Value: None

Parameters: <element-name>

Type: string

// Name of a component.

// Ports are created on the unconnected port instances of the selected components.

VB Example:

```
oEditor.CreatePortsOnComponents Array("NAME:elements", "0")
```

CreatePortsOnComponentsByNet (Layout Editor)

UI Access	After selecting one or more components, right-click within the Layout Editor and select Port > Create Ports On Component to open the Create Ports window. Select a net from the Net Filter area, then click OK to close the Create Ports window.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array containing these parameters in order: • <NAME:Components> • <ComponentNames>
	<NAME:Components>	String	Required first parameter which must have the specified value (i.e., "Components").
	<ComponentNames>	String	Names of the target components, separated by commas.
	<Parameters>	Array	Structured array containing these parameters in order: • <NAME:Nets> • <NetNames>
	<NAME:Nets>	String	Required first parameter which must have the specified value (i.e., "Nets").
	<NetNames>	String	Names of the target net(s), separated by commas (if more than one).
	<Type>	String	Required parameter which must have the value "Port".
	<R>	Integer	Value of resistor.

	<L>	Integer	Value of inductor.
	<C>	Integer	Value of capacitor.
Return Value	None.		

Python Syntax	CreatePortsOnComponentsByNet([<NAME:Components>, <ComponentNames>], [<NAME:Nets>, <NetNames>], <Type>, <R>, <L>, <C>)
Python Example	<pre>oEditor.CreatePortsOnComponentsByNet (["NAME:Components", "U0" "U1"], ["NAME:Nets", "trace"], "Port", "0", "0", "0")</pre>

VB Syntax	CreatePortsOnComponentsByNet Array(<NAME:Components>, <ComponentNames>), Array(<NAME:Nets>, <NetNames>), <Type>, <R>, <L>, <C>)
VB Example	<pre>oEditor.CreatePortsOnComponentsByNet ("NAME:Components", "U0", "U1"), ("NAME:Nets", "trace"), "Port", "0", "0", "0"</pre>

GetPortInfo (Layout Editor)

Use: Request information on a port or pin.

Command: None.

Syntax: GetPortInfo<"name">

Return Value: an array of text as follows:

For pins and ports that are not edge ports:

Name=<name>

Type=Pin, Padstack: <padstack definition name>

X=<X coordinate>

Y=<Y coordinate>

ConnectionPoints=<connection description>; <connection description>,...

NetName=<net name>

For edge ports:

Name=<name>

Type=EdgePort

X=<X coordinate>

Y=<Y coordinate>

ConnectionPoints=<connection description>; <connection description>,...

NetName=<net name>

<name>

Text containing the name of the pin or port

<padstack definition name>

Text containing the name of the associated padstack definition

<X coordinate>

Double indicating the X location of the pin or port location

<Y coordinate>

Double indicating the Y location of the pin or port location

<connection description>

< X coordinate> <Y coordinate> Dir:<direction> Layer: <layer name>

<direction>

Either NONE or a double giving the angle in degrees for the connection direction

<layer name>

Name of the layer for the connection point being described.

Example returned values:

Name=Pin1

Type=Pin, Padstack: Padstack

X=0.000744

Y=0.015537

ConnectionPoints= 0.000744 0.015537 Dir:NONE Layer: Top; 0.000744 0.015537

Dir:NONE Layer: Ground; 0.000744 0.015537 Dir:NONE Layer: Bottom

NetName=Pin1

Name=Port1

Type=EdgePort

X=-0.008120

Y=0.004264

ConnectionPoints= -0.008120 0.004264 Dir:90.000000 Layer: Bottom

NetName=Port1

Parameters: <"name">

Text that contains the name of the port or pin for which information is being requested.

VB Example:

```
Dim conns
```

```
conns = oEditor.GetPortInfo("Port1")
```

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")

RemovePortsFromAllNets (Layout Editor)

Use: Removes ports from all the pins in all the nets.

Command: Layout tab under Nets right-click and click **Remove Ports**.

Syntax: RemovePortsFromAllNets

Return Value: None

Parameters: None.

VB Example: oEditor.RemovePortsFromAllNets

RemovePortsFromNet (Layout Editor)

Removes ports from all the pins on the designated nets.

UI Access	In the Layout window, right click the net and click Port > Remove Ports from Net or on the Nets tab of the Nets window, highlight and right click one or more nets and choose Remove Ports .		
Parameters	Name	Type	Description
	< <i>requiredKeyword</i> >	String	Required parameter that must proceed all element names with the value "NAME:Nets".
	< <i>netName</i> >	String	Name of a net. Ports are removed from the port instances of the selected nets.
Return Value	None.		

Python Syntax	RemovePortsFromNet(["NAME:Nets", <netName>, ...])
Python Example	<pre>oEditor.RemovePortsFromNet (["NAME:Nets", "NET_1", "NET_2", "NET_3"])</pre>

VB Syntax	RemovePortsFromNet Array("NAME:Nets", <netName>, ...)
VB Example	<pre>oEditor.RemovePortsFromNet Array("NAME:Nets", "NET_1", "NET_2", "NET_3")</pre>

RemovePortsOnComponents (Layout Editor)

Remove ports on port instances of selected components. Multiple nets can be listed.

UI Access	Right click and choose Port > Remove Ports From Component .		
Parameters	Name	Type	Description
	<requiredKeyword>	String	Required parameter that must proceed all element names with the value "NAME:elements".
	<elementName>	String	Name of a component. Ports are removed from the port instances of the selected components.

Return Value	None.

Python Syntax	RemovePortsOnComponents(["NAME:elements", <elementName>,...])
Python Example	<pre>oEditor.RemovePortsOnComponents (["NAME:elements", "0"])</pre>

VB Syntax	RemovePortsOnComponents Array("NAME:elements", " <element-name>",...)
VB Example	oEditor.RemovePortsOnComponents Array("NAME:elements", "0")

SetHfssExtentsVisible (Layout Editor)

Hides or reveals HFSS extents.

UI Access	From the Layout tab, select HFSS Extents > Show/Hide .		
Parameters	Name	Type	Description
	<Enable>	Boolean	True displays HFSS extents. False hides HFSS extents.
Return Value	None.		

Python Syntax	SetHfssExtentsVisible(<Enable>)
Python Example	<code>oEditor.SetHfssExtentsVisible(False)</code>

VB Syntax	SetHfssExtentsVisible <Enable>
VB Example	<code>oEditor.SetHfssExtentsVisible False</code>

SetHfssPortsVisible (Layout Editor)

Enables or disables target components.

UI Access	From Layout , select Draw Ports/Sources .		
Parameters	Name	Type	Description
	<Enable>	Boolean	True displays HFSS ports. False hides HFSS ports.
Return Value	None.		

Python Syntax	SetHfssPortsVisible(<Enable>)
Python Example	<code>oEditor.SetHfssPortsVisible(False)</code>

VB Syntax	SetHfssPortsVisible <Enable>
VB Example	<code>oEditor.SetHfssPortsVisible False</code>

Miscellaneous Methods

The following miscellaneous methods are available.

[AlignObjects \(Layout Editor\)](#)

[ChangeOptions \(Layout Editor\)](#)

[ChangeProperty \(Schematic Editor and Layout Editor\)](#)

[ClipPlane \(Layout Editor\)](#)

[CopyToHFSS 3D Layout \(Layout Editor\)](#)

[CutOutSubDesign \(Layout Editor\)](#)

[DefeatureObjects \(Layout Editor\)](#)

[DissolveComponents \(Layout Editor\)](#)

[Duplicate \(Layout Editor\)](#)

[Edit3DComponentDefinition \(Layout Editor\)](#)

[EditComponent \(Layout Editor\)](#)

[EditPinGroup \(Layout Editor\)](#)

[EnableComponents \(Layout Editor\)](#)

[EraseMeasurements \(Layout Editor\)](#)

[ExportDXF \(Layout Editor\)](#)

[ExportGDSII \(Layout Editor\)](#)

[ExportGerber \(Layout Editor\)](#)

[ExportNCDrill \(Layout Editor\)](#)

[GetActiveUnits \(Layout Editor\)](#)

[GetComponentInfo \(Layout Editor\)](#)

[GetCompInstanceFromRefDes \(Layout Editor\)](#)

[GetComponentPins \(Layout Editor\)](#)

[GetComponentPinInfo \(Layout Editor\)](#)

[GetEditorName \(Layout Editor\)](#)

[GetMaterialList \(Layout Editor\)](#)

[GetProperties \(Layout Editor\)](#)

[GetPropertyValue \(Layout Editor\)](#)

[GetSelections \(Layout Editor\)](#)

[Heal \(Layout Editor\)](#)

[PageSetup \(Layout Editor\)](#)

[PushExcitations \(Layout Editor\)](#)

[Select \(Layout Editor\)](#)

[SelectAll \(Layout Editor\)](#)

[SetActiveUnits \(Layout Editor\)](#)

[SetPropertyValue \(Layout Editor\)](#)

[StitchLines \(Layout Editor\)](#)

[UnselectAll \(Layout Editor\)](#)

[ZoomToFit \(Layout Editor\)](#)

AlignObjects (Layout Editor)

Use: Align objects, e.g. primitives, relative to each other.

Command: AlignObjects

Syntax: AlignObjects

```
Array("NAME:align", <"alignment">, ...),
```

```
Array("NAME:elements", <"elem">, ...)
```

Return Value: Return Value: None

Parameters: <"alignment">

Type: text

Description: any one of:

"AlignLeft"

"AlignRight"

"CenterHorz" - align centers horizontally

"DistributeCentersHorz" - distribute object centers evenly horizontally

"SpaceEdgesHorz" - space objects equal distances apart horizontally

"AlignTop"

"AlignBottom"

"CenterVert" - align centers vertically

"DistributeCentersVert" - distribute object centers evenly vertically

"SpaceEdgesVert" - space objects equal distances apart vertically

"elem">

Type: text

Description: name of element (primitive, component, etc.) to align

Example:

```
Set oDesign = oProject.SetActiveDesign("HFSS 3D Layout1")  
Set oEditor = oDesign.SetActiveEditor("Layout")  
oEditor.AlignObjects Array("NAME:align", "AlignLeft"), Array("NAME:elements", "circle_2", "rect_1", "poly_3")
```

ChangeOptions (Layout Editor)

Use: Changes options for an existing layout. (Does not change global options specified in the registry.) Only those options being changed need to be specified. Options not specified are not affected.

Command: None.

Syntax: ChangeOptions Array("NAME:options",<OptionData>,...)

Return Value: None

Parameters: <OptionData> can be of varying forms:

Type: <String>

Type: integer

Type: Array(float, float, float, float)

VB Example: oEditor.ChangeOptions Array("NAME:options", _
"MajorSize:=", "20", _
"MinorSize:=", "1", "MajorColor:=", 8421376, _

```
"MinorColor:=", 16776960, _  
"ShowGrid:=", false, "PageExtent:=", _  
Array( -0.3, -0.1, 0.1, 0.1), _  
"background color:=", 4194368, _  
"DefaultToSketchMode:=", true, _  
"fillMode:=", false, "PixelSnapTolerance:=", 22, _  
"SnapTargetVertex_on:=", _  
false, "SnapTargetEdgeCenter_on:=", false, _  
"SnapTargetObjCenter_on:=", _  
true, "SnapTargetEdge_on:=", true, _  
"SnapTargetElecConnection_on:=", true, _  
"SnapTargetIntersection_on:=", true, _  
"SnapTargetGrid_on:=", false, _  
"SnapSourceVertex_on:=", false, _  
"SnapSourceEdgeCenter_on:=", false, _  
"SnapSourceObjCenter_on:=", true, _  
"SnapSourceEdge_on:=", true, _  
"SnapSourceElecConnection_on:=", true, _  
"ConstrainToGrid:=", false _  
"defaultholesize:=", "5mil", _  
"show connection points:=", true, _
```



```
"display vertex labels:=", true, _  
"NetColor:=", 8421440, _  
"rectangle description:=", 1, "snaptoport:=", false, _  
"sym footprint scaling:=", _  
0.385, "primary selection color:=", 32768, _  
"secondary selection color:=", _  
22784, "preview selection:=", true, "anglesnap:=", _  
"59deg", "AllowDragOnFirstClick:=", true, _  
"useFixedDrawingResolution:=", true, _  
"DrawingResolution:=", "0.002mm", "Tol:=", _  
Array(3E-009, 1.5E-008, 1E-012))
```

Note:

An error will be returned if this script command is not used at the top-level hierarchy.

- Global layout defaults are stored in the registry (Layout\Preferences\)
- Names of registry items were selected for backwards compatibility and are consistent with adsn, technology file, and scripting naming
- Local options are both script-able and undo-able
- Global options are neither script-able nor undo-able

Option	Description	Installed default	Registry, Scripting Names,

			Data Type
Arc Drawing Resolution	Controls drawing of segments for arcs - either dynamic and based on current zoom or fixed to specified value	Dynamic	"useFixedDrawingResolution" ¹ "DrawingResolution" ²
Major and Minor Grid lines	Spacing, color, and visibility	The grid is displayed. Spacing based on current default length units. Major grid lines are 10 minor grid lines apart. The line colors are light blue grays, with major grid lines being darker.	"MajorColor" ³ "MinorColor" ³ "MajorSize" ² "MinorSize" ² "ShowGrid" ¹
Drawing Extent	Specifies size and coordinates of the layout	Lower left of the layout is (-0.1, -0.1) and the upper right is (0.1, 0.1)	"PageExtent" ⁴
Background color	Color of the layout background	white	"background color" ³
Sketch Mode	Fill patterns and center lines are not drawn.	off	"DefaultToSketchMode" ¹
Solid Mode	Objects are filled in with solid color.	off	"fillMode" ¹
Draw Connection Points	Display pin symbols in the layout.	off	"show connection points" ¹
Draw Rats	Display lines indicating missing physical connections in nets	on	"draw rats" ¹
Label Vertices	Display property text labeling the vertices of selected polygons and lines.	off	"display vertex labels" ¹
Net Color	Color used for highlighting nets	light yellow	"NetColor" ³

Rectangle Description	Specifies if rectangles are described with two points or center point, width, and height	2 points	"rectangle description" ⁵
Default Hole Size	Default size for actual holes in a PC board	25 mil	"defaultholesize" ²
Align Microwave Components	Snaps components on entry.	on	"snaptoport" ¹
Symbol Footprint Scaling	Used to size symbol footprints placed in layout.	Based on data extent and current length units.	"sym footprint scaling" ⁶
Primary and Secondary Selection Colors	Colors used to indicate the first item selected and other items currently selected.	Primary color is red, secondary color is a darker red	"primary selectioncolor" ³ "secondary selection color" ³
Preview Selection	Highlight the object that would be selected with a left mouse button click in the current location.	off	"preview selection" ¹
Snapping options	Choose snapping sources and targets and the maximum distance (in pixels) for snapping to occur. Constrain edits to grid if desired. Note: Off-grid objects are ignored if ConstrainToGrid is asserted.	Snap distance is 20 pixels. Snapping sources are vertex, edge center, object center, & elec. conn. Snapping targets are vertex, edge center, object center, elec. conn, & grid. Edits are constrained to grid.	"PixelSnapTolerance" ⁷ "SnapTargetVertex_on" ¹ "SnapTargetEdgeCenter_on" ¹ "SnapTargetObjCenter_on" ¹ "SnapTargetEdge_on" ¹ "SnapTargetElecConnection_on" ¹ "SnapTargetIntersection_on" ¹ "SnapTargetGrid_on" ¹ "SnapSourceVertex_on" ¹ "SnapSourceEdgeCenter_on" ¹ "SnapSourceObjCenter_on" ¹ "SnapSourceEdge_on" ¹

			¹ "ConstrainToGrid" ¹
Always Show Merge Layers Dialog	Controls display of the layer merging dialog.	off – Only show dialog when the user must be involved.	"AlwaysShowLayerMergeDlg" ¹
Rotation Increment	During rotation, objects are rotated by multiples of this amount.	Based on current angle units.	"anglesnap" ²
Allow Drag on first click	Selection and move with one left mouse button click.	false	"AllowDragOnFirstClick" ¹

The footnotes in the table above correspond to the following:

¹ = Integer with a value of 1 for on/true and 0 for off/false

² = String containing a amount and units

³ = Integer representing a Color

⁴ = An Array containing (lower left x, lower left y, upper right x, upper right y)

⁵ = Integer with a value of 0 for two points, and 1 for center/width/height

⁶ = String holding a double.

⁷ = Integer

ChangeProperty (Schematic Editor and Layout 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 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>,  
      <NewPropArray>,  
      <ChangedPropsArray>,  
      <DeletedPropsArray>)
```

<PropServersArray>

```
Array("Name:PropServers", <PropServer>,  
      <PropServer>, ...)
```

<NewPropArray>

```
Array("Name:NewProp", <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 **RMxpert>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"))))
```

ClipPlane (Layout Editor)

Truncates graphics at the XY-plane in the +Z direction. Clip planes can be used to truncate graphics in the Layout view, including mesh and fields plots.

UI Access	From the View menu, select Clip Plane , or from the View ribbon, select Add Clipping Plane .
Parameters	None.
Return Value	None.

Python Syntax	ClipPlane()
Python Example	<pre>oEditor = oDesign.SetActiveEditor("Layout") Editor.ClipPlane ("n1, n2, n3, et cetera") success = oEditor.ClipPlane("n1, n2, n3, et cetera")</pre>

VB Syntax	ClipPlane
VB Example	<pre>oEditor = oDesign.SetActiveEditor"Layout" Editor.ClipPlane "n1, n2, n3, et cetera" Set success = oEditor.ClipPlane "n1, n2, n3, et cetera"</pre>

CopyToHFSS 3D Layout (Layout Editor)

Use: Generate a single, flat, HFSS 3D Layout design from a selection of components and/or sub-circuits.

Command: CopyToHFSS 3D Layout

Syntax: CopyToHFSS 3D Layout Array("NAME:elements", <"obj1">, <"obj2">, ...)

Return Value: None

Parameters: <"obj1">

Type: text

// component or sub-circuit instance name.

Example:

```
Set oDesign = oProject.SetActiveDesign("Nexxim1")  
Set oEditor = oDesign.SetActiveEditor("Layout")  
oEditor.CopyToHFSS 3D Layout Array("NAME:elements", "1", "2", "3")
```

CutOutSubDesign (Layout Editor)

Use: Cut out a subdesign.

Command: CutOutSubDesign

Syntax: oEditor.CutOutSubDesign Array(

```
"NAME:Params",  
"Name:=", <"name">,  
"EMDesign:=", <boolean>,  
"SubDesign:=", <boolean>,  
"Within:=", <boolean>,  
"Without:=", <boolean>,  
"AutoGenExtent":=<boolean>,  
"Expansion":=<double>,  
"RoundCorner":=<boolean>,  
"Increments":=<integer>,  
"ExtentSel:=", Array(<"extent-poly">, ...),  
Array("NAME:Nets", "net:=", Array(<"net-name">, <clip>), ...))
```

In place of "ExtentSel:=", Array(<"extent-poly">, ...) an explicit polygon can be used:

```
"Extent:=", Array("cl:=", true,  
"pt:=", Array(U:=", <"units">, "x:=", <double>, "y:=", <double>, ...))
```

Parameters: Name — name of the cutout design.

EMDesign — if true, create an EM design, otherwise create a Nexxim design.

Within — boolean; cut out the interior region.

Without — boolean; cut out the exterior region.

AutoGenExtent — boolean; when true, Circuit disregards both ExtentSel and Extent. Circuit will instead generate a polygonal shape around all nets which are not clipped.

Expansion — double; similar to the Auto Generate Extent dialog box. However, this is always a unitless fraction. By default it is set to .1.

RoundCorner — boolean; identical to the Auto Generate dialog box. By default this is set to true. In general, rounded corners are preferred when the cutout shape has acute or near-acute angles

Increments — integer; this is from the Auto Generate Dialog, and by default is set to true. This can greatly increase the running time and a small increase can make a big difference. It is probably best to experiment with this parameter first, rather than set it arbitrarily.

ExtentSel— an array of extent polygon names.

Extent — alternative to ExtentSel; an explicit polygon defined by coordinates.

cl — must always be true; indicates that the polygon is closed.

U — coordinate units, e.g. "mm"

pt — array of coordinate values.

x — x coordinate value

y — y coordinate value

Nets — the net information.

net — array of net information.

<"net-name">

"<no net trace>" — special value referring to traces not in a net.

"<no net fill>" — special value referring to fill polygons not in a net.

<design>:<net> — net within a particular design.

<net> — net within the active design.

<clip> — boolean true/false; if true, the net is clipped against the extent else the net is included but not clipped.

Return Value: None

Example: Using "extent_poly" as the selection extent:

```
oEditor.CutOutSubDesign Array("NAME:Params",  
"Name:=", "EMDesign1_cutout",  
"EMDesign:=", true,  
"SubDesign:=", false,  
"Within:=", true,  
"Without:=", false,  
"ExtentSel:=", Array("extent_poly", ...),  
Array("NAME:Nets",  
"net:=", Array("<no net trace>", true),  
"net:=", Array("<no net fill>", true),  
"net:=", Array("EMDesign1:GND", true) ... ))
```

Example: Example using an explicit polygon as the selection extent:

```
oEditor.CutOutSubDesign Array("NAME:Params",  
"Name:=", "EMDesign1_cutout",  
"EMDesign:=", true,  
"SubDesign:=", false,  
"Within:=", true,
```

```
"Without:=", false,  
"Extent:=", Array(  
"cl:=", true,  
"pt:=", Array(  
"U:=", "mm",  
"x:=", 0,  
"y:=" 0, ) ),  
Array("NAME:Nets",  
"net:=", Array("<no net trace>", true),  
"net:=", Array("<no net fill>", true),  
"net:=", Array("EMDesign1:GND", true) ... ))
```

Example: That will create a cutout around first the pos and then the neg trace from the Sample Project "Diff_Via" under Open Samples/EM/SI:

```
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("Diff_Via")
Set oDesign = oProject.SetActiveDesign("diffViaNominal")
Set oEditor = oDesign.SetActiveEditor("Layout")

oEditor.CutOutSubDesign Array("NAME:Params", "Name:=", "diffViaNominal_pos", "EMDesign:=", _
true, "SubDesign:=", false, "Within:=", true, "Without:=", false, "AutoGenExtent:=", _
true, "Expansion:=", 0.1, "RoundCorners:=", false, "Increments:=", 1, "ExtentSel:=", Array(),
Array("NAME:Nets", "net:=", Array( _
"diffViaNominal:neg", true), "net:=", Array("diffViaNominal:pos", false)))

oEditor.CutOutSubDesign Array("NAME:Params", "Name:=", "diffViaNominal_neg", "EMDesign:=", _
true, "SubDesign:=", false, "Within:=", true, "Without:=", false, "AutoGenExtent:=", _
true, "Expansion:=", 0.1, "RoundCorners:=", false, "Increments:=", 1, "ExtentSel:=", Array(),
Array("NAME:Nets", "net:=", Array( _
"diffViaNominal:neg", false), "net:=", Array("diffViaNominal:pos", true)))

```

Note that in this example, each sub design should have its own name, otherwise the results are not well defined. Also note that the "false" after the net indicates that it will be used to build the extent outline. "True" means that the net will be included but will be trimmed.

Defeature Objects (Layout Editor)

Use: Remove irrelevant features from a primitive. Vertices that deviate less than the tolerance from a chord are removed; narrow concave regions less than the tolerance wide are also eliminated. The command can also be used to fix self-intersections (only the 'positive' areas are kept).

Command: DefeatureObjects

Syntax: DefeatureObjects Array("NAME:options", "tolerance:=", <value>, "fix:=", <fix>), Array("NAME:elements", <"primitive">, ...)

Return Value: None

Parameters: <"value">

Type: double, e.g. 0.001

Description: tolerance value

<fix>

Type: boolean (true or false)

Description: if true, then self-intersections are fixed.

< primitive >

Type: text

Description: primitive name.

Example:

```
Set oDesign = oProject.SetActiveDesign("HFSS 3D Layout1")
```

```
Set oEditor = oDesign.SetActiveEditor("Layout")
```

```
oEditor.DefeatureObjects Array("NAME:options", "tolerance:=", 1E-006, "fix:=", true), Array  
("NAME:elements", "poly_3", "poly_4", "poly_5")
```

DissolveComponents (Layout Editor)

Dissolves the named components.

UI Access	After selecting one or more vias, right-click within the Layout Editor and select Components > Dissolve .		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array containing these parameters in order:

			• <NAME:elements> • <TargetObjects>
	<NAME:elements>	String	Required first parameter which must have the specified value (i.e., "vias").
	<TargetObjects>	String	Names of the target primitives, vias, or groups to dissolve, separated by commas.
Return Value	None.		

Python Syntax	DissolveComponents([<NAME:elements>, <TargetObjects>])
Python Example	<pre>oEditor.DissolveComponents (["NAME:elements", "poly_801", "ViaGroup_2"])</pre>

VB Syntax	DissolveComponents(<NAME:elements>, <TargetObjects>)
VB Example	oEditor.DissolveComponents Array("NAME:elements", "poly_801", "ViaGroup_2")

Duplicate (Layout Editor)

Use: Duplicates layout objects.

Command: The specified objects are duplicated by the given count with the specified offset.

Syntax: Duplicate Array("NAME:options", "count:=", <count data>), Array("NAME:elements", <element names>), Array(<x>, <y>)

Return Value: None

Parameters: <count data> is the number of sets of new objects to be generated (and does not include the original objects)

<element names> is one or more strings with the names of layout objects

<x> is the x value for the offset

<y> is the y value for the offset

VB Example: oEditor.Duplicate Array("NAME:options", "count:=", 2), Array("NAME:elements", "rect_56"), Array(-0.018, 0.017)

Edit3DComponentDefinition

Edits definitions of a specified 3D component.

UI Access	Layout > 3D Component Definitions. Select component and click Edit Definition .		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. Array("NAME:3D Components", "Definitions:=" , [<string component name>], "IsLocal:=" , <boolean True for local definition; False to save to file>, "EditFilePath:=", <string path of new component file location>)

			If saving locally or leaving the file path unaltered, pass empty string for EditFilePath.
Return Value	None.		
Python Syntax	Edit3DComponentDefinition (<Parameters>)		
Python Example	<pre>oEditor.Update3DComponentDefinitions(["NAME:3D Components", "Definitions:=" , ["My_Component"], "IsLocal:=" , True, "EditFilePath:=", ""])</pre>		
VB Syntax	Edit3DComponentDefinition <Parameters>		
VB Example	<pre>oEditor.Update3DComponentDefinitions Array("NAME:3D Components", _ "Definitions:=" , Array("My_Component"), "IsLocal:=" , True, "EditFilePath:=" , "")</pre>		

EditComponent (Layout Editor)

Modifies a current source.

id the selected component to add or remove pins

UI Access	After choosing a component, navigate to the Properties window. From the Pins row, select Edit to open the Edit Component window. From the Edit Component window, remove pins as necessary.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array containing these parameters in order:

			• <NAME:Contents> • <definition_name> • <ref_des:=> • <add> • <modified_elements>
	<NAME:Contents>	String	Required first parameter which must have the specified value (i.e., "Contents").
	<definition_name>	String	Identifies the component (e.g., Component_0).
	<ref_des>	String	Identifies the selected component's reference port (e.g., U0).
	<add>	Boolean	True adds pins False removes pins
	<modified_elements>	Array	Structured array containing the names of pins to add or remove (e.g., via_246, via_247).
Return Value	None.		

Python Syntax	EditComponent([<NAME:Contents>, <definition_name>, <ref_des>, <add>, <modified_elements>])
Python Example	<pre>oEditor.EditComponent(["NAME:Contents", "definition_name=", "Component_0", "ref_des=", "U0", "add=", "False", "modified_elements", ["via_246", "via_247"]])</pre>

	1)

VB Syntax	EditComponent (<NAME:Contents>, <definition_name>, <ref_des>, <add>, <modified_elements>)
VB Example	oEditor.EditComponent("NAME:Contents", "definition_name:=", "Component_0", "ref_des:=", "U0", "add:=", "False", "modified_elements", ("via_246", "via_247"))

EditPinGroup (Layout Editor)

Allows the user to edit the target pin group.

UI Access	From Draw , select Create/Manage Pin Groups to open the Create/Manage Pin Groups window. Select a group from the Pin Group List , then click Edit Pin Group to open the Edit Pin Group window. Check the adjacent box next to any pins to remove them, then click Delete Pins . Click OK to close the Edit Pin Group window.		
Parameters	Name	Type	Description
	<NAME>	String	Name of the target pin group.
	<Parameters1>	Array	Structured array containing these parameters in order: • <NAME:elements_remove> • <PinsRemove>
	<NAME:elements_remove>	String	Required first parameter which must have the specified value (i.e., "elements_remove").
	<PinsRemove>	String	Names of the pins to remove from the group, separated by commas.
	<Parameters2>	Array	Structured array containing these parameters in order: • <NAME:elements_add> • <PinsAdd>

	<NAME:elements_add>	String	Required first parameter which must have the specified value (i.e., "elements_add").
	<PinsAdd>	String	Names of the pins to add to the group, separated by commas.
Return Value	None.		

Python Syntax	EditPinGroup(<NAME>, [<NAME:elements_remove>, <PinsRemove>], [<NAME:elements_add>, <PinsAdd>])
Python Example	<pre>oEditor.EditPinGroup("GND_30_SQFP28X28_208_U1", ["NAME:elements_remove", "U1-50", "U1-82"], ["Name:elements_add", "U1-51"])</pre>

VB Syntax	EditPinGroup <NAME>, (<NAME:elements_remove>, <PinsRemove>), (<NAME:elements_add>, <PinsAdd>)
VB Example	<pre>oEditor.EditPinGroup "GND_30_SQFP28X28_208_U1", Array("NAME:elements_remove", "U1-50", "U1-82"), Array("Name:elements_add", "U1-51")</pre>

EnableComponents (Layout Editor)

Enables or disables target components.

UI Access	From the Components window, right-click a component and select Enable/Disable or Ctrl+click multiple components, then right-click any of the selected components and click Enable/Disable .		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array containing these parameters in order: • <NAME:Components> • <ObjectName>
	<NAME:Components>	String	Required first parameter which must have the specified value (i.e., "Components").
	<ObjectName>	String	List of target objects, separated by commas.
	<Enable>	Boolean	True enables selected components. False disables selected components
Return Value	None.		

Python Syntax	EnableComponents ([<NAME:Components>, <ObjectName>], (<Enable>))
Python Example	<pre>oEditor.EnableComponents (["NAME:Components", "U0", "U1"], False)</pre>

VB Syntax	EnableComponents (<NAME:Components>, <ObjectName>), <Enable>
VB Example	<code>oEditor.EnableComponents Array("NAME:Contents", "U0", "U1"), False</code>

EraseMeasurements (Layout Editor)

Use: Causes erasing of ALL measurements currently present.

Command: None.

Syntax: EraseMeasurements

Return Value: None.

Parameters: None.

VB Example: `oEditor.EraseMeasurements`

ExportDXF (Layout Editor)

Use: Export DXF

Command: File > Export > AutoCAD

Syntax: ExportDXF

`Array("NAME:options", "FileName:=", <"filename">, "AcadVersion:=", <version>, "ScaleFactor:=", <scale>),
Array("NAME:layers", <"layer">, ...)`

Return Value: None

Parameters:

`<"filename">`

Type: text

Description: file path for the export file

<version>

Type: integer

Description: version of AutoCAD. Acceptable values are:

- **0** corresponding to AutoCAD version 13
- **1** corresponding to AutoCAD version 14
- **2** corresponding to AutoCAD 2000-2003
- **3** corresponding to AutoCAD 2004-2006
- **4** corresponding to AutoCAD 2007-2009
- **5** corresponding to AutoCAD 2010-2012

<scale>

Type: double

Description: the scaling of the values written out, e.g. if the data is written out in "mm", the scaling factor will be 1000.

<"layer">

Type: text

Description: names of the layers to be exported.

VB Example:

```
Set oDesign = oProject.SetActiveDesign("HFSS3D1")
```

```
Set oEditor = oDesign.SetActiveEditor("Layout")
```

```
oEditor.ExportDXF Array("NAME:options", "FileName:=", "C:/Projects/Temp/output.dxf", "AcadVersion:=", 0, "ScaleFactor:=", 1000), Array("NAME:layers", "Assembly Bottom", "Assembly Top", "Bottom", "Bottom Dielectric", "Errors", "Ground", "Measures", "Rats", "Silkscreen Bottom", "Silkscreen Top", "Symbols", "Top", "Top Dielectric")
```

Python Example:

```
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("05514795_y1_10kv_v02-1mm")
oDesign = oProject.SetActiveDesign("PCB")
oEditor = oDesign.SetActiveEditor("Layout")
oEditor.ExportDXF(
    [
        "NAME:options",
        "FileName:=" , "E:/Anstranlator/resultsEDB/05514795_y1_10kv_v02-1mm_PCB.dxf",
        "AcadVersion:=" , 5,
        "ScaleFactor:=" , 1000
    ],
    [
        "NAME:layers",
        "BOTTOM",
        "Outline",
        "TOP"
    ]
)
```

ExportGDSII (Layout Editor)

Exports results to GDSII.

UI Access	File > Export > GDSII.

Parameters	Name	Type	Description
	<code><OptionParameters></code>	Array	Structured array. <pre> Array("NAME:options", "FileName:=", <string, GDSII export file name.>, "NumVertices:=", <integer, maximum number of vertices allowed in a polygon (specify zero if unlimited).>, "ArcTol:=", <double, arc tolerance (in meters); the value used to discretize arcs. Smaller the value the greater the number of edges exported. For 0.1mm use 0.0001.>, "LayerMap:=", Array("entry:=", Array("layer:=", <string, name of a layer to export.>, "id:=", <integer, GDSII layer ID to assign the exported layer.>, "include:=", <boolean, if true, the layer is exported.>)) </pre>
Return Value	None.		

Python Syntax	<code>ExportGDSII(<OptionParameters>)</code>
Python Example	<code>oEditor.ExportGDSII [(</code>

	<pre>"NAME:options", "FileName:=", "C:/Projects/ design.gds", "NumVertices:=", 0, "ArcTol:=", 2E-006, "LayerMap:=", ["entry:=", ["layer:=", "Bottom", "id:=", 2, "include:=", true], "entry:=", ["layer:=", "Top", "id:=", 1, "include:=", true]]]</pre>

VB Syntax	ExportGDSII <OptionParameters>
VB Example	<pre>oEditor.ExportGDSII Array("NAME:options", "FileName:=", "C:/Projects/ design.gds",</pre>

```

"NumVertices:=", 0,
"ArcTol:=", 2E-006,
"LayerMap:=", Array(
  "entry:=", Array(
    "layer:=", "Bottom",
    "id:=", 2,
    "include:=", true),
  "entry:=", Array(
    "layer:=", "Top",
    "id:=", 1,
    "include:=", true)))

```

ExportGerber (Layout Editor)

Use: Export the current design to Gerber files.

Command: File > Export > Gerber

Syntax: ExportGerber

```

Array("NAME:options",
  "FileName:=", <"file name">,
  "Units:=", <"units">,
  "IntPlace:=", <integer places>,
  "DecimalPlace:=", <decimal places>,
  "Suppress:=", <"suppress">,

```

```
Array("NAME:GerberPages",  
      "<page>:=", Array("<layer>", ...), ...))
```

Return Value: None

Parameters: "<file name>"

Type: text

Description: prefix for the exported Gerber files; the actual files will be named <file name>_<page>.ger

<"units">

Type: text

Description: unit code, e.g. "in", "mm", etc.

<integer places>

Type: integer

Description: number of integer places to use when formatting the numerical output.

<decimal places>

Type: integer

Description: number of decimal places to use when formatting the numerical output.

<"suppress">

Type: text

Description: either "Leading Zeros", or "Trailing Zeros".

<page>

Type: integer

Description: Gerber page number

<"layer">

Type: text

Description: Layers to be exported to the specified Gerber page.

Example:

```
Set oDesign = oProject.SetActiveDesign("HFSS3D1")
Set oEditor = oDesign.SetActiveEditor("Layout")
oEditor.ExportGerber
Array("NAME:options",
"FileName:=", "C:/ Projects/design.ger",
"Units:=", "in",
"IntPlace:=", 5,
"DecimalPlace:=", 5,
"Suppress:=", "Leading Zeros",
Array("NAME:GerberPages", "1:=", Array("Ground"), "2:=", Array("Top") ))
```

ExportNCDrill (Layout Editor)

Use: Export the vias to an NC Drill file

Command: File > Export > NCDrill

Syntax: ExportNCDrill

```
Array("NAME:options",
```

```
"FileName:=", <"file name">,
"Units:=", <"units">,
"IntPlace:=", <integer places>,
"DecimalPlace:=", <decimal places>,
"SuppressLeadingZero:=", <suppress leading zeros>,
Array("NAME:NCDFileOptsVec",
Array("NAME:NCDrillOptions",
"Top:=", <"layer">,
"Bottom:=", <"layer">,
"Create:=", <create>), ...))
```

Return Value: None

Parameters: <"file name">

Type: text

Description: prefix for the exported Gerber files; the actual files will be named <file name>_<page>.ger

<"units">

Type: text

Description: unit code, e.g. "in", "mm", etc.

<integer places>

Type: integer

Description: number of integer places to use when formatting the numerical output.

<decimal places>

Type: integer

Description: number of decimal places to use when formatting the numerical output.

<suppress leading zeros>

Type: boolean

Description: if true, then suppress leading zeros in the output file.

<layer>

Type: text

Description: Start/end layer names; each layer range is written to a separate NC drill file.

<create>

Type: boolean

Description: if true, create the corresponding NC drill file.

Example:

```
Set oDesign = oProject.SetActiveDesign("HFSS3D1")
Set oEditor = oDesign.SetActiveEditor("Layout")
oEditor.ExportNCDrill
Array("NAME:options",
"FileName:=", "C:/ Projects/drill.ncd",
"Units:=", "mm",
"IntPlace:=", 5,
"DecimalPlace:=", 5,
```

```
"SuppressLeadingZero:=", true,  
Array("NAME:NCDFileOptsVec",  
Array("NAME:NCDrillOptions",  
"Top:=", "Top",  
"Bottom:=", "Ground",  
"Create:=", true),  
Array("NAME:NCDrillOptions",  
"Top:=", "Top",  
"Bottom:=", "Bottom",  
"Create:=", true)))
```

GetActiveUnits (Layout Editor)

Returns the current active unit of measurement.

UI Access	N/A.
Parameters	None.
Return Value	Current active unit of measurement (e.g., mm).

Python Syntax	GetActiveUnits()
Python Example	oEditor.GetActiveUnits()

	'mm'

VB Syntax	GetActiveUnits
VB Example	<pre>oEditor.SetActiveUnits 'mm'</pre>

GetComponentInfo (Layout Editor)

Use: Informational.

Command: None.

Syntax: GetComponentInfo<compID>

Return Value: array of strings, as follows:

"ComponentName=*string*"

"PlacementLayer=*string*"

"LocationX=*number*"

"LocationY=*number*"

"BBoxLLx=*number*"

"BBoxLLy=*number*"

"BBoxURx=*number*"

"BBoxURy=*number*"

"Angle=*number*" (*in degrees*)

"Flip=true or false"

"Scale=*number*"

Parameters: The LayoutComp's ID

VB Example: dim info
sys= oEditor.GetComponentInfo ("1")

GetCompInstanceFromRefDes (Layout Editor)

Use: Informational.

Command: None.

Syntax: GetCompInstanceFromRefDes (<object_name>)

Return Value: Returns IDispatch for CompInstance.

Parameters: <object_name> // string is refDes

VB Example: oEditor.GetCompInstanceFromRefDes ("A1")

GetComponentPins (Layout Editor)

Use: Informational.

Command: None.

Syntax: GetComponentPins<compID>

Return Value: array of strings, which are the names of all the component's pins

Parameters: The LayoutComp's ID
CompInst@<ComponentName>;<CompInstID>

VB Example:

' -----

```
' Script Recorded by Ansys Electronics Desktop
```

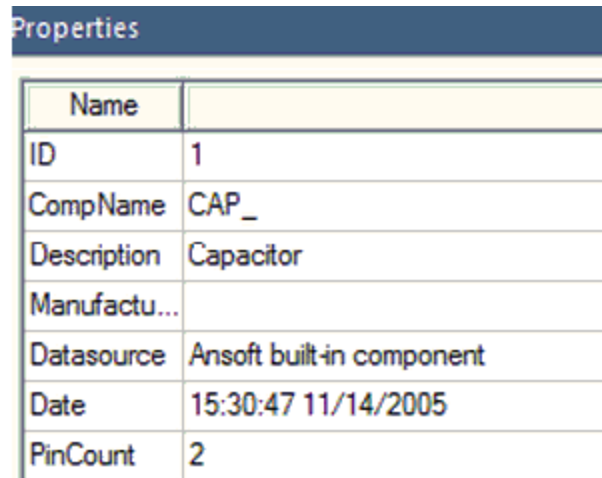
```
' Component Name is CAP_ and ID is 1
```

```
' -----
```

```
dim pins
```

```
pins = oEditor.GetComponentPins ("1")
```

Note: For the documented example, the capacitor component name is CAP_ and its ID is 1. If you can select the item of interest in the schematic, the **Properties** window gets updated with the corresponding component name, its ID, and other details as shown in the following figure.



Properties	
Name	
ID	1
CompName	CAP_
Description	Capacitor
Manufactu...	
Datasource	Ansoft built-in component
Date	15:30:47 11/14/2005
PinCount	2

GetComponentPinInfo (Layout Editor)

Use: Informational.

Command: None.

Syntax: GetComponentPinInfo<compID, pinName>

Return Value: retval = array of strings, as follows:

"X=val"

"Y=val"

"Angle=val"

"Flip=true/false"

"WireID=string"

Parameters: compID = The LayoutComp's ID

pinName = The name of the pin

VB Example: dim info
info = oEditor.GetComponentPinInfo ("1", "n1")

GetEditorName (Layout Editor)

Use: Informational.

Command: None.

Syntax: GetEditorName()

Return Value: Returns the name of the editor.

VB Example: dim info
info = oEditor.GetEditorName

GetMaterialList (Layout Editor)

Use: Get the names of all the materials for a layout.

Command: None.

Syntax: GetMaterialList

Return Value: Array of strings.

Parameters: None.

```
VB Example: Dim materialNames
            materialNames = oEditor.GetMaterialList
```

GetProperties (Layout Editor)

Use: 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 (Layout Editor)

Use: 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.GetPropertyValue("BaseElementTab", _  
"rect_1", "Name")
```

GetSelections (Layout Editor)

Returns an array of currently selected objects.

UI Access	N/A
Parameters	None.
Return Value	Array containing object IDs

Python Syntax	GetSelections()
Python Example	<code>oEditor.GetSelections()</code>

VB Syntax	GetSelections
VB Example	<code>oEditor.GetSelections</code>

GetSelectionsInHierarchy (Layout Editor)

Returns an array of currently selected objects including the ones inside a subdesign (i.e., an inactive design).

UI Access	N/A

Parameters	None.
Return Value	Array containing object IDs

Python Syntax	GetSelectionsInHierarchy()
Python Example	<code>oEditor.GetSelectionsInHierarchy()</code>

VB Syntax	GetSelectionsInHierarchy
VB Example	<code>oEditor.GetSelectionsInHierarchy</code>

Heal (Layout Editor)

Heals imprecise geometry.

UI Access	From the Layout ribbon, click Healing .		
Parameters	Name	Type	Description
	<HealArray>	Array	Structured array containing these parameters in order: • <NAME:category> • <Type> • <OptionalArgs> • <Tol>
	<NAME:category>	String	Required first parameter which identifies the category of healing (i.e., " Snap ", " Feature ", or " <i>Repair</i> ", such as <NAME:Snap>.
	<Type>	String	Dependent on <NAME:category>. Identifies the type of healing, per category: Snap — "Point", "Arc", or "Grid"

			Feature — "Voids" or "FloatingBodies" Repair — "Colinear" or "SelfIntersecting"
	<OptionalArgs>	String	Optional parameters (i.e., "Antipads", "ModifyArcs", and/or "InsertPointsInEdges". Multiple <OptionalArgs> can be entered, depending on <NAME:category> and <Type>.
	<Antipads>	Boolean	If <NAME:Feature> and <Type>="Voids", then this optional argument is available: True — voids are converted to antipads False — voids are not converted to antipads
	<ModifyArcs>	Boolean	If <NAME:Snap> and <Point>="Voids", then this optional argument is available: 0 — arcs do not snap to close primitive edges 1 — arcs snap to close primitive edges
	<InsertPointsInEdges>	Boolean	If <NAME:Snap> and <Point>="Voids", then this optional argument is available: 0 — new points will not appear when healed primitives snap together 1 — new points will appear at midpoints between existing edges and at the new edges created when healed primitives snap together
	<Tol>	Integer	Primitives at or within this tolerance are healed.
Return Value	None		

Python Syntax	<code>oEditor.Heal([<NAME:<category>, <Type>, <OptionalArgs>, <Tol>])</code>
Python Example	<pre>oProject = oDesktop.SetActiveProject("GeometryHealing") oDesign = oProject.SetActiveDesign("TestDesign") oEditor = oDesign.SetActiveEditor("Layout")</pre>

	<pre> oEditor.Heal (["NAME:Snap", "Type:=" , "Point", "ModifyArcs:=" , 1, "InsertPointsInEdges:=" , 1, "Tol:=" , "2mm", "Tol:=" , "2mm"]) </pre>

VB Syntax	oEditor.Heal(<NAME:<category>, <Type>, <OptionalArgs>, <Tol>)
VB Example	<pre> oProject = oDesktop.SetActiveProject("GeometryHealing") oDesign = oProject.SetActiveDesign("TestDesign") oEditor = oDesign.SetActiveEditor("Layout") oEditor.Heal Array("NAME:Snap", "Type:=", "Point", "ModifyArcs:=", 1, "InsertPointsInEdges:=", 1, "Tol:=", "2mm", "Tol:=" , "2mm") </pre>

PageSetup (Layout Editor)

Use: Specifies page setup for printing.

Command: **File>Page Setup**

Syntax: PageSetup <ArgArray>

Return Value: None.

Parameters: <Margins>: Page margins in implicit units of inches.

<Border>: Integer value indicating to draw border (1) or not to draw (0).

<DesignVars>: Integer value indicating to draw design vars (1) or not to draw (0).

```
VB Example: Set oProject = oDesktop.GetActiveProject()
Set oDesign = oProject.GetActiveDesign()
Set oEditor = oDesign.GetActiveEditor()
oEditor.PageSetup Array("NAME:PageSetupData", "margins:=", Array("left:=", 550, "right:=", _
550, "top:=", 500, "bottom:=", 500), "border:=", 1, "DesignVars:=", 0)
```

PushExcitations

Allows access to computed excitations for transient and linear frequency solutions. The script command can be accessed from three locations.

UI Access	• Layout Editor • Schematic Editor • Select a Nexxim solution in a 3D Layout design, right click, and choose Push Excitations.		
Parameters	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 =" , <true false>, "Sol:=", "<solution name>")</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", "transient:=", Array("start:=", <start time>, "stop:=", <stop time>, "maxHarmonics:=", <max harmonics>, "winType:=", <window>, ["widthPct:=", <width percentage>], ["kaiser:=", <Kaiser value>], ["correctCoherentGain:=", true]), "Sol:=", "<solution name>")</pre> winType can have the following values:

			• Rectangular • Bartlett • Blackman • Hamming • Hanning • Kaiser • Welch • Weber • Lanczos
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

VB Syntax	<code>PushExcitations "<refdes>", Array("NAME:options", "transient:=", ["CalcThevenin =", <true>], Array("start:=", <start time>, "stop:=", <stop time>, "maxHarmonics:=", <max harmonics>, "winType:=", <window>, ["widthPct:=", <width percentage>,"kaiser:=", <Kaier value>," ["correctCoherentGain:=", true]), "Sol:=", "<solution name>")</code>
VB Example	For a transient solution: <pre>Set oEditor = oDesign.SetActiveEditor("Layout") oEditor.PushExcitations "U3", Array("NAME:options", _</pre>

```

"transient:=", Array("start:=", 0, "stop:=", 5E-005, _
"maxHarmonics:=", 100, "winType:=", "Rectangular", _
"widthPct:=", 100, "kaiser:=", 0, "correctCoherentGain:=", _
true), "Sol:=", "Transient")

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:=", _
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")

```

Select (Layout Editor)

Chooses a target list of objects to make active.

UI Access	N/A.		
Parameters	Name	Type	Description
	<ObjectName>	Array	Array consisting of target object names, separated by commas (e.g., rect_0,

	<table><tr><td></td><td></td><td>rect_1, circle_0).</td></tr></table>			rect_1, circle_0).
		rect_1, circle_0).		
Return Value	None.			

Python Syntax	Select(<ObjectName>)
Python Example	<code>oEditor.Select(['rect_0', 'rect_1', 'circle_0'])</code>

VB Syntax	Select<ObjName>
VB Example	<code>oEditor.Select Array('rect_0', 'rect_1', 'circle_0')</code>

SelectAll (Layout Editor)

Use: Select all elements in the layout editor.

Command: None.

Syntax: SelectAll()

Parameters: None

VB Example: Dim removedDefs

```
removedDefs = oDefinitionEditor.SelectAll()
```

SetActiveUnits (Layout Editor)

Sets the current active unit of measurement to the target unit.

UI Access	N/A		
Parameters	Name	Type	Description
	<SelectedUnits>	Integer	User-specified unit of measurement to set to active (e.g., mm).
Return Value	The previous active unit of measurement.		

Python Syntax	SetActiveUnits(<SelectedUnits>)		
Python Example	<pre>oEditor.SetActiveUnits('um') 'mm'</pre>		

VB Syntax	SetActiveUnits <SelectedUnits>		
VB Example	<pre>oEditor.SetActiveUnits 'um' 'mm'</pre>		

SetPropertyValue (Layout Editor)

Use: Sets the value of one property. This is not supported for properties of the following types: **ButtonProp**, **PointProp**, and **VPointProp**. Only the **ChangeProperty** command can be used to modify these properties. This can be executed by the **oProject**, **oDesign**, or **oEditor** objects.

Command: None

Syntax: SetPropertyValue <PropTab>, <PropServer>, <PropName>, <PropValue>

Return Value: None

Parameters: <PropValue>

Type: String

Contains the value to set the property. The formatting is different depending on what type of property is being edited. Use **GetPropertyValue** for the desired property to see the expected format.

```
VB Example: oEditor.SetPropertyValue _  
"BaseElementTab", "rect_1", _  
"LineWidth", "3mm"
```

StitchLines (Layout Editor)

Use: Stitch together connected or crossing lines into polygons and lines.

Command: StitchLines

Syntax: StitchLines Array("NAME:elements", <"line">, ...)

Return Value: None

Parameters: <"line">

Type: text

Description: line name.

Example:

```
Set oDesign = oProject.SetActiveDesign("HFSS3D1")
```

```
Set oEditor = oDesign.SetActiveEditor("Layout")  
oEditor.StitchLines Array("NAME:elements", "line_1", "line_2", "line_3")
```

UnselectAll (Layout Editor)

Use: Unselect all active selections in the layout editor.

Command: Edit > Unselect All

Syntax: UnselectAll()

Parameters: None

VB Example: oLayout.UnselectAll()

ZoomToFit (Layout Editor)

Use: Set the current schematic zoom to fit the contents of the currently visible page

Command: None

Syntax: ZoomToFit()

Return Value: None

This page intentionally
left blank.

26 - 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:

[AddComponentManager](#)

[AddDataset](#)

[AddMaterial](#)

[AddPadstackManager](#)

[AddSymbolManager](#)

[AreMaterialPropertiesEqual](#)

[AreSurfaceMaterialPropertiesEqual](#)

[ChangeProperty](#)

[CloneMaterial](#)

[ComputeCoreLossCoefficients](#)

[DeleteDataset](#)

[DoesMaterialExist](#)

[EditComponentManager](#)

[EditDataset](#)

[EditMaterial](#)

[EditPadstackManager](#)

[ExportDataset](#)

[ExportFootprintManager](#)

[ExportMaterial](#)

[ExportPadstackManager](#)

[ExportScript](#)

[ExportSymbolManager](#)

[GetProjectMaterialNames](#)

[GetPropertyValue](#)

[ImportDataset](#)

[RemoveComponentManager](#)

[RemoveFootprintManager](#)

[RemoveMaterial](#)

[RemovePadstackManager](#)

[RemoveScript](#)

[RemoveSymbolManager](#)

[SetPropertyValue](#)

[UpdateDefFromBlock](#)

[UpdateDefFromBlockEx](#)

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](#)

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:=", <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> // [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",
""Radiant", "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>],

	<pre>["NAME:Properties", Any combination of the following: "CheckboxProp:=", <CheckBoxInfo>, "TextProp:=", <TextInfo>, "NumberProp:=", <NumberInfo>, "SeparatorProp:=", <SeparatorInfo>, "ValueProp:=", <ValueInfo>, "MenuProp:=", <MenuInfo>, "VPointProp:=", <VPointInfo>, "PointProp:=", <PointInfo>], ["Quantities", "QuantityProp:=", <QuantityPropInfo>...], ["NAME:CosimDefinitions", <CosimDefInfo>, <CosimDefInfo>...]]</pre>
Python Example	<pre>oComponentManager.Add(["NAME:Component", "Info:=", ["Type:=", 0, "NumTerminals:=", 0,</pre>

```
"DataSource:=", "",  
"ModifiedOn:=", 1467910752,  
"Manufacturer:=", "",  
"Symbol:=", "Component",  
"ModelNames:=", "",  
"Footprint:=", "",  
"Description:=", "",  
"InfoTopic:=", "",  
"InfoHelpFile:=", "",  
"IconFile:=", "",  
"Library:=", "",  
"OriginalLocation:=", "Project",  
"IEEE:=", "",  
"Author:=", "",  
"OriginalAuthor:=", "",  
"CreationDate:=", 1467910746,  
"ExampleFile:=", ""],  
"Refbase:=", "U",  
"NumParts:=", 1,  
"ModSinceLib:=", True,  
"CompExtID:=", 2
```

	)

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.		

Python Syntax	AddDataset <DatasetDataArray>
Python Example	<pre>oProject.AddDataset (["NAME:\$ds1", ["NAME:Coordinates", [</pre>

```
        "NAME:Coordinate",
        "X:=", 2,
        "Y:=", 4
    ],
    [
        "NAME:Coordinate",
        "X:=", 6,
        "Y:=", 8
    ]
]
)
oDesign.AddDataset (
[
    "NAME:$ds1",
    [
        "NAME:Coordinates",
        [
            "NAME:Coordinate",
            "X:=", 2,
            "Y:=", 4
```



```
    ],  
    [  
        "NAME:Coordinate",  
        "X:=", 6,  
        "Y:=", 8  
    ]  
]  
]  
)
```

VB Syntax	AddDataset <DatasetDataArray>
VB Example	<pre>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",</pre>

```

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))

```

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- y- t- h- o- n S- y- nt-	AddDefinitionFromBlock(<defBlock>, <defFolderName>, <newTimeStamp>, <replaceExisting>)

ax	
P- yt- h- o- n E- x- a- m- pl- e	<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,
```

```
        "IsLightweight:="          , False

    ])

oDefinitionManager = oProject.GetDefinitionManager()

defBlock = "$begin 'vacuum2' $begin 'AttachedData' $begin 'MatAppearanceData' property_data-
='appearance_data' Red=230 Green=230 Blue=230 Transparency=0.95 $end 'MatAppearanceData'
$end 'AttachedData' simple('permittivity', 1) ModTime=1499970477 $end 'vacuum2'"

added = oDefinitionManager.AddDefinitionFromBlock(defBlock, "Materials", "10101010", True)
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)
materialNameInQuotes = "\"" + addedName + "\""

oEditor.ChangeProperty(

    [

        "NAME:AllTabs",
```

```
[  
  
  "NAME:Geometry3DAttributeTab",  
  
  [  
  
    "NAME:PropServers",  
  
    "Box1"  
  
  ],  
  
  [  
  
    "NAME:ChangedProps",  
  
    [  
  
      "NAME:Material",  
  
      "Value:=", materialNameInQuotes  
  
    ]  
  
  ]  
  
]
```

	]
	])

AddMaterial

Adds a local material.

UI Access	Add Material in the material editor.		
Parameters	Name	Type	Description
	<MaterialParams>	Array	["NAME: <name of the material to be added>", <MatProperty>, <MatProperty>, ...]
	<MatProperty>	Array	For simple material: "<PropertyName>:=", <value> For anisotropic material: ["NAME:<PropertyName>", "property_type=", "AnisoProperty", "unit:=", <Unit>,"

		<pre>"component1:=", <value>, "component2:=", <value>, "component3:=", <value>))]</pre>
<PropertyName>	String	Should be one of the following (depending on the material, design, and solution types): Electromagnetic (Maxwell-exclusive material properties omitted, see Maxwell Scripting help): "permittivity", "permeability", "conductivity", "dielectric_loss_tangent", "magnetic_loss_tangent", "electric_coercivity", "magnetic_coercivity", "saturation_mag", "lande_g_factor", "delta_H", "delta_h_freq", "mass_density" Thermal (including solids, Icepak fluid flow, and Mechanical rotating fluid modeling): "thermal_conductivity", "mass_density", "specific_heat", "thermal_expansion_coefficient", "thermal_material_type", "viscosity", "diffusivity", "molecular_mass", "clarity_type" Structural: "mass_density", "youngs_modulus", "poissons_ratio", "thermal_expansion_coefficient"
<Unit>	String	Possible values (Maxwell-exclusive properties omitted, see Maxwell Scripting Help; other missing entries are unitless):

			<code>conductivity: "siemens/m"</code> <code>saturation_mag: "uTesla", "mTesla", "tesla", "kTesla", "uGauss", "mGauss", "gauss", "kGauss"</code> <code>delta_H: "A_per_meter", "kA_per_meter", "Oe", "kOe"</code> <code>delta_h_frequency: "Hz", "kHz", "MHz", "GHz", "THz", "rps", "per_sec"</code> <code>mass_density: "kg/m^3"</code> <code>thermal_conductivity: "W/m-C"</code> <code>specific_heat: "J/kg-C"</code> <code>youngs_modulus: "N/m^2"</code> <code>thermal_expansion_coefficient: "1/C"</code>
Return Value	None		

Python Syntax	<code>AddMaterial(["NAME:<MaterialName>", <MatProperty>, <MatProperty>, ...])</code>
Python Example	<pre>oDefinitionManager.AddMaterial(["permittivity:=", "2.2", "0.002"]) oDefinitionManager.AddMaterial [("NAME:Material2",_ "dielectric_loss_tangent:=", "44",</pre>

	<pre> Array("NAME:saturation_mag",_ "property_type:=", "AnisoProperty",_ "unit:=", "Gauss",_ "component1:=", "11", _ "component2:=", "22", _ "component3:=", "33"), _ "delta_H:=", "440e")]</pre>

VB Syntax	<pre> AddMaterial Array("NAME:<MaterialName>", <MatProperty>, <MatProperty>, ...)</pre>
VB Example	<pre> oDefinitionManager.AddMaterial Array("permittivity:=", "2.2", "0.002") oDefinitionManager.AddMaterial Array("NAME:Material2",_ "dielectric_loss_tangent:=", "44", Array("NAME:saturation_mag",_ "property_type:=", "AnisoProperty",_ "unit:=", "Gauss",_ "component1:=", "11", _ "component2:=", "22", _ "component3:=", "33"), _</pre>

	"delta_H:=", "440e")

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....>,
 "hle:=", <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>)
```

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:lgn",
```

```
"lay:=", <string>, // definition layer name
```

```
"id:=", <int>, // definition layer id
```

```
"pad:=", <PadInfo>, // pad
```

"ant:=", <PadInfo>, // antipad
"thm:=", <PadInfo>, // thermal 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))
```

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)))
```

AreMaterialPropertiesEqual

Checks whether named material compared to added material data have equivalent major properties.

UI Access	None.		
Parameters	Name	Type	Description
	material_data	<string>	This will be the same format as the AddMaterial() input parameter, material_data. For example, material_data can be something like this.["NAME:Mold_Material", "CoordinateSystemType:=", "Cartesian", "BulkOrSurfaceType:=", 1, ["NAME:PhysicsTypes", "set:=", ["Thermal"]], "thermal_conductivity:=", "0.8", "mass_density:=", "1980", "specific_heat:=", "960"].
	<MaterialName>	<String>	Name of the material to compare with material database
Return Value	Boolean: • True – named materials have equivalent major properties. • False – named materials do not have equivalent major properties.		

Python Syntax	AreMaterialPropertiesEqual(<material_data>, '<MaterialName>')
Python Example	oDefinitionManager.AreMaterialPropertiesEqual(material_data, 'Mold_Material')

VB Syntax	AreMaterialPropertiesEqual <material_data>, <MaterialName>
VB Example	oDefinitionManager.AreMaterialPropetiesEqual material_data, "modified_epoxy"

AreSurfaceMaterialPropertiesEqual

Checks whether named surface material compared to added surface material data have equivalent major properties.

UI Access	None.		
Parameters	Name	Type	Description
	material_data	<string>	This will be the same format as the AddMaterial() input parameter, material_data. For example, material_data can be something like this.["NAME:Mold_Material", "CoordinateSystemType:=", "Cartesian", "BulkOrSurfaceType:=", 1, ["NAME:PhysicsTypes", "set:=", ["Thermal"]], "thermal_conductivity:=", "0.8", "mass_density:=", "1980", "specific_heat:=", "960"]
	<MaterialName>	<String>	Name of the surface material to compare with material database
Return Value	Boolean: • True – named surface materials have equivalent major properties. • False – named surface materials do not have equivalent major properties.		

Python Syntax	AreSurfaceMaterialPropetiesEqual(<material_data>, '<MaterialName>')
Python Example	oDefinitionManager.AreSurfaceMaterialPropertiesEqual(material_data, 'Mold_Material')

VB Syntax	<code>AreMaterialPropertiesEqual <material_data>, <MaterialName></code>
VB Example	<code>oDefinitionManager.AreMaterialPropetiesEqual material_data, "modified_epoxy"</code>

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>,  
      <NewPropArray>,  
      <ChangedPropsArray>,  
      <DeletedPropsArray>)
```

```
<PropServersArray>  
Array("Name:PropServers", <PropServer>,  
      <PropServer>, ...)
```

<NewPropArray>

Array("Name:NewProp", <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"))))
```

CloneMaterial

Clones a local material.

UI Access	N/A		
Parameters	Name	Type	Description
	<matName>	String	Name of existing material.
	<newName>	String	Name for newly cloned material.
Return Value	Boolean: 1 - Material is cloned. 0 - Existing material not found or a conflict with the new material name.		

Python Syntax	CloneMaterial (<matName>, <newName>)
Python Example	oDefinitionManager.CloneMaterial("copper1", "copper3")

VB Syntax	CloneMaterial <matName>, <newName>

VB Example	<code>oDefinitionManager.CloneMaterial "copper1", "copper3"</code>

DeleteDataset

Deletes a specified dataset. This can be executed by the oProject, or oDesign variables.

UI Access	Project > Datasets > Remove.		
Parameters	Name	Type	Description
	<DatasetName>	String	Name of the dataset found in the project.
Return Value	None.		

Python Syntax	<code>DeleteDataset (<DatasetName>)</code>
Python Example	<code>oProject.DeleteDataset ('\$ds1')</code>
	<code>oDesign.DeleteDataset ('\$ds1')</code>

VB Syntax	<code>DeleteDataset <DatasetName></code>
VB Example	<code>oProject.DeleteDataset "\$ds1"</code>
	<code>oDesign.DeleteDataset "\$ds1"</code>

DoesMaterialExist

Checks for the presence of a material in the library by name

UI Access	None.

Parameters	Name	Type	Description
	<MaterialName>	<String>	Name of the material to search for in the material database
Return Value	Boolean: • True – specified material exists. • False – specified material does not exist.		

Python Syntax	DoesMaterialExist(<MaterialName>)
Python Example	<code>oDefinitionManager.DoesMaterialExist("modified_epoxy")</code>

VB Syntax	DoesMaterialExist <MaterialName>
VB Example	<code>oDefinitionManager.DoesMaterialExist "modified_epoxy"</code>

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",

""Radiant", "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 paralel", _ "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>,

```

	<pre> <menuprop:=", "quantityprop:=", <QuantityPropInfo>...], [" "vpointprop:=", <VPointInfo>, " <cosimdefinfo>,="" <cosimdefinfo>...])="" <menuinfo>,="" <="" name:cosimdefinitions",="" pointprop:=", <PointInfo>, [" pre="" quantities",=""> </menuprop:=",></pre>
Python Example	<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"], _ </pre>

```

"Terminal:=", ["base", "base", "A", false, _
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)", _

```

	<pre>"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"]]])</pre>
Python Example 2	<pre>name2 = oComponentManager.Edit ("MyComponent",_ (["NAME:MyOtherComponent","Info:=", ["Type:=", 4294901767, _ "NumTerminals:=", 2, "DataSource:=", "", _ "ModifiedOn:=", 1071096503, "Manufacturer:=", "Ansoft", _ "Symbol:=", "bendo", "Footprint:=", "BENDO", _ "Description:=", "", "InfoTopic:=", "", _ "InfoHelpFile:=", "", "IconFile:=", "", _</pre>

```

"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:=", "" ] ] )
```

EditDataset

Modifies a dataset. This can be executed by the oProject, or oDesign variables.

UI Access	Project > Datasets > Edit.		
Parameters	Name	Type	Description
	<OriginalName>	String	Name of the original dataset.
	<DatasetDataArray>	Array	Data for the modified dataset.
Return Value	None.		

Python Syntax	EditDataset (<OriginalName> <DatasetDataArray>)
Python Example	<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",
```

	<pre> "X:=", 3, "Y:=", 4]]])</pre>

VB Syntax	EditDataset <OriginalName> <DatasetDataArray>
VB Example	<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>

EditMaterial

Modifies an existing material.

UI Access	View/Edit Materials command in the material editor		
Parameters	Name	Type	Description
	<OriginalName>	String	Name of the material before editing.

	<code><MatProperties></code>	Array	Structured array containing material properties: ["NAME:<New material name>", "CoordinateSystemType:=", <string>, "BulkOrSurfaceType:=" , <integer>, ["NAME:PhysicsTypes", "set:=" , <array containing string physics types>], <Optional ModifierDataArray>, "permeability:=" , <string containing value>, "conductivity:=" , <string containing value>, "thermal_conductivity:=", <string containing value>, "mass_density:=" , <string containing value>, "specific_heat:=" , <string containing value>, "youngs_modulus:=" , <string containing value>, "poissons_ratio:=" , <string containing value>, "thermal_expansion_coefficient:=", <string containing value>]]
	<code><Modi-</code>	Array	Optional structured array containing thermal or spatial modifiers:

	<code>fierDataArray></code>	<pre>["NAME:ModifierData", ["NAME:<ThermalModifierData or SpatialModifierData>", "modifier_data:=" , <"thermal_modifier_data" or "spatial_modifier_data">, ["NAME:<all_thermal_modifiers or all_spatial_modifiers>", ["NAME:<modifierName>", "Property::=" , <string property being modified>, "Index::=" , <integer>, "prop_modifier:=" , <"thermal_modifier" or "spatial_modifier">, "use_free_form:=" , <Boolean>, "free_form_value:=" , <string modifier value>,]]]]</pre>
Return Value	None.	

Python Syntax	EditMaterial (<OriginalName>, <MatProperties>)
Python Example	Without Modifiers: <pre>oDefinitionManager.EditMaterial("alumina_92pct", ["NAME:alumina_92pct", "CoordinateSystemType:=" , "Cartesian", "BulkOrSurfaceType:=" , 1, ["NAME:PhysicsTypes", "set:=" , ["Electromagnetic","Thermal","Structural"]], "permittivity:=" , "9.3", "dielectric_loss_tangent:=" , "0.008", "thermal_conductivity:=" , "26", "mass_density:=" , "3720", "specific_heat:=" , "790", "youngs_modulus:=" , "267000000000", "poissons_ratio:=" , "0.26",</pre>

```

    "thermal_expansion_coefficient:=", "7.2e-006"
]
)

With Thermal Modifier:

oDefinitionManager.EditMaterial("copper",
[
    "NAME:copper",
    "CoordinateSystemType:=", "Cartesian",
    "BulkOrSurfaceType:=" , 1,
    [
        "NAME:PhysicsTypes",
        "set:=" , ["Electromagnetic","Thermal","Structural"]
    ],
    [
        "NAME:ModifierData",
        [
            "NAME:ThermalModifierData",
            "modifier_data:=" , "thermal_modifier_data",
            [
                "NAME:all_thermal_modifiers",
                [

```

```
        "NAME:one_thermal_modifier",
        "Property:=" , "permittivity",
        "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:=" , "58000000",
"thermal_conductivity:=" , "400",
"mass_density:=" , "8933",
"specific_heat:=" , "385",
"youngs_modulus:=" , "120000000000",
"poissons_ratio:=" , "0.38",
"thermal_expansion_coefficient:=" , "1.77e-05"
])
```

Transient Solve, Non-linear Drude Data Plasma

```
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("Drude_plasma_parameters_r231")
oDefinitionManager = oProject.GetDefinitionManager()
oDefinitionManager.EditMaterial("Drude",
[
    "NAME:Drude",
    "CoordinateSystemType:=", "Cartesian",
    "BulkOrSurfaceType:="    , 1,
    [
        "NAME:PhysicsTypes",
        "set:="                , ["Electromagnetic"]
    ],
    [
        "NAME:AttachedData",
        [
            "NAME:MatNonLinearDrudeFreqDepData",
            "property_data:="    , "nonlinear_drude_data",
            "EpsilonInfinity:="  , "1",
            "PlasmaFrequency:="  , "4.62348462366278GHz",
```


	<pre>"CollisionFrequency:=" , "0.00054491190162662GHz", "FieldBreakdown:=" , "10000V_per_meter", "PlasmaMaintainFrequency:=" , "2.31174231183139GHz", "NeutralDensity:=" , 2.65164580488373E+20, "ElectronDensity:=" , 2.65164580488373E+17, "CollisionRateConstant:=" , 2.05499505485618E-15]]])</pre>

VB Syntax	EditMaterial <OriginalName>, <MatProperties>
VB Example	Without Modifiers: <pre>oDefinitionManager.EditMaterial "alumina_92pct", Array("NAME:alumina_92pct", "CoordinateSystemType:=", "Cartesian", "BulkOrSurfaceType:=", 1, Array("NAME:PhysicsTypes", "set:=", Array("Electromagnetic", "Thermal", "Structural")),</pre>

```

"permittivity:=", "9.3",
"dielectric_loss_tangent:=", "0.008",
"thermal_conductivity:=", "26",
"mass_density:=", "3720",
"specific_heat:=", "790",
"youngs_modulus:=", "267000000000",
"poissons_ratio:=", "0.26",
"thermal_expansion_coefficient:=", "7.2e-006"
)

```

With Thermal Modifier:

```

oDefinitionManager.EditMaterial "copper",
    Array("NAME:copper",
        "CoordinateSystemType:=", "Cartesian",
        "BulkOrSurfaceType:=" , 1,
        Array("NAME:PhysicsTypes",
            "set:=" , Array("Electromagnetic","Thermal","Structural")),
        Array("NAME:ModifierData",
            Array("NAME:ThermalModifierData",
                "modifier_data:=" , "thermal_modifier_data",
                Array("NAME:all_thermal_modifiers",
                    Array("NAME:one_thermal_modifier",

```

```
        "Property:=" , "permittivity",
        "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:=" , "58000000",
"thermal_conductivity:=" , "400",
"mass_density:=" , "8933",
"specific_heat:=" , "385",
"youngs_modulus:=" , "120000000000",
"poissons_ratio:=" , "0.38",
"thermal_expansion_coefficient:=" , "1.77e-05"
)
```

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....>),  
            "hle:=", <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>)

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
"thm:=", <PadInfo>, // thermal 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.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))

```

ExportDataset

Exports a dataset to a named file. This can be executed by the oProject, or oDesign variables.

UI Access	Project > Datasets > Export.

Parameters	Name	Type	Description
	<datasetFilePath>	String	The full path to the file.
Return Value	None.		

Python Syntax	ExportDataset (<datasetFilePath>)
Python Example	<pre>oProject.ExportDataset('e:/tmp/dsdata.txt') oDesign.ExportDataset('e:/tmp/dsdata.txt')</pre>

VB Syntax	ExportDataset <datasetFilePath>
VB Example	<pre>oProject.ExportDataset "e:/tmp/dsdata.txt" oDesign.ExportDataset "e:/tmp/dsdata.txt"</pre>

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"
```

ExportMaterial

Exports a local material to a library.

UI Access	Export to Library command in the material editor.		
Parameters	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.
	<LibraryLocation>	String	Location to save the library. Only "PersonalLib" and "UserLib" are allowed.

Return Value	None.

Python Syntax	<code>ExportMaterial (<ExportData>, <LibraryLocation>)</code>
Python Example	<pre>oDefinitionManager.ExportMaterial (["NAME:mylib",_ "Material1", "Material2", "Material3"], "PersonalLib")</pre>

VB Syntax	<code>ExportMaterial <ExportData>, <LibraryLocation></code>
VB Example	<pre>oDefinitionManager.ExportMaterial Array("NAME:mylib",_ "Material1", "Material2", "Material3"), "PersonalLib"</pre>

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"
```

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"
```

Python Syntax	ExportScript(<ExportData>,<Library location>)
Python Example	<pre>oProject.ExportComponent (["NAME:mylib", "myscript"], "PersonalLib")</pre>

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"
```

GetProjectMaterialNames

Returns the material names belonging to an active Project.

UI Access	N/A		
Parameters	Name	Type	Description
	None		
Return Value	String names of the materials in the active project.		

Python Syntax	GetProjectMaterialNames()
Python Example	<pre>oProject = oDesktop.GetActiveProject() oDefinitionManager = oProject.GetDefinitionManager() materials = oDefinitionManager.GetProjectMaterialNames() AddWarningMessage(str(materials))</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.

UI Access	N/A		
Parameters	Name	Type	Description
	<i><PropTab></i>	String	One of the following, where tab titles are shown in parentheses: - PassedParameterTab ("Parameter Values") - DefinitionParameterTab (Parameter Defaults") - LocalVariableTab ("Variables" or "Local Variables") - ProjectVariableTab ("Project variables") - ConstantsTab ("Constants") - BaseElementTab ("Symbol" or "Footprint") - ComponentTab ("General") - Component("Component") - CustomTab ("Intrinsic Variables") - Quantities ("Quantities") - Signals ("Signals")
	<i><PropServer></i>	String	An object identifier, generally returned from another script method, such as <code>CompInst@R;2;3</code>
	<i><PropName></i>	String	Name of the property.

Return Value	String value of the property.

Python Syntax	<code>GetPropertyValue (<PropTab>, <PropServer>, <PropName>)</code>
Python Example	<pre>selectionArray = oEditor.GetSelections() for k in selectionArray: val = oEditor.GetPropertyValue("PassedParameterTab", k, "R") ...</pre>

VB Syntax	<code>GetPropertyValue (<PropTab>, <PropServer>, <PropName>)</code>
VB Example	<pre>selectionArray = oEditor.GetSelections for k in selectionArray: val = oEditor.GetPropertyValue("PassedParameterTab", k, "R") ...</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.

Parameters	Name	Type	Description
	<datasetFilePath>	String	The full path to the file containing the dataset values. *.tab files recommended (see note below).
	<optionalDatasetName>	String	<i>Optional.</i> User-defined dataset name.
Return Value	None.		

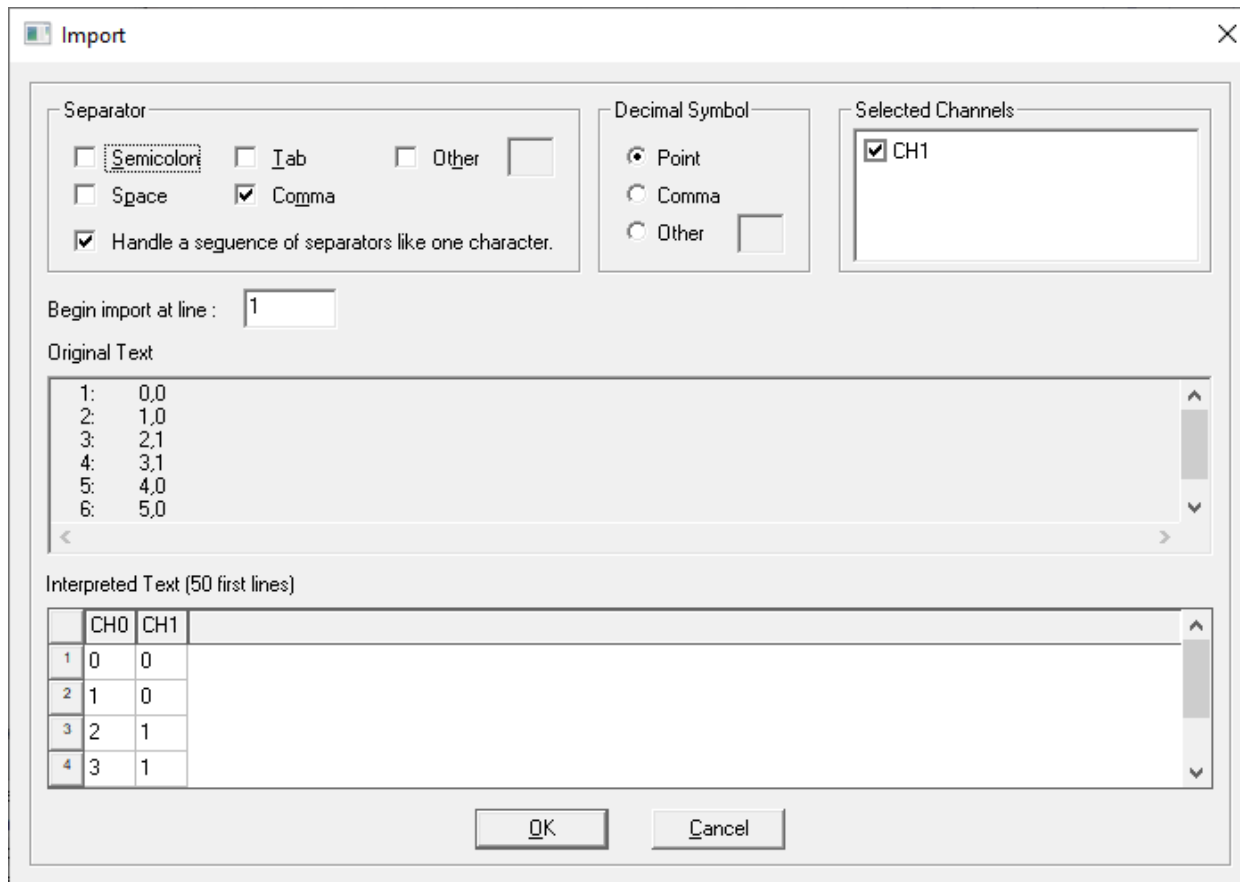
Python Syntax	<code>ImportDataset (<datasetFilePath>,<optionalDatasetName>)</code>
Python Example	<pre>oProject.ImportDataset('e:\tmp\dsdata.tab') oDesign.ImportDataset('e:\tmp\dsdata.tab') oProject.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName') oDesign.ImportDataset('e:\tmp\dsdata.tab', 'MyDatasetName')</pre>

VB Syntax	<code>ImportDataset <datasetFilePath>, <optionalDatasetName></code>
VB Example	<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**.



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"
```

Python Syntax	Remove (<ComponentName>, <IsProjectComponent>, <LibraryName>, <LibraryLocation>)
Python Example	<pre>oComponentManager.Remove ("Simplorer Circuit Elements\BJTs:Level01_NPN", _ true, "Project")</pre>

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"
```

RemoveMaterial

Removes a material from a library.

UI Access	Remove Material(s) command in the material editor		
Parameters	Name	Type	Description
	<MaterialName>	String	Name of the material to be removed.
	<IsProjectMaterial>	Boolean	If True, 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".
Return Value	None.		

Python Syntax	RemoveMaterial (<MaterialName>, <IsProjectMaterial>, <LibraryName>, <LibraryLocation>)

Python Example	<pre>oDefinitionManager.RemoveMaterial (["Material1", false, "mo0907", "UserLib"])</pre>

VB Syntax	<pre>RemoveMaterial <MaterialName>, <IsProjectMaterial>, <LibraryName>, <LibraryLocation></pre>
VB Example	<pre>oDefinitionManager.RemoveMaterial "Material1", false, "mo0907", "UserLib"</pre>

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"
```

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"
```

Python Syntax	RemoveScript (<ScriptName>,<IsProjectScript>, <LibraryName>,<LibraryLocation>)
Python Example	<pre>oDefinitionManager.RemoveScript("myscript", true, "Local", "Project")</pre>

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"
```

RemoveUnusedDefinitions

Removes any unused project definitions.

UI Access	Tools > Project Tools > Remove Unused Definitions.		
Parameters	Name	Type	Description
	<Definitions>	Array	Definitions to be removed, such as materials and surface materials.
Return Value	None.		

Python Syntax	RemoveUnusedDefinitions(<Definitions>)
Python Example	<pre>oProject.RemoveUnusedDefinitions([["NAME:Materials", "Al-Extruded"], ["NAME:SurfaceMaterials", "Steel-oxidised-surface"]])</pre>

VB Syntax	RemoveUnusedDefinitions <Definitions>
VB Example	<pre>oProject.RemoveUnusedDefinitions Array(Array("NAME:Materials", "Al-Extruded"), Array ("NAME:SurfaceMaterials", "Steel-oxidised-surface"))</pre>

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.

UI Access	N/A		
Parameters	Name	Type	Description
	<propTab>	String	One of the following, where tab titles are shown in parentheses: - PassedParameterTab ("Parameter Values") - DefinitionParameterTab (Parameter Defaults") - LocalVariableTab ("Variables" or "Local Variables") - ProjectVariableTab ("Project variables") - ConstantsTab ("Constants") - BaseElementTab ("Symbol" or "Footprint") - ComponentTab ("General") - Component("Component") - CustomTab ("Intrinsic Variables") - Quantities ("Quantities") - Signals ("Signals")
	<propServer>	String	An object identifier, generally returned from another script method, such as CompInst@R;2;3
	<propName>	String	Name of the property.
	<propValue>	String	The value for the property
Return Value	None.		

Python Syntax	SetPropertyValue(<propTab>, <propServer>, <propName>, <propValue>)
Python Example	oEditor.SetPropertyValue("PassedParameterTab", "k", "R", "2200")

VB Syntax	SetPropertyValue <propTab>, <propServer>, <propName>, <propValue>
VB Example	oEditor.SetPropertyValue "PassedParameterTab", "k", "R", "2200"

UpdateDefFromBlock

Updates 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
	<targetDefName>	String	Name of the target definition, i.e. the name of the material to update.
	<defBlock>	String	Text of the new material definition in block format (string); this block could use a new definition name, which will cause a rename as part of the update.
	<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.
Return Value	A property scripting object for the definition.		

P- yt- h- o- n S- y- nt- ax	UpdateDefFromBlock(<targetDefName>, <defBlock>, <defFolderName>, <newTimeStamp>)
P- yt- h- o- n E- x- a- m- pl- e	<pre> oProject = oDesktop.NewProject() oProject.InsertDesign("HFSS", "HFSSDesign1", "DrivenModal", "") oDesign = oProject.SetActiveDesign("HFSSDesign1") oEditor = oDesign.SetActiveEditor("3D Modeler") oDefinitionManager = oProject.GetDefinitionManager() defBlock = "\$begin 'vacuum2' \$begin 'AttachedData' \$begin 'MatAppearanceData' property_data- ='appearance_data' Red=230 Green=230 Blue=230 Transparency=0.95 \$end 'MatAppearanceData' \$end 'AttachedData' simple('permittivity', 1) ModTime=1499970477 \$end 'vacuum2'" added = oDefinitionManager.AddDefinitionFromBlock(defBlock, "Materials", "10101010", True) addedName = '' if isinstance(added, basestring): addedName = added elif isinstance(added, list): addedName = added[0] else: </pre>


```

addedName = added.GetName().replace("Materials:", "")
AddInfoMessage(os.path.basename(__file__) + " result: " + addedName)
materialNameInQuotes = "\"" + addedName + "\""

# rename vacuum2 to vacuum3

newDefBlock = "$begin 'vacuum3' $begin 'AttachedData' $begin 'MatAppearanceData' property_
data='appearance_data' Red=230 Green=230 Blue=230 Transparency=0.95 $end 'MatAp-
pearanceData' $end 'AttachedData' simple('permittivity', 1) ModTime=1499970477 $end
'vacuum3'"

updatedObj = UpdateDefFromBlock(addedName, newDefBlock, "Materials")
    
```

UpdateDefFromBlockEx

Updates 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. This resembles the UpdateDefFromBlock command, except UpdateDefFromBlockEx also allows for:

- updating a definition using a block where the name has changed without triggering a rename of the definition being updated
- updating multiple definitions from the same block
- renaming a definition to something other than the name in the block

UI Access	N/A		
Parameters	Name	Type	Description
	<targetDefName>	String	Name of the target definition, i.e. the name of the material to update.

		Name of the target definition, i.e. the name of the material to update (string). Name of the target definition, i.e. the name of the material to update (string).		
	<desiredDefName>	String	New name to use for a rename. If this is the same as <i>targetDefName</i>, it means the definition is not being renamed. If this is blank, the block name will be used for this purpose instead.	New name to use for a rename. If this is the same as <i>targetDefName</i>, it means the definition is not being renamed. If this is blank, the block name will be used for this purpose instead. New name to use for a rename. If this is the same as <i>targetDefName</i>, it means the definition is not being renamed. If this is blank, the block name will be used for this purpose instead.
	<defBlock>	String	Text of the updated material definition in block form. (Same form as the defBlock input to the AddDefinitionFromBlock and UpdateDefFromBlock commands, for example.) Note that, unlike in the UpdateDefFromBlock command, the name of the definition in the block can be ignored, provided that desiredDefName is set.	
	<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.	

	<continuelIfConflict>	Boolean	A flag for whether to continue processing if there is some sort of conflict. A conflict only comes from the target definition being in use, such as a material assigned to a particular part, and if trying to rename the target definition. If this argument is True, and there's a rename conflict, the tactic is to add/update the definition with the new name and leave the target definition untouched. If this argument is False, and there's a rename conflict, the command results in an exception.
Return Value	A property scripting object (extended definition object) for the definition. (In the event of a failure, the command throws an exception and there will be no result.)		

Python Syntax	UpdateDefFromBlockEx(<targetDefName>, <desiredDefName>, <defBlock>, <defFolderName>, <newTimeStamp>, <continuelIfConflict>)
Python Example	<pre># Note that the material block name will be ignored in this case defBlock = "\$begin 'unused_name' \$begin 'AttachedData' \$begin \ 'MatAppearanceData' property_data='appearance_data' \ Red=255 Green=0 Blue=0 Transparency=0.19 \$end \ 'MatAppearanceData' \$end 'AttachedData' \ simple('permittivity', 1.9) ModTime=1509090909 \ \$end 'unused_name'"</pre>

```
try:
    updated = oDefinitionManager.UpdateDefFromBlockEx("red_material",
        "red_material",
        defBlock,
        "Materials",
        "1609090909")
    updatedName = updated.GetName().replace("Materials:", "")
    AddInfoMessage("Result: " + updatedName)
except:
    AddErrorMessage("Unexpected error in UpdateDefFromBlockEx")
```

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 .
Parameters	None.
Return Value	None.

Python Syntax	UpdateDefinitions()
Python Example	oProject.UpdateDefinitions()

VB Syntax	UpdateDefinitions
VB Example	<code>oProject.UpdateDefinitions</code>

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](#)

[AddNPortData](#)

[AddSolverOnDemandModel](#)

[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:=", <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> // [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

SlwaveDeNovo -- 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",

""Radiant", "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", _  
  ), _  
  "ButtonProp:=", Array("CosimDefinition", _  
    "D", _  
    "", _  
    "", _  
    "Edit", _  
  ), _  
  "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>,
"ValueProp:=", <ValueInfo>,
"MenuProp:=", <MenuInfo>,
"VPointProp:=", <VPointInfo>,

```

	<pre>"PointProp:=", <PointInfo>], ["Quantities", "QuantityProp:=", <QuantityPropInfo>...], ["NAME:CosimDefinitions", <CosimDefInfo>, <CosimDefInfo>...]]</pre>
Python Example	<pre>oComponentManager.Add(["NAME:Component", "Info:=", ["Type:=", 0, "NumTerminals:=", 0, "DataSource:=", "", "ModifiedOn:=", 1467910752, "Manufacturer:=", "", "Symbol:=", "Component", "ModelNames:=", "", "Footprint:=", "", "Description:=", "", "InfoTopic:=", "",</pre>

```

"InfoHelpFile:=", "",
"IconFile:=", "",
"Library:=", "",
"OriginalLocation:=", "Project",
"IEEE:=", "",
"Author:=", "",
"OriginalAuthor:=", "",
"CreationDate:=", 1467910746,
"ExampleFile:=", ""],
"Refbase:=", "U",
"NumParts:=", 1,
"ModSinceLib:=", True,
"CompExtID:=", 2
])

```

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"
"datamode:=", <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".

<dcInfo>:

<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", _  
"datamode:=", "Import", _  
"devicename:=", "", _  
"ImpedenceTab:=", 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",
""Radiant", "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 curents, 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 paralel", _ "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	<pre>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>,</pre>

	<pre> "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>), "VPointProp:=", <VPointInfo>, "PointProp:=", <PointInfo>), ["Quantities", "QuantityProp:=", <QuantityPropInfo>...], ["NAME:CosimDefinitions", <CosimDefInfo>, <CosimDefInfo>...]) </pre>
Python Example	<pre> name = oComponentManager.Edit ("Simplorer Circuit Elements\BJTs:Level01_NPN", _ ["NAME:Level01_NPN", "Info:=", ["Type:=", 4294901764,_ </pre>

```

"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, _
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

```

	<pre> 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"]]]) </pre>
Python Example 2	<pre> name2 = oComponentManager.Edit ("MyComponent",_ (["NAME:MyOtherComponent","Info:=", ["Type:=", 4294901767, _ </pre>

```
"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"
```

Python Syntax	Export(["NAME:<LibraryName>", <ComponentName>, <ComponentName>...], <LibraryLocation>)
Python Example	<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.

Python Syntax	GetData(<DefinitionName>)
Python Example	<code>compData = oComponentManager.GetData("Level01_NPN")</code>

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()
```

Python Syntax	GetNames()
Python Example	<code>componentNames = oComponentManager.GetNames()</code>

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
    End If
End For
```

```
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")
```

Python Syntax	IsUsed(<ComponentName>)
Python Example	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"
```

Python Syntax	Remove (<ComponentName>, <IsProjectComponent>, <LibraryName>, <LibraryLocation>)
Python Example	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.

Python Syntax	RemoveUnused()
Python Example	<code>removedDefs = oComponentManager.RemoveUnused()</code>

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")
```

Footprint Manager Script Commands

The footprint manager provides access to footprints in a project. The manager object is accessed via the definition manager.

```
Set oDefinitionManager = oProject.GetDefinitionManager()
```

```
Set oFootprintManager = oDefinitionManager.GetManager("Footprint")
```

The footprint manager script commands are listed below:

[Add](#)

[Edit](#)

[EditWithComps](#)

[Export](#)

[GetData](#)

[GetNames](#)

[IsUsed](#)

[Remove](#)

[RemoveUnused](#)

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:Lyrs",  
"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:=", "Tracel", _  
"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:=", "Dielecl", _  
"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:Lyrs",  
"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_Footprints: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.

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")
```

Python Syntax	GetData(<SimpleName>)
Python Example	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", "per-
mittivity:=", "0.123") )
oMaterialMgr.Add( Array("NAME:MyMaterial2", "CoordinateSystemType:=", "Cartesian", "per-
mittivity:=", "0.456") )
Dim materialNames
materialNames = oMaterialMgr.GetNames()
```

Python Syntax	GetNames()
Python Example	materialnames = oMaterialManager.GetNames()

GetProperties [material manager]

Get material properties. Differs from GetData in that only material properties available to the user are returned by GetProperties.

UI Access	NA		
Parameters	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.
Return Value	Array of material data		

Python Syntax	GetProperties (<materialname>)
Python Example	<pre>oMaterialManager.GetProperties("Gold", True) oMaterialManager.GetProperties("vacuum")</pre>

VB Syntax	GetProperties <material_name>
VB Example	<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" )
```

Python Syntax	IsUsed(<ComboName>)
Python Example	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()
```

Python Syntax	RemoveUnused()
Python Example	thereWereExtras = oMaterialManager.RemoveUnused()

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"
```

Python Syntax	Export (["NAME:<LibraryName>", <ComboName>, <ComboName>...], <LibraryLocation>)
Python Example	<code>oModelManager.Export (["NAME:mylib", "model1", "model2"])</code>

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.

Python Syntax	GetData(<ComboName>)
Python Example	<code>dataArray = oModelManager.GetData</code>

	("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()
```

Python Syntax	GetNames()
Python Example	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")
```

Python Syntax	IsUsed(<ComboName>)
Python Example	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	<pre>oModelManager.Export (["NAME:mylib", "model1", "model2"])</pre>

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"
"RelativeErrorToleranc:=", 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. <pre>Array(<SolnSelector>, <SolnSelector>, ...)</pre> 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 syntax, but ignored if DoRenorm is false.
<DataType>	String	Optional. Type: "S", "Y", or "Z". The matrix to export.
<pass>	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.
<ComplexFormat>	Integer	Optional. Type: "0", "1", or "2" The format to use for the exported data. 0 = Magnitude/Phase. 1 = Real/Imaginary. 2 = db/Phase.

	<i><Precision></i>	Integer	Optional. Touchstone number of digits precision. Default if not specified is 15.
	<i><UseExportFreqs></i>	Boolean	Specifies whether to use export frequencies.
	<i><IncludeGammaComments></i>	Boolean	Specifies whether to include Gamma and Impedance comments.
	<i><SupportNonStdExport></i>	Boolean	Specifies whether to support non-standard Touchstone extensions for mixed reference impedance.
Return Value	None.		

Python Syntax	ExportNetworkData(<DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>, <FreqsArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <SupportNonStdExport>])
Python Example	oModule.ExportNetworkData("", ["Setup1:Sweep1"], 3, "C://Documents/package_HFSSDesign1.s4p", ['all'], True, 50, "S", -1, 0, 15, True, True, True

VB Syntax	ExportNetworkData <DesignVariationKey>, <SolnSelectionArray>, <SolnSelector>, <FileFormat>, <OutFile>, <FreqsArray>, <DoRenorm>, <RenormImped>, [Optional <DataType>], [Optional <pass>], [Optional <ComplexFormat>], [Optional <Precision>], [Optional <UseExportFreqs>], [Optional <IncludeGammaComments>], [Optional <SupportNonStdExport>]
VB Example	oModule.ExportNetworkData "width='2in'", _ Array("Setup1:Sweep1"), 2, "c:\mydir\out.tab", _ Array("all"), false, 0 oModule.ExportNetworkData "width='2in'", _

	<pre>Array("Setup1:Sweep1", "Setup1:Sweep2"), 3, _ "c:\mydir\out.s2p", Array(1.0e9, 1.5e9, 2.0e9), _ true, 50.0</pre>

ExportNMFData

Exports s-parameters in neutral file format (*.nmf).

UI Access	Results > Solution Data. On the Matrix tab, click Export tab. Then click Export > S-parameter . Select Neutral Model format and export.		
Parameters	Name	Type	Description
	<sourceName>	Array	String names of solutions. Use more than one array entry for a combined Interpolating sweep.
	<outputFileName>	String	Full path of output file, including *.nmf extension.
	<reduceMat>	String	Either "Original" or name of reduced matrix.
	<refImp>	Double	Reference impedance.
	<freqs>	Array	Either "All" or array of doubles representing frequencies.
	<variationKey>	String	Design variation key. Pass empty string for the current nominal variation.
	<displayFormat>	String	"MagPhase", "Reallmag", "DbPhase".
	<pass>	Integer	Optional. The pass to export. This is ignored if the <i>sourceName</i> is a frequency sweep. Leaving out this value or specifying -1 gets all passes.
	<ComplexFormat>	Integer	Optional. The format to use for the exported data. Options are 0 = Magnitude/Phase 1= Real/Imaginary. 2= db/Phase.

	<i><Precision></i>	Integer	Optional. Number of digits precision. If not specified, the default is 6.
Return Value	None.		

Python Syntax	ExportNMFDData (<sourceName>, <outputFileName>,<reduceMat>, <refImp>, <freqs>,<variationKey>, <displayFormat>, <pass>, <ComplexFormat>, <Precision>)		
Python Example	<pre>oDesign.ExportNMFDData("Setup2 : LastAdaptive", "C:/outputs/data.nmf", "Original", 50, [10000000, 20000000, 30000000, 40000000, 50000000, 60000000, 70000000, 80000000, 90000000, 100000000], "", "MagPhase", 0, 1, 11)</pre>		

VB Syntax	ExportNMFDData(<sourceName>, <outputFileName>,<reduceMat>, <refImp>, <freqs>,<variationKey>, <displayFormat>, <pass>, <ComplexFormat>, <Precision>)		
VB Example	<pre>dim solns[1] solns[0] = "Setup1:Sweep1" solns[1] = "Setup1:Sweep2" dim freqs[5] freqs[0] = 10000000 freqs[1] = 20000000 freqs[2] = 30000000 freqs[3] = 40000000 freqs[4] = 50000000 freqs[5] = 60000000 oDesign.ExportNMFDData(solns, "C:/outputs/data.nmf", "Original", 3.2133, freqs, "", "MagPhase", 2, 1, 11)</pre>		

Padstack Manager Script Commands

The padstack manager provides access to padstacks in a project. The manager object is accessed via the definition manager.

```
Set oDefinitionManager = oProject.GetDefinitionManager()
```

```
Set oPadstackManager = oDefinitionManager.GetManager("Padstack")
```

The padstack manager script commands are listed below.

[Add](#)

[Edit](#)

[EditWithComps](#)

[Export](#)

[GetData](#)

[GetNames](#)

[IsUsed](#)

[Remove](#)

[RemoveUnused](#)

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....>,
"hle:=", <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>)
```

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>, // thermal 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

"Bl" // 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:lgn", "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....),
"hle:=", <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>)

```

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

"thm:=", <PadInfo>, // thermal 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.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....>),
 "hle:=", <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>, // thermal 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.

UI Access	N/A		
Parameters	Name	Type	Description
	<AddScriptArray>	Array	Structured array. Array("NAME:<string script name>", "ScriptLang:=", <string "vbscript" or "javascript"> "ScriptText:=", <string text of script>)
Return Value	None.		

Python Syntax	AddScript(<AddScriptArray>)
Python Example	<pre>oDefinitionManager.AddScript(["NAME:MyScript", "ScriptLang:=", "vbscript", "ScriptText:=", "MsgBox(\"HelloWorld\") "])</pre>

VB Syntax	AddScript <AddScriptArray>
VB Example	<pre>oDefinitionManager.AddScript Array("NAME:MyScript", "ScriptLang:=", "vbscript", "ScriptText:=", "MsgBox(""HelloWorld"")")</pre>

EditScript

Edits a script in the definition manager.

UI Access	N/A		
Parameters	Name	Type	Description
	<OriginalName>	String	Name of the script to be edited.
	<EditScriptArray>	Array	Structured array. <pre>Array("NAME:<string script name>", "ScriptLang:=", <string "vbscript" or "javascript"> "ScriptText:=, <string text of script>)</pre>
Return Value	None.		

Python Syntax	EditScript(<OriginalName>, <EditScriptArray>)
Python Example	<pre>oDefinitionManager.EditScript("MyScript", ["NAME:MyNewScript", "ScriptLang:=", "vbscript", "ScriptText:=", "MsgBox(\"HelloAgain\") "])</pre>

VB Syntax	EditScript <OriginalName> <EditScriptArray>
VB Example	<pre>oDefinitionManager.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"
```

Python Syntax	ExportScript(<ExportData>,<Library location>)
Python Example	<pre>oProject.ExportComponent (["NAME:mylib", "myscript"], "PersonalLib")</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>",<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>

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"
```

Python Syntax	RemoveScript (<ScriptName>,<IsProjectScript>, <LibraryName>,<LibraryLocation>)
Python Example	<pre>oDefinitionManager.RemoveScript("myscript", true, "Local", "Project")</pre>

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.

Technology Manager Script Commands

The technology manager provides access to technologies in a project. The manager object is accessed via the definition manager.

```
Set oDefinitionManager = oProject.GetDefinitionManager()  
Set oTechnologyManager = oDefinitionManager.GetManager("Technology")
```

The technology manager script commands are listed below.

[Add](#)

[GetData](#)

[GetNames](#)

[IsUsed](#)

[Remove](#)

Add (Technology Manager)

Adds a technology definition to the library.

UI Access	From the Project Manager window, right-click Definitions > Technology and select Add Definition .		
Parameters	Name	Type	Description
	<propertyArgs>	Array	An array that contains these parameters in order: • <Name> • <ModifiedOn> • <Library> • <VLCTech> • <GFD> • <LayerFilter>

			• <IsLocal>
	<NAME>	String	Name and extension of the technology file.
	<ModifiedOn>	Integer	<i>Optional</i> . An integer corresponding to the number of seconds that elapsed since 00:00 hours, Jan 1, 1970 UTC from the system clock.
	<Library>	String	<i>Optional</i> . Identifies a unique definition library.
	<VLCTech>	String	Directory path to a technology file.
	<GFD>	String	<i>Optional</i> . Directory path to the Green's Function Data file.
	<LayerFilter>	String	<i>Optional</i> . Directory path to the layer map file.
	<IsLocal>	Boolean	<i>Optional</i> . True – technology file is saved/stored locally False – technology file is saved/stored on an external drive
Return Value	None.		

Python Syntax	Add([<NAME>:, <ModifiedOn>, <Library>, <VLCTech>, <LayerFilter>, <IsLocal>])
Python Example	<pre>oTechnologyManager.Add(["NAME:Model.vlc.tech", "ModifiedOn:" , 1694708380, "Library:" , "", "VLCTech:" , "C:/raptorx/gds/gpdsimport/Model.vlc.tech", "LayerFilter:" , "", "IsLocal:" , False])</pre>

VB Syntax	Add Array(<NAME:>, <ModifiedOn>, <Library>, <VLCTech>, <LayerFilter>, <IsLocal>)
VB Example	<pre>oTechnologyManager.Add Array("NAME:Model.vlc.tech", "ModifiedOn:", 1694708380, "Library:", "", "VLCTech:", "C:/raptorx/gds/gpdsimport/Model.vlc.tech", "LayerFilter:", "", "IsLocal:", False)</pre>

GetData (Technology Manager)

Shows data pertaining to a specified technology definition in the active design's **Technology Library**.

UI Access	None.		
Parameters	Name	Type	Description
	<DefName>	String	Name of technology in the Technology Library.
Return Value	None		

Python Syntax	GetData(<DefName>)
Python Example	<pre>oTechnologyManager.GetData("Model.vlc.tech")</pre>

VB Syntax	GetData<DefName>

VB Example	<code>oTechnologyManager.GetData "Model.vlc.tech"</code>

GetNames (Technology Manager)

Shows the complete list of technologies in the active design's **Technology Library**.

UI Access	From the Project Manager window, right-click Technology and select Edit Definition .
Parameters	None.
Return Value	Array of strings containing the names of identified technology files.

Python Syntax	<code>GetNames()</code>
Python Example	<code>oTechnologyManager.GetNames()</code>

VB Syntax	<code>GetNames</code>
VB Example	<code>oTechnologyManager.GetNames</code>

IsUsed (Technology Manager)

Confirms whether a specified technology is applied to the active design.

UI Access	None.		
Parameters	Name	Type	Description
	<TechName>	String	Name of technology in the active design's Technology Library.
Return Value	True – the specified technology is applied to the active design False – the specified technology is not applied to the active design		

Python Syntax	IsUsed(<TechName>)
Python Example	<code>oTechnologyManager.IsUsed("Model.vlc.tech")</code>

VB Syntax	IsUsed<TechName>
VB Example	<code>oTechnologyManager.IsUsed "Model.vlc.tech"</code>

Remove (Technology Manager)

Removes a technology definition from the library.

UI Access	From the Project Manager window, right-click Definitions > Technology and select Edit Definition . to open the Edit Libraries window. Then select a technology and click Remove Technology .		
Parameters	Name	Type	Description
	<Name>	String	Name and extension of the technology file.
	<ProjectDefinition>	Boolean	True – removes (i.e., deletes) the technology file False – does not remove the technology file
	<LibraryName>	String	Identifies a unique definition library.
	<LibraryLocation>	String	Directory path to a technology file.
Return Value	None.		

Python Syntax	<code>Remove(<Name>, <ProjectDefinition>, <LibraryName>, <LibraryLocation>)</code>

Python Example	<code>oTechnologyManager.Remove("Model.vlc.tech", True, "", "Project")</code>

VB Syntax	<code>Remove<Name>, <ProjectDefinition>, <LibraryName>, <LibraryLocation></code>
VB Example	<code>oTechnologyManager.Remove "Model.vlc.tech", True, "", "Project"</code>

This page intentionally
left blank.

27 - 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:

[Activate](#)

[AddCircuitRefPort](#)

[AddLayer](#)

[AddPortsToAllNets](#)

[AddPortsToNet](#)

[AddRefPort](#)

[AddStackupLayer](#)

[AlignObjects](#)

[AlignPorts](#)

[AssignCircuitRefPort](#)

[AssignRefPort](#)

[ChangeLayers](#)

[ChangeOptions](#)

[ChangeProperty](#)

[ClearLayerMappings](#)

[ClearRefPort](#)

[ClearRelative](#)

[Connect](#)

[ConvertPlanetoTrace](#)

[ConvertTracetoPlane](#)

[Copy](#)

[Create3DStructure](#)

[CreateCS](#)

[CreateCircle](#)

[CreateCircleVoid](#)

[CreateCircuitPort](#)

[CreateCircuitPortsOnComponents](#)

[CreateComponent](#)

[CreateEdgePort](#)

[CreateGroundPortComponent](#)

[CreateGroupSelected](#)

[CreateHole](#)

[CreateInterfaceGround](#)

[CreateInterfacePort](#)

[CreateInterfacePortComponent](#)

[CreateLine](#)

[CreateLineFromPolygon](#)

[CreateLineVoid](#)

[CreateMeasure](#)

[CreateNPort](#)

[CreateNetClass](#)

[CreateObjectFromPolygon](#)

[CreatePin](#)

[CreatePinGroup](#)

[CreatePinGroupPort](#)

[CreatePinGroupPortsOnComponents](#)

[CreatePolygon](#)

[CreatePolygonVoid](#)

[CreatePortInstancePorts](#)

[CreatePortsOnComponents](#)

[CreateRectangle](#)

[CreateRectangleVoid](#)

[Create Text](#)

[CreateTrace](#)

[CreateVia](#)

[CutOutSubDesign](#)

[DeactivateOpen](#)

[DeactivateShort](#)

[DefeatureObjects](#)

[DelNetClass](#)

[Delete](#)

[DeleteNets](#)

[DeletePinGroup](#)

[Disconnect](#)

[Duplicate](#)

[DuplicateAcrossLys](#)

[Edit](#)

[Edit3DComponentDefinition](#)

[EraseMeasurements](#)

[Expand](#)

[ExpandWithPorts](#)

[ExportDXF](#)

[ExportGDSII](#)

[ExportGerber](#)

[ExportImage](#)

[ExportNCDrill](#)

[ExportToHFSS](#)

[ExportToQ3D](#)

[FilterObjectList](#)

[FindObjectsByPoint](#)

[FindObjectsByPolygon](#)

[FlipHorizontal](#)

[FlipVertical](#)

[GeometryCheckAndAutofix](#)

[GetAllLayerNames](#)

[GetBBox](#)

[GetCSObjects](#)

[GetCompInstanceFromRefDes](#)

[GetComponentInfo](#)

[GetComponentPinInfo](#)

[GetComponentPins](#)

[GetEditorName](#)

[GetLayerInfo](#)

[GetMaterialList](#)

[GetNetClassNets](#)

[GetNetClasses](#)

[GetNetConnections](#)

[GetPolygon](#)

[GetPolygonDef](#)

[GetPortInfo](#)

[GetProperties](#)

[GetPropertyValue](#)

[GetSelections](#)

[Group](#)

[Heal](#)

[HighlightNet](#)

[ImportDXF](#)

[ImportGDSII](#)

[ModifyNetClass](#)

[Move](#)

[PageSetup](#)

[Paste](#)

[Point](#)

[Polygon](#)

[PositionRelative](#)

[PushExcitations](#)

[RemoveLayer](#)

[RemovePortsFromAllNets](#)

[RemovePortsFromNet](#)

[RemovePortsOnComponents](#)

[Rotate](#)

[SelectAll](#)

[SetCS](#)

[SetLayerMapping](#)

[SetNetVisible](#)

[StitchLines](#)

[Subtract](#)

[ToggleViaPin](#)

[Ungroup](#)

[Unite](#)

[UnselectAll](#)

[ZoomToFit](#)

Also refer to the following subtopics:

[Symbol Editor Scripts](#)

[Footprint Editor Scripts](#)

Activate (Layout Editor)

Use: Causes activation of one or more components.

Syntax: Activate Array("NAME:components",
<object_name>, // 1st component name
<object_name>, // 2nd component, if any
...) // etc

VB Example:

```
oEditor.Activate Array("NAME:components", "2", "3")
```

AddCircuitRefPort (Layout Editor)

Use: Assign an edge as a reference to an existing port; model as a circuit port.

Command: AddCircuitRefPort

Syntax: AddCircuitRefPort

```
Array(<"port name">, <"port name">, ...),  
Array("NAME:Contents",  
"edge:=", Array(<edge description>), "edge:=", Array(<edge description>), ...)
```

Return Value: None.

Parameters: <edge description> for primitive edges

"et:=", "pe", "prim:=", <"prim">, "edge:=", <edge#>

<"prim">:

text that is the primitive name

<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)

```
VB Example: oEditor.AddCircuitRefPort Array("Port3"), Array("NAME:Contents", "edge:=", _
Array("et:=", "pe", "prim:=", "rect_2", "edge:=", 1))
```

AddLayer (Layout Editor)

Adds a layer in a layout definition. This command can take all parameters available in [ChangeLayers \(Layout Editor\)](#) but cannot be recorded.

Note As with other Layout scripting interface commands that modify the layout, this command is not intended for use within scripts that define footprints. The command behavior from within such a script is undefined and may be unexpected. Use the LayoutHost scripting interface commands within scripts that define footprints.

UI Access	None		
Parameters	Name	Type	Description
	<LayerArray>	Array	An array that contains these parameters in order: • <Layername> • <Type> • <TopBottom> • <Color> • <Transparency>

		• <Pattern> • <VisFlag> • <Locked> • <Draw>
<LayerName>	String	Name of layer.
<Type>	String	Type of layer, such as "user", "soldermask", "solderpaste", "silkscreen", "assembly", "wirebond", "glue", "user", "Postprocessing", or "outline".
<TopBottom>	String	Optional. Indicates whether the layer is the top or bottom. Acceptable values are "top", "bottom", or "neither".
<Color>	Integer	Optional. A code to indicate the color of the layer. It is an RGB code in the hex format <i>bbggrr</i> . For example red (#0000ff) is 255, green (#00ff00) is 65280, and blue (#ff0000) is 16711680.
<Transparency>	Integer	Optional. A percentage that indicates the transparency of the layer between 0 (opaque) and 100 (completely transparent and not visible).
<Pattern>	Integer	Optional. A code for the pattern of the layer: 0 is hollow, 1 is solid, and the other patterns 3-8 are various hatch patterns.
<VisFlag>	Integer	Sets visibility for all objects on the layer using the following additive code: • 1 makes primitives (rectangles, circles, polygons) visible. • 2 makes lines (also known as paths or traces) visible. • 4 makes pads (from padstacks, aka vias) visible. • 8 makes holes (or vias) visible. • 16 makes components visible. • 32 makes the mesh overlay visible if they are plotted. • 64 makes the mesh for the background material visible. For example, a value of 25 (1+8+16) makes primitives, vias, and components visible. Use 127 to show everything and 0 to hide all.
<Locked>	Boolean	Optional. Whether the layer is locked.
<Draw>	Integer	Optional. Allows drawing on the layer as wireframe, outlines, or solid. Acceptable values are -1 for wireframe, 0 for normal, and 1 for solid.

Return Value	None

Python Syntax	<pre>oEditor.AddLayer(["NAME:layers", "Name:=", "<LayerName>", "Type:=", "<Type>", "Top Bottom:=", "<TopBottom>", "Color:=", <Color>, "Transparency:=", <Transparency>, "Pattern:=", <Pattern>, " VisFlag:=", <VisFlag>, "Locked:=", <Locked>, "DrawOverride:=", <Draw>,])</pre>
Python Example	<pre>oEditor = oDesign.SetActiveEditor("Layout") oEditor.AddLayer(["NAME:layers",</pre>

	<pre> "Name:=", "signal", "Type:=", "assembly", "Top Bottom:=", "neither", "Color:=", 16711680, "Transparency:=", 0, "Pattern:=", 1, "VisFlag:=", 127, "Locked:=", False, "DrawOverride:=", 0,]) </pre>

VB Syntax	<pre> oEditor.AddLayer Array("NAME:stackup layer", "Name:=", "<LayerName>", "Type:=", "<Type>", "Top Bottom:=", "<TopBottom>", "Color:=", <Color>, "Transparency:=", <Transparency>, "Pattern:=", <Pattern>, "VisFlag:=", <VisFlag>, "Locked:=", <Locked>, "DrawOverride:=", <Draw>,) </pre>
VB Example	<pre> Set oEditor = oDesign.SetActiveEditor("Layout") oEditor.AddLayer Array("NAME:stackup layer", "Name:=", _ "NewLayer", "Type:=", "assembly", "Top Bottom:=", "neither", "Color:=", _ 3361318, "Transparency:=", 60, "Pattern:=", 1, "VisFlag:=", 127, "Locked:=", _ false, "DrawOverride:=", 0,) </pre>

AddPortsToAllNets (Layout Editor)

Use: Adds ports to all the pins in all the nets.

Command: Layout tab under Nets right-click and click **Create Ports**.

Syntax: AddPortsToAllNets

Return Value: None

Parameters: None.

VB Example:

```
oEditor.AddPortsToAllNets
```

AddPortsToNet (Layout Editor)

Use: Add ports to all the pins on the designated nets.

Command: In Layout right-click and click **Port > Create Ports on Net**.

Layout tab under Nets, right-click and click **Create Ports**.

Syntax: AddPortsToNet Array("NAME:Nets", "net-name", ...)

Return Value: None

Parameters: net-name the name of a net; all pins on this particular net receive ports.

VB Example:

```
oEditor.AddPortsToNet Array("NAME:Nets", "CB1", "CB5")
```

AddRefPort (Layout Editor)

Use: Create a reference port from edges and associate with each of the named ports.

Command: (When more than 2 edges are selected.)

Draw > Port > Create

Right-click > Port > Create

Also available through toolbar icon.

Syntax: AddRefPort

```
Array(<"port name">, <"port name">, ...),  
Array("NAME:Contents",  
"edge:=", Array(<edge description>), "edge:=", Array(<edge description>), ...)
```

Return Value: None.

Parameters: <edge description> for primitive edges

"et:=", "pe", "prim:=", <"prim">, "edge:=", <edge#>

<"prim">:

text that is the primitive name

<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)

VB Example: oEditor.AddRefPort Array("Port3"), Array("NAME:Contents", "edge:=",

```
Array("et:=", "pe", "prim:=", "line_998", "edge:=", 0))
oEditor.AddRefPort Array("Port1"), Array("NAME:Contents", "edge:=",
Array("et:=", "pse", "sel:=", "via_5", "layer:=", 10, "sx:=", 0.0015, "sy:=", 0.0015,
"ex:=", -0.0015, "ey:=", 0.0015, "h:=", 0, "rad:=", 0))
```

AlignObjects (Layout Editor)

Use: Align objects, e.g. primitives, relative to each other.

Command: AlignObjects

Syntax: AlignObjects

```
Array("NAME:align", <"alignment">, ...),
```

```
Array("NAME:elements", <"elem">, ...)
```

Return Value: Return Value: None

Parameters: <"alignment">

Type: text

Description: any one of:

"AlignLeft"

"AlignRight"

"CenterHorz" - align centers horizontally

"DistributeCentersHorz" - distribute object centers evenly horizontally

"SpaceEdgesHorz" - space objects equal distances apart horizontally

"AlignTop"

"AlignBottom"

"CenterVert" - align centers vertically

"DistributeCentersVert" - distribute object centers evenly vertically

"SpaceEdgesVert" - space objects equal distances apart vertically

"elem">

Type: text

Description: name of element (primitive, component, etc.) to align

Example:

```
Set oDesign = oProject.SetActiveDesign("HFSS 3D Layout1")
```

```
Set oEditor = oDesign.SetActiveEditor("Layout")
```

```
oEditor.AlignObjects Array("NAME:align", "AlignLeft"), Array("NAME:elements", "circle_2", "rect_1", "poly_3")
```

AlignPorts (Layout Editor)

Use: Causes automatic alignment of the microwave ports of the components, EdgePorts, and Pins referenced in the argument. In case the list is empty, aligns ALL the microwave ports in the layout.

Command: None.

Syntax: **AlignPorts** Array ("NAME: *elements*",

<object_name>, // 1st object

<object_name>, // 2nd object, if any

...)// etc

Return Value: None.

Parameters: The LayoutComp's ID

VB Example: oEditor.Paste Array ("NAME:elements", "1", "2")
oEditor.Paste Array ("NAME:elements")

AssignCircuitRefPort (Layout Editor)

Use: Assign the internal port as a reference to an existing port; model as a circuit port.

Command: AssignCircuitRefPort

Syntax: AssignCircuitRefPort

Array(<"port name">, <"port name">, ...),
Array("NAME:Contents",
"edge=", Array(<edge description>), "edge=", Array(<edge description>), ...)

Return Value: None

Parameters: <"layer">

Type: text

Description: layer name.

<"port">

Type: text

Description: a port or pin name.

<edge description> for primitive edges

```
"et:=", "pe", "prim:=", <"prim">, "edge:=", <edge#>
```

<"prim">

Type: text

Description: primitive name

<edge#>

Type: integer

Description: 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)

VB Example: oEditor.AssignCircuitRefPort Array("pin_1"), "pin_2"

AssignRefPort (Layout Editor)

Use: Assign the named internal port as a reference for each of the named ports.

Command: AssignRefPort

Syntax: AssignRefPort

Array(<"port name">, <"port name">, ...),

Array("NAME:Contents",

"edge:=", Array(<edge description>), "edge:=", Array(<edge description>), ...)

Return Value: None

Parameters: <"layer">

Type: text

Description: layer name.

<"port">

Type: text

Description: a port or pin name.

<edge description>

for primitive edges

"et:=", "pe", "prim:=", <"prim">, "edge:=", <edge#>

<"prim">

Type: text

Description: primitive name

<edge#>

Type: integer

Description: 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)

VB Example: oEditor.AssignRefPort Array("pin_1"), "pin_2"

ChangeProperty

Changes the properties of an object in the history tree.

UI Access	Right-click an object in the History Tree and select Properties .		
Parameters	Name	Type	Description
	<propertyArgs>	Array	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 Automation tab.
Return Value	None.		

Python Syntax	ChangeProperty(<propertyArgs>)
Python Example	Example: Changing the Position of a Box <pre>oEditor.ChangeProperty(["NAME:AllTabs", ["NAME:Geometry3DCmdTab", ["NAME:PropServers" , "Box1:CreateBox:1"],],)</pre>

```

        [
            "NAME:ChangedProps",
            [
                "NAME:Position",
                "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: Change Default Handle (reference coordinate system)  
  
oEditor.ChangeProperty(  
  [  
    "NAME:AllTabs",  
    [  
      "NAME:General",  
      [  
        "NAME:PropServers",  
        "HFSSDesign1_1"  
      ],  
      [  
        "NAME:ChangedProps",  
        [  
          "NAME:Handle",  
          "Value:=" , " NotchOutY"  
        ]  
      ]  
    ]  
  ]  
])
```

VB Syntax	ChangeProperty <propertyArgs>
VB Example	Example: Changing the Position of a Box oEditor.ChangeProperty Array("NAME:AllTabs",

```

        Array("NAME:Geometry3DCmdTab",
            Array("NAME:PropServers" ,
                "Box1:CreateBox:1"
            ),
        ),
        Array("NAME:ChangedProps",
            Array("NAME:Position",
                "X:="      , "0.35in",
                "Y:="      , "0.55in",
                "Z:="      , "0in"
            )
        )
    )
)

```

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: Change Default Handle (reference coordinate system)

```

oEditor.ChangeProperty

```

```
Array("NAME:AllTabs",  
      Array("NAME:General",  
            Array("NAME:PropServers",  
                  "HFSSDesign1_1"  
            ),  
            Array("NAME:ChangedProps",  
                  Array("NAME:Handle",  
                        "Value:="      , " NotchOutY"  
                  )  
            )  
      )  
)
```

ClearLayerMappings (Layout Editor)

Use: Clear layer mappings.

Command: None.

Syntax: ClearLayerMappings <component name>

Return Value: None.

Parameters: <component name> is a string that specifies the name of the component.

VB Example: oEditor.ClearLayerMappings "2"

ClearRefPort (Layout Editor)

Use: Clear references (or referencing) ports from each of the named ports.

Command: Draw > Clear reference Port

Syntax: **ClearRefPort** Array("Port1",...)

Return Value: None.

VB Example: oEditor.ClearRefPort Array("Port1", ...)

ClearRelative (Layout Editor)

Use: Clear the relative positioning between a coordinate system (CS) and another object.

Command: None.

Syntax: oEditor.ClearRelative Array("NAME:elements", <CS1>, <CS2>, ...)

Return Value: None.

VB Example: oEditor.ClearRelative Array("NAME:elements", "CS_13")

Connect (Layout Editor)

Use: Causes connecting of the referenced object. After the command the objects will be in the same electrical net.

Syntax: **Connect** Array ("NAME: elements",

<object_name>, // 1st object

<object_name>, // 2nd object, if any

...) // etc

VB Example:

```
oEditor.Connect Array("NAME:elements", "2:n2", "circle_4")
```

ConvertPlaneToTrace (Layout Editor)

Use: Convert planes to traces

Syntax: ConvertPlaneToTrace <polygon information>

Return Value: None

Parameters: <Polygon_Data>

Type: array

VB Example: oEditor.ConvertPlanetoTrace Array("NAME:elements", "poly_10", "poly_9", "poly_8", ...)

ConvertTraceToPlane (Layout Editor)

Use: Convert traces to planes

Syntax: ConvertTraceToPlane <line information>

Return Value: None

Parameters: <Line_Data>

Type: array

VB Example: oEditor.ConvertTracetoPlane Array("NAME:elements", " line _10", " line _9", " line _8", ...)

Copy (Layout Editor)

Use: Copies the referenced objects to the clipboard.

Syntax: **Copy** Array (<object_name>, // 1st object

```
<object_name>, // 2nd object
```

Return Value: Returns the name of the new net.

VB Example:

```
oEditor.Copy Array("circle_0", "rect_2")
```

CopyToHFSS 3D Layout (Layout Editor)

Use: Generate a single, flat, HFSS 3D Layout design from a selection of components and/or sub-circuits.

Command: CopyToHFSS 3D Layout

Syntax: CopyToHFSS 3D Layout Array("NAME:elements", <"obj1">, <"obj2">, ...)

Return Value: None

Parameters: <"obj1">

Type: text

// component or sub-circuit instance name.

Example:

```
Set oDesign = oProject.SetActiveDesign("Nexxim1")  
Set oEditor = oDesign.SetActiveEditor("Layout")  
oEditor.CopyToHFSS 3D Layout Array("NAME:elements", "1", "2", "3")
```

Create3DStructure (Layout Editor)

Use: Place a set of primitives into a 3D structure coordinate system (i.e. a 3D via or cross-layer plate).

Command: None.

Syntax: oEditor.Create3DStructure <3D structure name>, Array("NAME:elements", <element1>, <element2>, ...)

Return Value: None.

VB Example: oEditor.Create3DStructure "S3D_11", Array("NAME:elements", "circle_10", "circle_9")

CreateCS (Layout Editor)

Use: Inserts a new coordinate system (CS) into the currently active Layout design.

Command: Draw > Coordinate System > Create.

Syntax: There are two forms available:

```
CreateCS
Array("NAME:Contents",
Array("NAME:full_definition", "type:=", "CS", "N:=", <"CS name">),
"placement:=", Array("x:=", <"x_coord">, "y:=", <"y_coord">),
"Clf:=", <color flag>, "Col:=", <color> )
```

```
CreateCS
Array("NAME:Contents",
"Name:=", <"CS name">),
"RelPos:=", Array(edge description))
```

Return Value: Text containing the name of the created relative coordinate system.

Parameters: <"CS name">

Text with the requested name for the relative coordinate system

<x_coord>

Text that is the X location for the relative coordinate system origin

<y_coord>

Text that is the Y location for the relative coordinate system origin

// the following arguments are optional

<color flag>

True if a color is being specified by the next parameter

<color>

Integer specifying the color for the relative coordinate system.

Only used if color flag is true;

<edge description> for primitive edges

"t:=", "pe", "from:=", <"prim">, "pos:=", <edge position>, "et:=", "pe", "prim:=",

`<"prim">, "edge:=", <edge#>`

`<"prim">`: text that is the primitive name

`<edge position>`

double between 0 and 1, inclusive

0 indicates the start of the edge

1 indicates the end of the edge

values in between are positions along the edge

`<edge#>`: integer that is the edge number on the primitive

`<edge description>` for via edges

`"t:=", "pe", "from:=", <"via">, "pos:=", <edge position>, "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)

```
VB Example: oEditor.CreateCS Array("NAME:Contents", Array("NAME:full_definition",  
"type:=", "CS", "N:=", "CS_232"), "placement:=", Array("x:=", "-15mm", "y:=",  
"15mm"), "Clf:=", false, "Col:=", 8421504)
```

```
oEditor.CreateCS Array("NAME:Contents", "Name:=", "CS_15", "RelPos:=",  
Array("t:=", "pe", "from:=", "poly_1", "pos:=", 0, "et:=", "pe", "prim:=", "poly_1",  
"edge:=", 1))
```

```
oEditor.CreateCS Array("NAME:Contents", "Name:=", "CS_196", "RelPos:=",  
Array("t:=", "pe", "from:=", "via_0", "pos:=", 0, "et:=", "pse", "sel:=", "via_0",  
"layer:=", 10, "sx:=", -0.0015, "sy:=", -0.0015, "ex:=", 0.0015, "ey:=", -0.0015,  
"h:=", 0, "rad:=", 0))
```

CreateCircle (Layout Editor)

Use: Creates a circle primitive object and adds it to the current layout. Returns the name of the newly created object.

Syntax: **CreateCircle**<circle_description>

Parameters: **<circle_description>:**

```
Array("NAME:Contents",  
"circleGeometry:=", <circle_geometry>) // object description
```

<circle_geometry> :

```
Array("LayerName:=", <layer_name>, // name of the layer  
"lw:=", <value>, // border line width  
"x:=", <value>, // center x coordinate  
"y:=", <value>, // center y coordinate  
"r:=", <value>) // radius
```

<layer_id> :

integer; never used in scripting

<object_name> :

quoted string, uniquely identifying an object

<value> :

quoted_string_parseable_as_value (e.g. "0.111mm")

VB Example:

```
oEditor.CreateCircle
Array("NAME:Contents",
"circleGeometry:=",
Array("Layer:=", 6,
"Name:=", "circle_150",
"LayerName:=", "Top",
"lw:=", "0mm",
"x:=", "34mm",
"y:=", "-15mm",
"r:=", "8.60232526704263mm"))
```

CreateCircleVoid (Layout Editor)

Use: Creates a circle void and adds it to a particular parent primitive.

Syntax: **CreateCircleVoid**<circle_void_description>

Return Value: Returns the name of the newly created object

Parameters: <circle_void_description>:

```
Array("NAME:Contents",  
"owner:=", <object_name>, // parent primitive name  
"circle voidGeometry:=", <circle_geometry>) // definition
```

VB Example:

```
oEditor.CreateCircleVoid  
Array("NAME:Contents",  
"owner:=", "rect_4",  
"circle voidGeometry:=",  
Array ("Layer:=", 6,  
"Name:=", "circle void_10",  
"LayerName:=", "Top",  
"lw:=", "0mm",  
"x:=", "26mm",  
"y:=", "11mm",  
"r:=", "1.41421356237309mm"))
```

CreateCircuitPort (Layout Editor)

Use: Create a circuit port between two points.

Command: Draw > Create Circuit Ports

Syntax: CreateCircuitPort Array("NAME:Location", "PosLayer:=", "layer name", "X0:=", _
x-value, "Y0:=", y-value, "NegLayer:=", "layer name", "X1:=", x-value, "Y1:=", y-value)

Return Value: None

Parameters: <"layer">

Type: text

Description: layer name.

<"port">

Type: text

Description: a port or pin name.

<edge description>

for primitive edges

"et:=", "pe", "prim:=", <"prim">, "edge:=", <edge#>

<"prim">

Type: text

Description: primitive name

<edge#>

Type: integer

Description: 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)

VB Example: oEditor.CreateCircuitPort Array("NAME:Location", "PosLayer:=", "Top", "X0:=", _
-0.003, "Y0:=", 0.003, "NegLayer:=", "Top", "X1:=", 0.002, "Y1:=", 0.003)

CreateCircuitPortsOnComponents (Layout Editor)

Creates circuit ports on a selected component.

UI Access	After selecting a component, right-click within the Layout Editor and select Port > Create Ports on Component .		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array containing these parameters in order: • <NAME:Components> • <ComponentName>
	<NAME:Components>	String	Required first parameter which must have the specified value (i.e., "Components").
	<ComponentName>	String	Names of the target components, separated by commas.
	<Parameters>	Array	Structured array containing these parameters in order:

			• <NAME:Nets> • <NetNames>
	<NAME:Nets>	String	Required first parameter which must have the specified value (i.e., "Nets")
	<NetNames>	String	Names of the nets to place ports on, separated by commas.
	<ReferenceNet>	String	Name of the net to use as a reference.
Return Value	None.		

Python Syntax	CreateCircuitPortsOnComponents ([<NAME:Components>,<ComponentName>], [<NAME:Nets>,<NetNames>], <ReferenceNet>)
Python Example	<pre>oEditor.CreateCircuitPortsOnComponents (["NAME:Components", U0], ["NAME:Nets", "net3", "net4"], "net5"])</pre>

VB Syntax	CreateCircuitPortsOnComponents Array(<NAME:Components>,<ComponentName>), Array(<NAME:Nets>,<NetNames>), <ReferenceNet>
VB Example	<code>oEditor.CreateCircuitPortsOnComponents Array("NAME:Components", U0), Array("NAME:Nets", "net3", "net4"), "net5"</code>

CreateComponent (Layout Editor)

Use: Places a component.

Syntax: CreateComponent <Array>

Return Value: Returns the name of the newly created component.

Parameters: Array("NAME:Contents",
 "definition_name:=", <component name>,
 "placement:=", Array("x:=", <x position>, "y:=", <y position>),
 "layer:=", <placement layer name>,
 "StackupLayers:=", Array("<footprint stackup layer>:<design stackup layer>", ...),
 "DrawLayers:=", Array("<footprint layer>:<design layer>", ...))

VB Example: `oEditor.CreateComponent Array("NAME:Contents",
 "definition_name:=", "MSTRL",
 "placement:=", Array("x:=", "-13mm", "y:=", "5mm"),
 "layer:=", "Top",
 "StackupLayers:=", Array("Top:Top", "0:Dielectric", "0:Ground"),
 "DrawLayers:=", Array("Measures:Measures", "Assembly Top:Assembly Top", "Silkscreen Top:Silkscreen Top", "Wirebonds:0"))`

Notes:

Each Layer mapping is specified as a single text string in quotes, "<footprint layer>:<design layer>". If the layer does not map to anything, use a "0" for the layer. For example, you can use "Measures:0", if the footprint "Measures" layer does not map to a design layer. Use "0:Dielectric", if the design "Dielectric" layer does not map to a footprint layer.

Note, also, that the "StackupLayers" and the "DrawLayers" arguments may be omitted, in which case, the mappings are determined automatically.

CreateEdgePort (Layout Editor)

Creates an edge port using the specified edges.

UI Access	Draw > Port > Create or Right-click > Port > Create.		
Parameters	Name	Type	Description
	<EdgePortArray>	Array	Identifies the contents of the array (i.e., "Contents") containing the following arguments: <NAME:Contents> <Edge> <External>
	<NAME:Contents>	String	Identifies the contents of the array (i.e., "Contents").
	<Edge>	Array	Array containing separate arrays for primitive edges and via edges. For primitive edges, array contains: <Et> <Prim> <Edge>

		For via edges, array contains: <Et> <Sel> <Layers> <Sx> <Sy> <Ex> <Ey> <H> <Rad> Note: Array can repeat in the same command.
<Et>	String	Type of new edge port (i.e., "pe" for primitive edge or "pse" for pad-stack edge or via).
<Prim>	String	Name of the target primitive.
<Edge#>	Integer	Edge number of the target primitive.
<Sel>	String	Name of the target via.
<Layers>	Integer	Layer ID of the pad of the target via.
<Sx>	Integer	Starting X coordinate for the arc of the new edge port.
<Sy>	Integer	Starting Y coordinate for the arc of the new edge port.
<Ex>	Integer	Ending X coordinate for the arc of the new edge port.
<Ey>	Integer	Ending Y coordinate for the arc of the new edge port.
<H>	Integer	Arc height of the new edge port (e.g., enter 0 for a straight edge).
<rad>	String	Arc size of the new edge port (e.g., enter 0 for a straight edge).

	<table><tr><td><External></td><td>Boolean</td><td>True — the port is external. False — the port is internal.</td></tr></table>	<External>	Boolean	True — the port is external. False — the port is internal.
<External>	Boolean	True — the port is external. False — the port is internal.		
Return Value	Text containing the name of the created edge port. (Returns an empty name if the edge port is not created.)			

Python Syntax	CreateEdgePort(<EdgePortArray>)
Python Example	Primitive edge port: <pre>oEditor.CreateEdgePort ("NAME:Contents", "edge:=", ("et:=", "pe", "prim:=", "rect_167", "edge:=", 0), "edge:=", ("et:=", "pe", "prim:=", "rect_167", "edge:=", 3), "external:=", true))</pre> Via edge port: <pre>oEditor.CreateEdgePort ("NAME:Contents", "edge:=", ("et:=", "pse", "sel:=", "via_0", "layer:=", 10, "sx:=", -0.0015, "sy:=", -0.0015, "ex:=", 0.0015, "ey:=", -0.0015, "h:=", 0, "rad:=", 0), "external:=", true))</pre>

VB Syntax	CreateEdgePort<EdgePortArray>
VB Example	Primitive edge port: <pre>oEditor.CreateEdgePort Array("NAME:Contents", "edge:=", Array("et:=", "pe", "prim:=", "rect_167", "edge:=", 0), "edge:=", Array("et:=", "pe", "prim:=",</pre>

	<pre>"rect_167", "edge:=", 3), "external:=", true) Via edge port: oEditor.CreateEdgePort Array("NAME:Contents", "edge:=", Array("et:=", "pse", "sel:=", "via_0", "layer:=", 10, "sx:=", -0.0015, "sy:=", -0.0015, "ex:=", 0.0015, "ey:=", -0.0015, "h:=", 0, "rad:=", 0), "external:=", true)</pre>

CreateGroundPortComponent (Layout Editor)

Use: Create a ground port component

Command: Right-click selected port > Port > Nexxim Ports > Add Grounds at Unconnected Pins

Return Value: None

Parameters: <Component_Data>

Type: array

VB Example: oEditor.CreateGroundPortComponent Array("NAME:elements", "Comp37", "Comp10", ...)

CreateGroupSelected (Layout Editor)

Use: Groups a set of primitives into a subdesign.

Command: Layout > Group into Subdesign

Syntax: oEditor.CreateGroupSelected Array("NAME:elements", <selectable name>, ...)

Return Value: None.

VB Example: oEditor.CreateGroupSelected Array("NAME:elements", "1", "2", "3", "4", "5", "6")

CreateHole (Layout Editor)

Use: Places a Hole object.

Syntax: **CreateHole** <Array>

Return Value: Returns the name of the newly created Hole

```
Parameters: Array("NAME:Contents",
Array("NAME:full_definition",
"type:=", "hole",
Array("NAME:Properties",
"VariableProp:=",
Array("radius", "D", "", <value>), // radius
"VariableProp:=",
Array("sides", "D", "", integer)), //side count
"from_layer:=", <layer_name>,
"to_layer:=", <layer_name>),
"placement:=", <vpoint>, // placement position
"layer:=", <layer_name>) // placement layer
```

<vpoint> :

```
Array("x:=", <value>, // X coordinate
"y:=", <value>) // Y coordinate
```

```
VB Example: oEditor.CreateHole
Array("NAME:Contents",
Array("NAME:full_definition",
"type:=", "hole",
Array("NAME:Properties",
"VariableProp:=",
Array("radius", "D", "", "0.635mm"),
"VariableProp:=",
Array("sides", "D", "", "6")),
"from_layer:=", 6,
"to_layer:=", 6),
"placement:=", Array("x:=", "-36mm", "y:=", "-9mm"),
"layer:=", "Top")
```

CreateInterfaceGround (Layout Editor)

Use: Create an interface ground

Command: Right-click selected port > Port > Nexxim Ports > Circuit Ground

Return Value: None

Parameters: <Port_Data>

Type: array

VB Example: oEditor.CreateInterfaceGround Array("NAME:NexximPort", "Name:=", "NexximGnd3", _
"X:=", -0.00103067385498434, "Y:=", -0.00738649582490325, "Rot:=", 0)

CreateInterfacePort (Layout Editor)

Use: Create an interface port

Command: Right-click selected port > Port > Nexxim Ports > Circuit Interface

Syntax: CreateInterfacePort <Port_Data>

Parameters: <Port_Data>

Type: array

VB Example: oEditor.CreateInterfacePort Array("NAME:NexximPort", "Name:=", "Port1", _
"X:=", -0.00309202168136835, "Y:=", 0.00314928125590086, "Rot:=", 0)

CreateInterfacePortComponent (Layout Editor)

Use: Create an interface port component

Command: Right-click selected port > Port > Nexxim Ports > Add Interface Ports at Unconnected Pins

Syntax: CreateInterfacePortComponent <Port_Data>

Parameters: <Component_Data>

Type: array

VB Example: oEditor.CreateInterfacePortComponent Array("NAME:elements", "Comp37", "Comp10", ...)

CreateLine (Layout Editor)

Use: Creates a line primitive object.

Syntax: **CreateLine** <line_description>

Return Value: Returns the name of the newly created object.

Parameters: <line_description>:

```
    Array("NAME:Contents",  
        "lineGeometry:=", <line_geometry>)
```

<line_geometry> :

```
    Array("LayerName:=", <layer_name>,  
        "lw:=", <value>, // line width  
        "endstyle:=", <end_style>,  
        "joinstyle:=", <join_style>,  
        <vertex_sequence> )
```

<end_style> : integer

```
    FlatEnd = 0  
    ExtendedEnd = 1  
    RoundEnd = 2
```

<join_style> : integer

```
    UnmiteredJoin = 0  
    MiteredJoin = 1  
    OptimallyMiteredJoin = 2
```

```
RadiallyMiteredJoin = 3
```

```
ChordMiteredJoin = 4
```

<vertex_sequence>:

```
"n:=", integer, // the total count of the vertices
```

```
"x0:=", <value>, "y0:=", <value>,
```

```
"x1:=", <value>, "y1:=", <value>,
```

```
etc. up to vertex count)
```

VB Example:

```
oEditor.CreateLine
```

```
Array("NAME:Contents",
```

```
"lineGeometry:=",
```

```
Array("Layer:=", 6,
```

```
"Name:=", "line_1",
```

```
"LayerName:=", "Top",
```

```
"lw:=", "2mil",
```

```
"endstyle:=", 0,
```

```
"joinstyle:=", 0,
```

```
"n:=", 4,
```

```
"x0:=", "-32mm", "y0:=", "2mm",
```

```
"x1:=", "-5mm", "y1:=", "12mm",  
"x2:=", "1mm", "y2:=", "-4mm",  
"x3:=", "19mm", "y3:=", "3mm"))
```

CreateLineFromPolygon (Layout Editor)

Use: Create a line/trace object on the specified layer and net from the provided Polygon object. Net is optional.

Syntax: oLayout.CreateLineFromPolygon(<oPolygon>, <width>, <bendType>, <startCapType>, <endCapType>, <Layer>, <Net>)

Parameters: width - line width, in meters

bendType - {'corner', 'round'}

startCapType, endCapType - {'flat', 'extended', 'round'}

VB Example:

```
newobj = oLayout.CreateLineFromPolygon(oPolygon, 0.1e-3, 'round', 'flat', 'round', 'L1', 'DQ0')
```

CreateLineVoid (Layout Editor)

Use: Creates a line void and adds it to a specified as parameter parent primitive.

Syntax: **CreateLineVoid**<line_void_description>

Return Value: Returns the name of the newly created object.

Parameters: **<line_void_description>:**

```
Array("NAME:Contents",  
"owner:=", <object_name>, // parent primitive name  
"line voidGeometry:=", <line_geometry>) // definition
```

```
VB Example: oEditor.CreateLineVoid
Array("NAME:Contents",
"owner:=", "rect_4",
"line voidGeometry:=",
Array("Layer:=", 6,
"Name:=", "line void_14",
"LayerName:=", "Top",
"lw:=", "2mil",
"endstyle:=", 0,
"joinstyle:=", 0,
"n:=", 3,
"x0:=", "27mm", "y0:=", "5mm",
"x1:=", "35mm", "y1:=", "5mm",
"x2:=", "36mm", "y2:=", "9mm"))
```

CreateMeasure (Layout Editor)

Use: Creates a measurement.

Syntax: CreateMeasure

Use: Returns the name of the created object.

Parameters: Array("NAME:Contents",

```
"MeasurementGeometry:=",  
Array("LayerName:=", <layer_name>, // layer  
"lw:=", <value>, // line width  
"sx:=", <value>, // start X coordinate  
"sy:=", <value>, // start Y coordinate  
"ex:=", <value>, // end X coordinate  
"ey:=", <value>, // end Y coordinate  
<text_style>)
```

<text_style> :

```
"name:=", <quoted string>, // its name  
"isPlot:=", <bool>,  
"font:=", <font_name>,  
"size:=", double, // size in current units  
"angle:=", <value>,  
"weight:=", <text_weight>,  
"just:=", <text_justification>,  
"mirror:=", <bool>,  
"scales:=", <bool>))  
  
VB Example: oEditor.CreateMeasure  
Array("NAME:Contents",
```



```
"MeasurementGeometry:=",  
Array("Layer:=", 0,  
"Name:=", "Measurement_2",  
"LayerName:=", "Measures",  
"lw:=", "0mm",  
"sx:=", "-32mm",  
"sy:=", "-13mm",  
"ex:=", "32mm",  
"ey:=", "-11mm",  
"name:=", "<DefaultAnnotation>",  
"isPlot:=", false,  
"font:=", "Arial",  
"size:=", 10,  
"angle:=", "0deg",  
"weight:=", 3,  
"just:=", 4,  
"mirror:=", false,  
"scales:=", false))
```

CreateNportCircuitElements

Creates an HFSS Circuit Element from one or more ports.

UI Access	Right-click Circuit Elements > Create Single Port Model... or Circuit Elements > Create Multi-Terminal Model...						
Parameters	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><NPortArray></td><td>Array</td><td>Structured array. Array("NAME:<NPortName>", "Definition:=", <string>, <AssignmentArray>)</td></tr></table>	Name	Type	Description	<NPortArray>	Array	Structured array. Array("NAME:<NPortName>", "Definition:=", <string>, <AssignmentArray>)
Name	Type	Description					
<NPortArray>	Array	Structured array. Array("NAME:<NPortName>", "Definition:=", <string>, <AssignmentArray>)					
Return Value	None.						

Python Syntax	CreateNportCircuitElement (<NPortArray>)
Python Example	<pre>oModule.CreateNportCircuitElement(["NAME:Nport1", "Definition:=", "Model", ["NAME:Assignments", ["NAME:Model", "Assign:=", "1"]]])</pre>

	<pre>oModule.CreateNportCircuitElement(["NAME:Nport2", "Definition:=", "Model2", ["NAME:Assignments", ["NAME:P01__NET179__T1", "Assign:=", "2"], ["NAME:P02__NET178__T1", "Assign:=", "3"], ["NAME:P03__NET178__T1", "Assign:=", "4"], ["NAME:P04__NET179__T1", "Assign:=", "5"],]])</pre>

VB Syntax	CreateNportCircuitElement <NPortArray>
VB Example	<pre>oModule.CreateNportCircuitElement _ Array("NAME:Nport1", _ "Definition:=", "Model", _</pre>

```

Array("NAME:Assignments", Array("NAME:Model", "Assign:=", "1"))))

oModule.CreateNportCircuitElement _
Array("NAME:Nport2", _
"Definition:=", "Model2", _
Array("NAME:Assignments", Array("NAME:P01__NET179__T1", "Assign:=", _
"2"), Array("NAME:P02__NET178__T1", "Assign:=", "3"), _
Array("NAME:P03__NET178__T1", "Assign:=", _
"4"), Array("NAME:P04__NET179__T1", "Assign:=", "5")))

```

CreateNetClass

Use: Create a new net class into a layout.

Command: CreateNetClass.

Syntax: CreateNetClass (<name> <description> <nets name list>)

Return Value: None

VB Example: oEditor.CreateNetClass "power", "power nets", Array("A_D2D_VDD_0", "A_D2D_VDD_1", "A_D2D_VSS_0")

CreateObjectFromPolygon (Layout Editor)

Use: Create a polygon on the specified layer and net from the provided Polygon object. Return the name of the new object. Net is optional.

Syntax: oLayout.CreateObjectFromPolygon(<oPolygon>, <Layer>, <Net>)

Return Value: Object name.

VB Example:

```
newobj = oLayout.CreateObjectFromPolygon(oPolygon, 'L1', 'GND')
```

CreatePin (Layout Editor)

Use: Creates a pin.

Syntax: **CreatePin** <pin_description>

Return Value: Returns the name of the newly created pin.

Parameters: <pin_description>:

```
    Array("NAME:Contents",  
    Array("NAME:Port", "Name:=", <object_name>), // not used  
    "Rotation:=", Array(<value>), // rotation  
    "Offset:=", <vpoint>, // position  
    "Padstack:=", <quoted_string>) // padstack name
```

VB Example: oEditor.CreatePin

```
Array("NAME:Contents",  
Array("NAME:Port", "Name:=", "Pin_1"),  
"Rotation:=", Array("0deg"),  
"Offset:=", Array("x:=", "-2mm", "y:=", "-1mm"),  
"Padstack:=", "NoPad SMT East")
```

CreatePinGroup (Layout Editor)

Use: Create a pin group from a selection of pins.

Syntax: `CreatePinGroup(STRING pin-group-name, ["NAME:elements", STRING pin, STRING pin, ...])`

VB Example: `oEditor.CreatePinGroup("J1_GND_Group", [ "NAME:elements", "J1-1", "J1-2", "J1-35", "J1-56" ])`

CreatePinGroupPort (Layout Editor)

Use: Create a pin group port from two pin-groups; first is positive, second is negative.

Syntax: `CreatePinGroupPort(["Name:elements", STRING pos-pin-group, STRING neg-pin-group])`

VB Example: `oEditor.CreatePinGroupPort( [ "NAME:elements", "J1_GND_Group", "J1_VCC_Group" ] )`

CreatePinGroupPortsOnComponents (Layout Editor)

Creates pin group ports on a selected component.

UI Access	After selecting a component, right-click within the Layout Editor and select Port > Create Ports on Component .		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array containing these parameters in order: • <NAME:Components> • <ComponentName>
	<NAME:Components>	String	Required first parameter which must have the specified value (i.e., "Components").

	<ComponentName>	String	Names of the target components, separated by commas.
	<Parameters>	Array	Structured array containing these parameters in order: • <NAME:Nets> • <NetNames>
	<NAME:Nets>	String	Required first parameter which must have the specified value (i.e., "Nets").
	<NetNames>	String	Names of the nets to place ports on, separated by commas.
	<ReferenceNet>	String	Name of the net to use as a reference.
	<PinGroupPerNet>	Boolean	True creates a different pin group for each net. False creates a single pin group across all nets.
Return Value	None.		

Python Syntax	CreatePinGroupPortsOnComponents([<NAME:Components>,<ComponentName>], [<NAME:Nets>,<NetNames>], <ReferenceNet>, <PinGroupPerNet>)
Python Example	<pre>oEditor.CreatePinGroupPortsOnComponents (["NAME:Components", U0], ["NAME:Nets", "net3", "net4"])</pre>

	<pre>], "net5", False])</pre>

VB Syntax	CreatePinGroupPortsOnComponents Array(<NAME:Components>,<ComponentName>), Array(<NAME:Nets>,<NetNames>), <ReferenceNet>, <PinGroupPerNet>
VB Example	oEditor.CreatePinGroupPortsOnComponents Array("NAME:Components", U0), Array("NAME:Nets", "net3", "net4"), "net5", False

CreatePolygon (Layout Editor)

Use: Creates a polygon primitive object.

Syntax: **CreatePolygon**<polygon_description>

Return Value: Returns the name of the newly created object.

Parameters: <polygon_description>:

```
Array("NAME:Contents",
      "lineGeometry:=", <polygon_geometry>)
```

<polygon_geometry> :

```
Array("LayerName:=", <layer_name>, // layer
      "lw:=", <value>, // border line width
      <vertex_sequence> ) // polygon boundary
```

VB Example:

```
oEditor.CreatePolygon
```



```
Array("NAME:Contents",  
      "polyGeometry:=",  
      Array("Layer:=", 6,  
            "Name:=", "poly_5",  
            "LayerName:=", "Top",  
            "lw:=", "0mm",  
            "n:=", 5,  
            "x0:=", "-35mm", "y0:=", "17mm",  
            "x1:=", "-37mm", "y1:=", "5mm",  
            "x2:=", "-30mm", "y2:=", "8mm",  
            "x3:=", "-30mm", "y3:=", "8mm",  
            "x4:=", "-30mm", "y4:=", "8mm"))
```

CreatePolygonVoid (Layout Editor)

Use: Creates a polygon void and adds it to a specified as parameter parent primitive.

Syntax: **CreatePolygonVoid** <polygon_void_description>

Return Value: Returns the name of the newly created object.

Parameters: <polygon_void_description>:

```
Array("NAME:Contents",
```

```
"owner:=", <object_name>, // owner name
"poly voidGeometry:=", <polygon_geometry>) // definition
VB Example: oEditor.CreatePolygonVoid
Array("NAME:Contents",
"owner:=", "rect_4",
"poly voidGeometry:=",
Array("Layer:=", 6,
"Name:=", "poly void_18",
"LayerName:=", "Top",
"lw:=", "0mm",
"n:=", 5,
"x0:=", "21mm", "y0:=", "9mm",
"x1:=", "21mm", "y1:=", "5mm",
"x2:=", "24mm", "y2:=", "7mm",
"x3:=", "24mm", "y3:=", "7mm",
"x4:=", "24mm", "y4:=", "7mm"))
```

CreatePortInstancePorts (Layout Editor)

Use: Create port on port instances.

Command: Right-Click-Menu > Port > Create

Syntax: CreatePortInstancePorts Array("NAME:elements", "element-name",...)

Return Value: None

Parameters: <element-name>

Type: string

// Name of a port instance on which ports will be created.

VB Example:

```
oEditor.CreatePortInstancePorts Array("NAME:elements", "0:106")
```

CreatePortsOnComponents (Layout Editor)

Use: Create port on port instances of selected components.

Command: Right-Click-Menu > Port > Create Ports on Component

Syntax: CreatePortsOnComponentsArray("NAME:elements", "element-name",...)

Return Value: None

Parameters: <element-name>

Type: string

// Name of a component.

// Ports are created on the unconnected port instances of the selected components.

VB Example:

```
oEditor.CreatePortsOnComponents Array("NAME:elements", "0")
```

CreateRectangle (Layout Editor)

Use: Creates a rectangle primitive object and adds it to the current layout.

Syntax: **CreateRectangle**<rectangle_description>

Return Value: Returns the name of the newly created object.

Syntax: **<rectangle_description>**:

```
Array("NAME:Contents",  
      "rectGeometry:=", <rectangle_geometry>)  
  
  <rectangle_geometry> :  
  Array("LayerName:=", <layer_name>, // placement layer  
        "lw:=", <value>, // line width  
        "x:=", <value>, // center X coordinate  
        "y:=", <value>, // center Y coordinate  
        "w:=", <value>, // width (X-direction size)  
        "h:=", <value>, // height (Y-direction size)  
        "ang:=", <value>)) // rotation
```

VB Example:

```
oEditor.CreateRectangle  
Array("NAME:Contents",  
      "rectGeometry:=",  
      Array("Layer:=", 6,  
            "Name:=", "rect_4",  
            "LayerName:=", "Top",  
            "lw:=", "0mm",  
            "x:=", "29mm",
```

```
"y:=", "9.5mm",  
"w:=", "24mm",  
"h:=", "15mm",  
"ang:=", "0deg"))
```

CreateRectangleVoid (Layout Editor)

Use: Creates a rectangle void and adds it to a specified as parameter parent primitive.

Syntax: **CreateRectangleVoid**<rectangle_void_description>

Return Value: Returns the name of the newly created object.

Syntax: **<rectangle_void_description>**:

```
Array("NAME:Contents",  
      "owner:=", <object_name>, // owner_name  
      "rect voidGeometry:=", <rectangle_geometry>)
```

VB Example:

```
oEditor.CreateRectangleVoid  
Array("NAME:Contents",  
      "owner:=", "rect_4",  
      "rect voidGeometry:=",  
      Array("Layer:=", 6,  
            "Name:=", "rect void_16",
```

```
"LayerName:=", "Top",  
"lw:=", "0mm",  
"x:=", "34.5mm",  
"y:=", "13mm",  
"w:=", "3mm",  
"h:=", "2mm",  
"ang:=", "0deg"))
```

CreateText (Layout Editor)

Use: Creates a text primitive object and adds it to the current layout.

Syntax: **CreateText** <text_description>

Return Value: Returns the name of the newly created object.

Parameters: <text_description>:

```
Array("NAME:Contents",  
      "textGeometry:=", <text_geometry>)
```

<text_geometry> :

```
Array("LayerName:=", <layer_name>,  
      "x:=", <value>, // origin X  
      "y:=", <value>, // origin Y  
      "ang:=", <value>, // rotation  
      "isPlot:=", <bool>, // is it plotted or not
```

```
"font:=", <font_name>,  
"size:=", <value>,  
"weight:=", <text_weight>,  
"just:=", <text_justification>,  
"mirror:=", <bool>,  
"text:=", <quoted_string>) // text itself
```

<bool> : *true* | *false*

<font_name> : quoted string, a font name

<text_weight> : integer

```
Thin = 0,  
ExtraLight = 1  
Light = 2  
Normal = 3  
Medium = 4  
SemiBold = 5  
Bold = 6  
ExtraBold = 7
```

Heavy =8

<text_justification> : integer

LeftTop = 0

LeftBase = 1

LeftBottom = 2

CenterTop = 3

CenterBase= 4

CenterBottom = 5

RightTop = 6

RightBase = 7

RightBottom = 8

VB Example: oEditor.CreateText

Array("NAME:Contents",

"textGeometry:=",

Array("Layer:=", 6,

"Name:=", "text_6",

"LayerName:=", "Top",

"x:=", "-26mm",

"y:=", "-16mm",

"ang:=", "0deg",


```
"isPlot:=", true,  
"font:=", "Roman",  
"size:=", "0.508mm",  
"weight:=", 3,  
"just:=", 4,  
"mirror:=", false,  
"text:=", "Sample"))
```

CreateTrace (Layout Editor)

Use: Create a trace with a manually specified path between pins, ports, or selected edges.

Command: Draw > Route > Manual

Syntax: CreateTrace

```
Array("NAME:options", "Layer1:=", <"layer">, "Layer2:=", <"layer">),  
Array("NAME:Lines", "Num:=", <n_lines>, "l0:=",  
Array("Name:=", <"line">, "LayerName:=", <"layer">, "lw:=", <"width">,  
"endstyle:=", <endstyle>, "joinstyle:=", <joinstyle>, "n:=", <n_vertex>,  
"U:=", <"units">, "x:=", <value>, "y:=", <value>, ), ...),  
Array("NAME:Vias", "Num:=", <n_vias>,  
Array("NAME:v1", "name:=", <"via">, "ReferencedPadstack:=", <"padstack">,  
"vposition:=", Array("x:=", <"vertex">, "y:=", <"vertex">), "vrotation:=",
```

```
Array(<"angle">), "overrides hole:=", <override>, "hole diameter:=",  
Array(<"diameter">), "highest_layer:=", <"layer">, "lowest_layer:=", <"layer">), ...),  
Array("NAME:elements", <"port">, ...),  
Array("NAME:EdgeRefs", "edge:=", Array(<edge description>), ...)
```

Return Value: None

Parameters: <"layer">

Type: text

Description: layer name

<n_lines>

Type: integer

Description: number of line definitions following

<"line">

Type: text

Description: line name

<"width">

Type: text

Description: line width; value with units, e.g. "1mm"

<endstyle>

Type: integer

Description: end (cap) style value for the line.

<joinstyle>

Type: integer

Description: join style value.

<n_vertex>

Type: integer

Description: number of vertices in the line

<value>

Type: double

Description: simple value, e.g. 10

<n_vias>

Type: integer

Description: number of via definitions following

<"via">

Type: text

Description: via name

<"padstack">

Type: text

Description: padstack definition name.

<"angle">

Type: text

Description: via orientation; value with units, e.g. "180deg"

<override>

Type: boolean (true or false)

Description: if true, the diameter is used to override the via hole definition.

<"diameter">

Type: text

Description: via hole diameter override; a value with units, e.g. "1mm"

<"port">

Type: text

Description: a port or pin name.

<edge description>

for primitive edges

"et:=", "pe", "prim:=", <"prim">, "edge:=", <edge#>

<"prim">

Type: text

Description: primitive name

<edge#>

Type: integer

Description: 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)

```
VB Example: oEditor.CreateTrace Array("NAME:options", "Layer1=", "Top",  
"Layer2=", "Ground"), Array("NAME:elements"), Array("NAME:EdgeRefs",  
"edge=", Array("et=", "pe", "prim=", "rect_4", "edge=", 3), "edge=",  
Array("et=", "pse", "sel=", "via_3", "layer=", 8, "sx=", -0.0015, "sy=", 0.0015,  
"ex=", -0.0015, "ey=", -0.0015, "h=", 0, "rad=", 0))
```

CreateVia (Layout Editor)

Use: Creates a new via.

Syntax: **CreateVia** <via_description>

Return Value: Returns the name of the created via

Parameters: <via_description>:

```
Array("NAME:Contents",  
"name:=", <object_name>, // not used  
"vposition:=", <vpoint>, // position  
"rotation:=", double, // rotation in radians  
"overrides hole:=", <bool>, // overrides or not padstack hole  
"hole diameter:=", Array(<value>), // via diameter  
"ReferencedPadstack:=", <quoted_string>, // padstack name  
"highest_layer:=", <layer_name>,  
"lowest_layer:=", <layer_name>)  
  
VB Example: oEditor.CreateVia  
Array("NAME:Contents",  
"name:=", "",  
"vposition:=", Array("x:=", "-22mm", "y:=", "5mm"),  
"rotation:=", 0,
```

```
"overrides hole:=", false,  
"hole diameter:=", Array( "1mm"),  
"ReferencedPadstack:=", "Round 1mm/0.5mm",  
"highest_layer:=", "Top",  
"lowest_layer:=", "Bottom")
```

CutOutSubDesign (Layout Editor)

Use: Cut out a subdesign.

Command: CutOutSubDesign

Syntax: oEditor.CutOutSubDesign Array(

```
    "NAME:Params",  
    "Name:=", <"name">,  
    "EMDesign:=", <boolean>,  
    "SubDesign:=", <boolean>,  
    "Within:=", <boolean>,  
    "Without:=", <boolean>,  
    "AutoGenExtent":=<boolean>,  
    "Expansion":=<double>,  
    "RoundCorner":=<boolean>,  
    "Increments":=<integer>,  
    "ExtentSel:=", Array(<"extent-poly">, ...),
```



```
Array("NAME:Nets", "net:=", Array(<"net-name">, <clip>), ...))
```

In place of "ExtentSel:=", Array(<"extent-poly">, ...) an explicit polygon can be used:

```
"Extent:=", Array("cl:=", true,
    "pt:=", Array(U:=", <"units">, "x:=", <double>, "y:=", <double>, ...))
```

Parameters: Name — name of the cutout design.

EMDesign — if true, create an EM design, otherwise create a Nexxim design.

Within — boolean; cut out the interior region.

Without — boolean; cut out the exterior region.

AutoGenExtent — boolean; when true, Circuit disregards both ExtentSel and Extent. Circuit will instead generate a polygonal shape around all nets which are not clipped.

Expansion — double; similar to the Auto Generate Extent dialog box. However, this is always a unitless fraction. By default it is set to .1.

RoundCorner — boolean; identical to the Auto Generate dialog box. By default this is set to true. In general, rounded corners are preferred when the cutout shape has acute or nea-acute angles

Increments — integer; this is from the Auto Generate Dialog, and by default is set to true. This can greatly increase the running time and a small increase can make a big difference. It is probably best to experiment with this parameter first, rather than set it arbitrarily.

ExtentSel— an array of extent polygon names.

Extent — alternative to ExtentSel; an explicit polygon defined by coordinates.

cl — must always be true; indicates that the polygon is closed.

U — coordinate units, e.g. "mm"

pt — array of coordinate values.

x — x coordinate value

y — y coordinate value

Nets — the net information.

net — array of net information.

<"net-name">

"<no net trace>" — special value referring to traces not in a net.

"<no net fill>" — special value referring to fill polygons not in a net.

<design>:<net> — net within a particular design.

<net> — net within the active design.

<clip> — boolean true/false; if true, the net is clipped against the extent else the net is included but not clipped.

Return Value: None

Example: Using "extent_poly" as the selection extent:

```
oEditor.CutOutSubDesign Array("NAME:Params",  
"Name:=", "EMDesign1_cutout",  
"EMDesign:=", true,  
"SubDesign:=", false,  
"Within:=", true,  
"Without:=", false,  
"ExtentSel:=", Array("extent_poly", ...),  
Array("NAME:Nets",
```

```
"net:=", Array("<no net trace>", true),  
"net:=", Array("<no net fill>", true),  
"net:=", Array("EMDesign1:GND", true) ... ))
```

Example: Example using an explicit polygon as the selection extent:

```
oEditor.CutOutSubDesign Array("NAME:Params",  
"Name:=", "EMDesign1_cutout",  
"EMDesign:=", true,  
"SubDesign:=", false,  
"Within:=", true,  
"Without:=", false,  
"Extent:=", Array(  
"cl:=", true,  
"pt:=", Array(  
"U:=", "mm",  
"x:=", 0,  
"y:=" 0, ) ),  
Array("NAME:Nets",  
"net:=", Array("<no net trace>", true),  
"net:=", Array("<no net fill>", true),  
"net:=", Array("EMDesign1:GND", true) ... ))
```

Example: That will create a cutout around first the pos and then the neg trace from the Sample Project "Diff_Via" under Open Samples/EM/SI:

```
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("Diff_Via")
Set oDesign = oProject.SetActiveDesign("diffViaNominal")
Set oEditor = oDesign.SetActiveEditor("Layout")

oEditor.CutOutSubDesign Array("NAME:Params", "Name:=", "diffViaNominal_pos", "EMDesign:=", _
true, "SubDesign:=", false, "Within:=", true, "Without:=", false, "AutoGenExtent:=", _
true, "Expansion:=", 0.1, "RoundCorners:=", false, "Increments:=", 1, "ExtentSel:=", Array(),
Array("NAME:Nets", "net:=", Array( _
"diffViaNominal:neg", true), "net:=", Array("diffViaNominal:pos", false)))

oEditor.CutOutSubDesign Array("NAME:Params", "Name:=", "diffViaNominal_neg", "EMDesign:=", _
true, "SubDesign:=", false, "Within:=", true, "Without:=", false, "AutoGenExtent:=", _
true, "Expansion:=", 0.1, "RoundCorners:=", false, "Increments:=", 1, "ExtentSel:=", Array(),
```

```
Array("NAME:Nets", "net:=", Array( _  
"diffViaNominal:neg", false), "net:=", Array("diffViaNominal:pos", true)))
```

Note that in this example, each sub design should have its own name, otherwise the results are not well defined. Also note that the "false" after the net indicates that it will be used to build the extent outline. "True" means that the net will be included but will be trimmed.

DeactivateOpen (Layout Editor)

Use: Causes deactivation of one or more components to *open circuit* state.

Syntax: **ActivateOpen** Array ("NAME:components",
 <object_name>, // 1st component name
 <object_name>, // 2nd component, if any
 ...) // etc

VB Example:

```
oEditor.DeactivateShort Array("NAME:components", "2", "3")
```

DeactivateShort (Layout Editor)

Use: Causes deactivation of one or more components to *short circuit* state.

Syntax: **ActivateShort** Array("NAME:components",
 <object_name>, // 1st component name
 <object_name>, // 2nd component, if any
 ...) // etc

VB Example:

```
oEditor.DeactivateShort Array("NAME:components", "2", "3")
```

Defeature Objects (Layout Editor)

Use: Remove irrelevant features from a primitive. Vertices that deviate less than the tolerance from a chord are removed; narrow concave regions less than the tolerance wide are also eliminated. The command can also be used to fix self-intersections (only the 'positive' areas are kept).

Command: DefeatureObjects

Syntax: DefeatureObjects Array("NAME:options", "tolerance:=", <value>, "fix:=", <fix>), Array("NAME:elements", <"primitive">, ...)

Return Value: None

Parameters: <"value">

Type: double, e.g. 0.001

Description: tolerance value

<fix>

Type: boolean (true or false)

Description: if true, then self-intersections are fixed.

< primitive >

Type: text

Description: primitive name.

Example:

```
Set oDesign = oProject.SetActiveDesign("HFSS 3D Layout1")
```

```
Set oEditor = oDesign.SetActiveEditor("Layout")
```

```
oEditor.DefeatureObjects Array("NAME:options", "tolerance=", 1E-006, "fix=", true), Array  
("NAME:elements", "poly_3", "poly_4", "poly_5")
```

DelNetClass

*Use:*Delete a net class in a layout.

*Command:*DelNetClass

*Syntax:*DelNetClass (<names>)

*Return Value:*None

*VB Example:*oEditor.DelNetClass Array ("power")

Delete (Layout Editor)

*Use:*Deletes one or more objects.

Syntax: **Delete** <object_name_list>// names of the objects to be deleted

<object_name_list>:

```
Array ("NAME:elements", <object_name>, <object_name>, ...)
```

VB Example:

```
oEditor.Delete Array("NAME:elements", "circle_0", "rect_2")
```

DeleteNets (Layout Editor)

Removes specified nets from the project.

UI Access	None.		
Parameters	Name	Type	Description
	< <i>netName</i> >	String	Name of unwanted net.
Return Value	None.		

Python Syntax	DeleteNets ([< <i>netName</i> >, ...])
Python Example	<pre>oEditor.DeleteNets (["NET_1", "NET_2", "NET_3"])</pre>

VB Syntax	DeleteNets Array (< <i>netName</i> >, ...)
VB Example	<pre>oEditor.DeleteNets Array("NET_1") oEditor.DeleteNets Array("NET_2", "NET_3", "NET_4")</pre>

DeletePinGroup (Layout Editor)

Use: Deletes a pin group.

Syntax: DeletePinGroup(**STRING** *pin-group-name*)

VB Example: oEditor.DeletePinGroup("J1_GND_Group")

Disconnect (Layout Editor)

Use: Removes the specified object(s) from their current electrical net(s) (if any).

Syntax: **Disconnect** Array("NAME: *elements*",

<object_name>, // 1st object

<object_name>, // 2nd object, if any

...) // etc

VB Example: oEditor.Disonnect Array("NAME:elements", "2:n2")

Duplicate (Layout Editor)

Use: Duplicates layout objects.

Command: The specified objects are duplicated by the given count with the specified offset.

Syntax: Duplicate Array("NAME:options", "count:=", <count data>), Array("NAME:elements", <element names>), Array(<x>, <y>)

Return Value: None

Parameters: <count data> is the number of sets of new objects to be generated (and does not include the original objects)

<element names> is one or more strings with the names of layout objects

<x> is the x value for the offset

<y> is the y value for the offset

```
VB Example: oEditor.Duplicate Array("NAME:options", "count=", 2), Array("NAME:elements", "rect_56"), Array( -0.018, 0.017)
```

DuplicateAcrossLyrs (Layout Editor)

Use: Duplicate selected objects (layout and footprint) to other layers.

Command: Draw > Duplicate > Across Layers

Syntax: DuplicateAcrossLyrs Array("NAME:elements", "element-name",...), Array("NAME:layers", "layer-name",...)

Return Value: None

Parameters: <element-name> // The name of the element to be duplicated.

<layer-name> // The name of the layer to duplicate elements to.

```
VB Example: oEditor.DuplicateAcrossLyrs Array("NAME:elements", "poly_550"), Array("NAME:layers", "Top")
```

Edit (Layout Editor)

Use: Causes modification of one or more existing object(s).

Syntax: **Edit** <Array("NAME:items".....)> ,

```
Parameters: <Array("NAME:items"  
<edit_object_info>, // one object info  
<edit_object_info>, // another one  
...) // etc
```

<edit_object_info>:

```
Array("NAME:item",  
"name:=", <object_name>, // name of the object  
<object_description>) // 'new' object state, type should be consistent with <object_name>:
```

```
<object_description>:  
<circle_description> |  
<rectangle_description> |  
<line_description> |  
<polyon_description> |  
<text_description> |  
<circle_void_description> |  
<rectangle_void_description> |  
<line_void_description> |  
<polyon_void_description> |  
<component_description> |  
<pin_description> |  
<via_description>
```

VB Example:

```
oEditor.Edit Array("NAME:items", Array("NAME:item",  
"name:=", "circle_0", Array("NAME:contents",
```

```
"circleGeometry:=", Array("Layer:=", 6,
"Name:=", "circle_0", "LayerName:=", "Top",
"lw:=", "0mm", "x:=", "-0.008meter",
"y:=", "13.2924281984334mm", "r:=", 10.2924281984334mm)))
```

Edit3DComponentDefinition

Edits definitions of a specified 3D component.

UI Access	Layout > 3D Component Definitions. Select component and click Edit Definition .		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre>Array("NAME:3D Components", "Definitions:=", [<string component name>], "IsLocal:=", <boolean True for local definition; False to save to file>, "EditFilePath:=", <string path of new component file location>)</pre> If saving locally or leaving the file path unaltered, pass empty string for EditFilePath.
Return Value	None.		
Python Syntax	Edit3DComponentDefinition (<Parameters>)		
Python Example	<pre>oEditor.Update3DComponentDefinitions(["NAME:3D Components",</pre>		

	<pre>"Definitions:=" , ["My_Component"], "IsLocal:=" , True, "EditFilePath:=", "")</pre>

VB Syntax	Edit3DComponentDefinition < <i>Parameters</i> >
VB Example	<pre>oEditor.Update3DComponentDefinitions Array("NAME:3D Components", _ "Definitions:=" , Array("My_Component"), "IsLocal:=" , True, "EditFilePath:=" , "")</pre>

EraseMeasurements (Layout Editor)

Use: Causes erasing of ALL measurements currently present.

Command: None.

Syntax: EraseMeasurements

Return Value: None.

Parameters: None.

VB Example: oEditor.EraseMeasurements

Expand (Layout Editor)

Scales an existing circle, square, or polygon. To expand an object while preserving ports, use [ExpandWithPorts](#).

UI Access	Draw > Expand.		
Parameters	Name	Type	Description

	<expansion>	String	Expansion amount, including unit. Value can be negative to shrink an object.
	<join>	String	Corner style. One of: • "ROUND" • "MITER" • "CORNER"
	<replace>	Boolean	True to replace original object; False to create new object.
	<items>	Array	Structured array containing object to expand: <pre>Array("NAME:elements", "<Name of Object to Expand>")</pre>
Return Value	None.		

Python Syntax	<code>Expand(<expansion>, <join>, <replace>, <items>)</code>
Python Example	<code>oEditor.Expand("2mm", "ROUND", True, ["NAME:elements", "circle_0"])</code>

VB Syntax	<code>Expand <expansion>, <join>, <replace>, <items></code>
VB Example	<code>oEditor.Expand "2mm", "ROUND", true, Array("NAME:elements", "poly_0")</code>

ExpandWithPorts (Layout Editor)

Scales an existing circle, square, or polygon and preserves ports. To expand an object without preserving ports, use [Expand](#).

UI Access	Draw > Expand.		
Parameters	Name	Type	Description

	<code><expansion></code>	String	Expansion amount, including unit. Value can be negative to shrink an object.
	<code><join></code>	String	Corner style. One of: • "ROUND" • "MITER" • "CORNER"
	<code><replace></code>	Boolean	True to replace original object; False to create new object.
	<code><items></code>	Array	Structured array containing object to expand: <pre>Array("NAME:elements", "<Name of Object to Expand>")</pre>
Return Value	None.		

Python Syntax	<code>ExpandWithPorts (<expansion>, <join>, <replace>, <items>)</code>
Python Example	<code>oEditor.ExpandWithPorts("2mm", "ROUND", True, ["NAME:elements", "circle_0"])</code>

VB Syntax	<code>ExpandWithPorts <expansion>, <join>, <replace>, <items></code>
VB Example	<code>oEditor.ExpandWithPorts "2mm", "ROUND", true, Array("NAME:elements", "poly_0")</code>

ExportDXF (Layout Editor)

Use: Export DXF

Command: File > Export > AutoCAD

Syntax: ExportDXF

```
Array("NAME:options", "FileName:=", <"filename">, "AcadVersion:=", <version>, "ScaleFactor:=", <scale>),  
Array("NAME:layers", <"layer">, ...)
```

Return Value: None*Parameters:*

<"filename">

Type: text

Description: file path for the export file

<version>

Type: integer

Description: version of AutoCAD. Acceptable values are:

- **0** corresponding to AutoCAD version 13
- **1** corresponding to AutoCAD version 14
- **2** corresponding to AutoCAD 2000-2003
- **3** corresponding to AutoCAD 2004-2006
- **4** corresponding to AutoCAD 2007-2009
- **5** corresponding to AutoCAD 2010-2012

<scale>

Type: double

Description: the scaling of the values written out, e.g. if the data is written out in "mm", the scaling factor will be 1000.

<"layer">

Type: text

Description: names of the layers to be exported.

VB Example:

```
Set oDesign = oProject.SetActiveDesign("HFSS3D1")
Set oEditor = oDesign.SetActiveEditor("Layout")

oEditor.ExportDXF Array("NAME:options", "FileName:=", "C:/Projects/Temp/output.dxf", "AcadVer-
sion:=", 0, "ScaleFactor:=", 1000), Array("NAME:layers", "Assembly Bottom", "Assembly Top", "Bot-
tom", "Bottom Dielectric", "Errors", "Ground", "Measures", "Rats", "Silkscreen Bottom",
"Silkscreen Top", "Symbols", "Top", "Top Dielectric")
```

Python Example:

```
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("05514795_y1_10kv_v02-1mm")
oDesign = oProject.SetActiveDesign("PCB")
oEditor = oDesign.SetActiveEditor("Layout")
oEditor.ExportDXF(
    [
        "NAME:options",
        "FileName:=" , "E:/Anstranlator/resultsEDB/05514795_y1_10kv_v02-1mm_PCB.dxf",
        "AcadVersion:=" , 5,
        "ScaleFactor:=" , 1000
    ],
    [
        "NAME:layers",
```

```
"BOTTOM",  
"Outline",  
"TOP"  
  
l)
```

ExportGDSII (Layout Editor)

Exports results to GDSII.

UI Access	File > Export > GDSII.		
Parameters	Name	Type	Description
	<OptionParameters>	Array	Structured array. Array("NAME:options", "FileName:=", <string, GDSII export file name.>, "NumVertices:=", <integer, maximum number of vertices allowed in a polygon (specify zero if unlimited).>, "ArcTol:=", <double, arc tolerance (in meters); the value used to discretize arcs. Smaller the value the greater the number of edges exported. For 0.1mm use 0.0001.>, "LayerMap:=", Array("entry:=", Array("layer:=", <string, name of a layer to export.>,

			<code>"id:=", <integer, GDSII layer ID to assign the exported layer.>,</code> <code>"include:=", <boolean, if true, the layer is exported.>))</code>
Return Value	None.		

Python Syntax	ExportGDSII(<OptionParameters>)
Python Example	<pre>oEditor.ExportGDSII [("NAME:options", "FileName:=", "C:/Projects/ design.gds", "NumVertices:=", 0, "ArcTol:=", 2E-006, "LayerMap:=", ["entry:=", ["layer:=", "Bottom", "id:=", 2, "include:=", true], "entry:=", ["layer:=", "Top", "id:=", 1,</pre>

	<pre> "include:=", true]]] </pre>

VB Syntax	ExportGDSII <OptionParameters>
VB Example	<pre> oEditor.ExportGDSII Array("NAME:options", "FileName:=", "C:/Projects/ design.gds", "NumVertices:=", 0, "ArcTol:=", 2E-006, "LayerMap:=", Array("entry:=", Array("layer:=", "Bottom", "id:=", 2, "include:=", true), "entry:=", Array("layer:=", "Top", "id:=", 1, "include:=", true))) </pre>

ExportGerber (Layout Editor)

Use: Export the current design to Gerber files.

Command: File > Export > Gerber

Syntax: ExportGerber

```
Array("NAME:options",  
      "FileName:=", <"file name">,  
      "Units:=", <"units">,  
      "IntPlace:=", <integer places>,  
      "DecimalPlace:=", <decimal places>,  
      "Suppress:=", <"suppress">,  
      Array("NAME:GerberPages",  
            "<page>:=", Array(<"layer">, ...), ...))
```

Return Value: None

Parameters: <"file name">

Type: text

Description: prefix for the exported Gerber files; the actual files will be named <file name>_<page>.ger

<"units">

Type: text

Description: unit code, e.g. "in", "mm", etc.

<integer places>

Type: integer

Description: number of integer places to use when formatting the numerical output.

<decimal places>

Type: integer

Description: number of decimal places to use when formatting the numerical output.

<"suppress">

Type: text

Description: either "Leading Zeros", or "Trailing Zeros".

<page>

Type: integer

Description: Gerber page number

<"layer">

Type: text

Description: Layers to be exported to the specified Gerber page.

Example:

```
Set oDesign = oProject.SetActiveDesign("HFSS3D1")
Set oEditor = oDesign.SetActiveEditor("Layout")
oEditor.ExportGerber
Array("NAME:options",
"FileName:=", "C:/ Projects/design.ger",
"Units:=", "in",
"IntPlace:=", 5,
"DecimalPlace:=", 5,
```

```
"Suppress:=", "Leading Zeros",  
Array("NAME:GerberPages", "1:=", Array("Ground"), "2:=", Array("Top") ))
```

ExportNCDrill (Layout Editor)

Use: Export the vias to an NC Drill file

Command: File > Export > NCDrill

Syntax: ExportNCDrill

```
Array("NAME:options",  
"FileName:=", <"file name">,  
"Units:=", <"units">,  
"IntPlace:=", <integer places>,  
"DecimalPlace:=", <decimal places>,  
"SuppressLeadingZero:=", <suppress leading zeros>,  
Array("NAME:NCDFileOptsVec",  
Array("NAME:NCDrillOptions",  
"Top:=", <"layer">,  
"Bottom:=", <"layer">,  
"Create:=", <create>), ...))
```

Return Value: None

Parameters: <"file name">

Type: text

Description: prefix for the exported Gerber files; the actual files will be named <file name>_<page>.ger

<"units">

Type: text

Description: unit code, e.g. "in", "mm", etc.

<integer places>

Type: integer

Description: number of integer places to use when formatting the numerical output.

<decimal places>

Type: integer

Description: number of decimal places to use when formatting the numerical output.

<suppress leading zeros>

Type: boolean

Description: if true, then suppress leading zeros in the output file.

<layer>

Type: text

Description: Start/end layer names; each layer range is written to a separate NC drill file.

<create>

Type: boolean

Description: if true, create the corresponding NC drill file.

Example:


```
Set oDesign = oProject.SetActiveDesign("HFSS3D1")
Set oEditor = oDesign.SetActiveEditor("Layout")
oEditor.ExportNCDrill
Array("NAME:options",
"FileName:=", "C:/ Projects/drill.ncd",
"Units:=", "mm",
"IntPlace:=", 5,
"DecimalPlace:=", 5,
"SuppressLeadingZero:=", true,
Array("NAME:NCDFileOptsVec",
Array("NAME:NCDrillOptions",
"Top:=", "Top",
"Bottom:=", "Ground",
"Create:=", true),
Array("NAME:NCDrillOptions",
"Top:=", "Top",
"Bottom:=", "Bottom",
"Create:=", true)))
```

ExportToHFSS (Layout Editor)

Use: Exports an HFSS 3D Layout design to a new project with an equivalent HFSS 3D design. This command is only valid on HFSS Setup types.

Syntax: ExportToHfss(<setupName>, <pathToExportFile>)

VB Example: oSetup.ExportToHfss("HFSS Setup 1", "C:\Temp\myexport.aedt")

ExportToQ3D (Layout Editor)

Use: Exports an HFSS 3D Layout design to a new project containing an equivalent Q3D design. This command is only valid on HFSS Setup types.

Syntax: ExportToQ3d(<setupName>, <pathToExportFile>)

VB Example: oSetup.ExportToQ3d("HFSS Setup 1", "C:\Temp\myexport.aedt")

FilterObjectList (Layout Editor)

Filters a list of object names using a key/value pair. See: [FindObjects](#).

UI Access	N/A.		
Parameters	Name	Type	Description
	<key>	String	One of: • "Name" to search by object name. • "Type" to search by object type. Valid <value> strings for this type include: 'pin', 'via', 'rect', 'arc', 'line', 'poly', 'plg', 'circle void', 'line void', 'rect void', 'poly void', 'plg void', 'text', 'cell', 'Measurement', 'Port', 'Port Instance', 'Port Instance Port', 'Edge Port', 'component', 'CS', 'S3D', 'ViaGroup' • "Layer" to search by layer.

			An asterisk (*) in the <value> string matches all layers. Vias and pins match using their defined layer range. 'Multi' matches (non via/pin) multi-layer objects. • "Net" to search by net assignment.
	<value>	String	Specific query. Queries use a basic string match. For instance, an asterisk (*) matches anything. All matching is case insensitive.
Return Value	Filtered list of object names.		

Python Syntax	FilterObjectList (<key>, <value>)
Python Example	<pre>RFIN_pins = oLayout.FilterObjectList('Type', 'Pin', oLayout.FindObjects('Net', 'RFIN'))</pre>

VB Syntax	FilterObjectList <key>, <value>
VB Example	<pre>Set objectlist = oEditor.FindObjects("Net", "RFIN") Set RFIN_pins = oEditor.FilterObjectList("Type", "Pin", objectlist)</pre>

FindObjectsByPoint (Layout Editor)

Use: Get a list of all objects intersected by the specified Point object on a given layer.

Syntax: oLayout.FindObjectsByPoint(<oPoint>, <Layer>)

Return Value: Object list.

VB Example:

```
objs = oLayout.FindObjectsByPoint(oLayout.Point().Set(-1.149e-3, 3.465e-3), 'L3')
```

FindObjectsByPolygon (Layout Editor)

Use: Get a list of all objects intersecting the specified Polygon object on the given layer.

Syntax: oLayout.FindObjectsByPolygon(<oPolygon>, <Layer>)

Return Value: Object list.

VB Example:

```
# Find all objects intersecting box = {(0,0), (1e-3,1e-3)}
p0 = oLayout.Point().Set(0, 0)
p1 = oLayout.Point().Set(1e-3, 0)
p2 = oLayout.Point().Set(1e-3, 1e-3)
p3 = oLayout.Point().Set(0, 1e-3)
box = oLayout.Polygon().AddPoint(p0).AddPoint(p1).AddPoint(p2).AddPoint(p3).SetClosed(True)
objs = oLayout.FindObjectsByPolygon(box, '*')
```

FlipHorizontal (Layout Editor)

Use: Causes horizontal flipping of one or more specified objects (mirroring about a vertical axis given its X coordinate).

Syntax: **FlipHorizontal** <object_name_list>,//objects to be flipped <position_or_center> // (X of flip_axis / not used)

VB Example:

```
oEditor.FlipHorizontal Array("NAME:elements", "circle_0", "rect_2"),
```

```
Array(0.01, -0.001)
```

FlipVertical (Layout Editor)

Use: Causes vertical flipping of one or more specified objects (mirroring about a horizontal axis given its Y coordinate).

Syntax: **FlipVertical** <object_name_list>, //objects to be flipped <position_or_center> // not used, flip axis

VB Example: oEditor.FlipVertical Array("NAME:elements", "circle_0", "rect_2"), Array(0.01, -0.001)

GeometryCheckAndAutofix

Use: Runs Geometry Check and optionally applies autofixes.

Command: **HFSS 3D Layout > Geometry Check**

Syntax: GeometryCheckAndAutofix <ChecksArray>,

minimum_area_meters_squared,

<FixesArray>

Return Value: None

Parameters: **<ChecksArray>** - **Array("NAME:checks", <check 1>, <check 2>, ..., <check n>)**

Specify the checks that should be included. Specifying fewer checks may speed up execution but may also result in less problems reported in the message manager (or the logfile) and consequently less problems that can be fixed by autofixes.

The following are valid checks that can be specified:

- "Self-Intersecting Polygons"

- "Disjoint Nets (Floating Nodes) "
- "DC-Short Errors"
- "Identical/Overlapping Vias"
- "Misalignments"

There may be no checks, all 5 of the checks, or anything in between. The order that checks are specified in is not relevant.

minimum_area_meters_squared

Specify a decimal value for the minimum area (e.g. .000015) optionally in scientific notation (e.g. 2E-006). Cutouts smaller than this minimum area may be ignored during validation check.

<FixesArray> - Array("NAME:fixes", <fix 1>, <fix 2>, ..., <fix n>)

Specify the autofixes that should be applied if they are found by a check.

The following are valid fixes that can be specified:

- "Self-Intersecting Polygons"
- "Disjoint Nets",
- "Identical/Overlapping Vias"
- "Traces-Inside-Traces Errors"
- "Misalignments (Planes/Traces/Vias) "

There may be no fixes specified, all 5 fixes specified, or anything in between. The order that fixes are specified in is not relevant.

Example:

```
oEditor.GeometryCheckAndAutofix _
```

```
Array("NAME:checks", "Self-Intersecting Polygons", _  
    "Disjoint Nets (Floating Nodes)", "DC-Short Errors", _  
    "Identical/Overlapping Vias", "Misalignments"), _  
    "minimum_area_meters_squared:=", 2E-006, _  
Array("NAME:fixes", "Self-Intersecting Polygons", _  
    "Disjoint Nets", "Identical/Overlapping Vias", _  
    "Traces-Inside-Traces Errors", _  
    "Misalignments (Planes/Traces/Vias)")
```

GetAllLayerNames (Layout Editor)

Use: Informational.

Command: None.

Syntax: GetAllLayerNames

Return Value: Array of strings which are the names of all layers in the layout, blackbox, or footprint.

VB Example: None

GetBBox (Layout Editor)

Use: Get a [Polygon Object](#) defining the bounding box for the specified object name, or None if the object does not exist.

Syntax: oLayout.GetBBox(<Name>)

Return Value: Polygon object.

VB Example: oPolygon = oLayout.GetBBox('rect_10')

GetCSObjects (Layout Editor)

Use: Given a coordinate system name, returns a list of the objects inside. Coordinate system names can be found using FindObjects ('Type', 'CS'). See [Find Objects](#).

Syntax: oLayout.GetCSObjects(<CS_name>)

Return Value: List of objects in the coordinate system.

VB Example: CS_objects = oLayout.GetCSObjects("CS_1")

GetCompInstanceFromRefDes (Layout Editor)

Use: Informational.

Command: None.

Syntax: GetCompInstanceFromRefDes (<object_name>)

Return Value: Returns IDispatch for CompInstance.

Parameters: <object_name> // string is refDes

VB Example: oEditor.GetCompInstanceFromRefDes("A1")

GetComponentInfo (Layout Editor)

Use: Informational.

Command: None.

Syntax: GetComponentInfo<compID>

Return Value: array of strings, as follows:

 "ComponentName=string"

 "PlacementLayer=string"

"LocationX=*number*"

"LocationY=*number*"

"BBoxLLx=*number*"

"BBoxLLy=*number*"

"BBoxURx=*number*"

"BBoxURy=*number*"

"Angle=*number*" (*in degrees*)

"Flip=true or false"

"Scale=*number*"

Parameters: The LayoutComp's ID

VB Example: dim info
sys= oEditor.GetComponentInfo ("1")

GetComponentPinInfo (Layout Editor)

Use: Informational.

Command: None.

Syntax: GetComponentPinInfo<compID, pinName>

Return Value: retval = array of strings, as follows:

"X=val"

"Y=val"

"Angle=val"

"Flip=true/false"

"WireID=string"

Parameters: compID = The LayoutComp's ID

pinName = The name of the pin

VB Example: dim info

```
info = oEditor.GetComponentPinInfo ("1", "n1")
```

GetComponentPins (Layout Editor)

Use: Informational.

Command: None.

Syntax: GetComponentPins<compID>

Return Value: array of strings, which are the names of all the component's pins

Parameters: The LayoutComp's ID

CompInst@<ComponentName>;<CompInstID>

VB Example:

```
' -----
```

```
' Script Recorded by Ansys Electronics Desktop
```

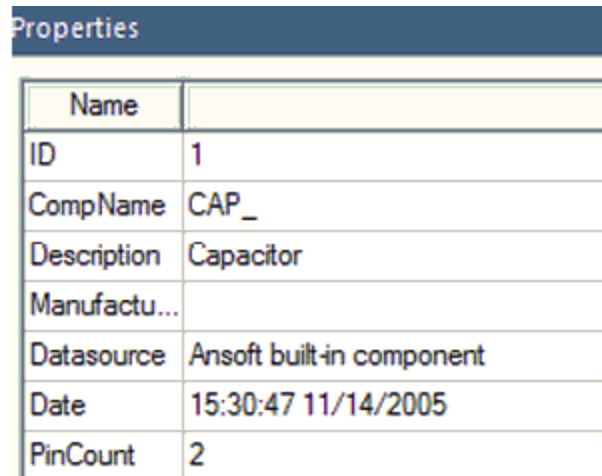
```
' Component Name is CAP_ and ID is 1
```

```
' -----
```

```
dim pins
```

```
pins = oEditor.GetComponentPins ("1")
```

Note: For the documented example, the capacitor component name is CAP_ and its ID is 1. If you can select the item of interest in the schematic, the **Properties** window gets updated with the corresponding component name, its ID, and other details as shown in the following figure.



The screenshot shows a 'Properties' window with a table of component details. The table has two columns: 'Name' and a value column. The rows are as follows:

Name	
ID	1
CompName	CAP_
Description	Capacitor
Manufactu...	
Datasource	Ansoft built-in component
Date	15:30:47 11/14/2005
PinCount	2

GetEditorName (Layout Editor)

Use: Informational.

Command: None.

Syntax: GetEditorName()

Return Value: Returns the name of the editor.

VB Example:

```
dim info
info = oEditor.GetEditorName
```

GetMaterialList (Layout Editor)

Use: Get the names of all the materials for a layout.

Command: None.

Syntax: GetMaterialList

Return Value: Array of strings.

Parameters: None.

```
VB Example: Dim materialNames  
            materialNames = oEditor.GetMaterialList
```

GetNetClassNets

Use: Gets the list of nets in a net class.

Command: GetNetClassNets

Syntax: GetNetClassNets (<name>)

Return Value: None

```
VB Example: oEditor. GetNetClassNets Array ("power")
```

GetNetClasses

Use: Gets all the net classes in a layout.

Command: GetNetClasses

Syntax: GetNetClasses ()

*Return Value:*None

VB Example:`oEditor. GetNetClasses`

GetNetConnections (Layout Editor)

Use: Informational.

Command: None.

Syntax: GetNetConnections(<netName>)

Return Value: Array of strings containing the connections for the net identified by the

netName argument. The strings in the array are in one of four formats:

"ComponentPin compID pinname x y EdgePort"

"ComponentPin compID pinname x y PinPadstack: padstackName"

"InterfacePort portname portID x y Padstack: padstackName"

"EdgePort portname portID EdgeInfo: Primitive id, edge index[Primitive id, edge index]"

where *compID* is the component instance identifier, *pinname* is the name of the connected pin, *x* and *y* are the connection point, *padstackName* is the name of the padstack, *portname* is the name of the port, *portID* is the identifier for the interface port, and *id* and *index* identify edges involved in an edgeport.

Parameters: <netName>

Type: String

The name of the net.

VB Example: netArray = oEditor.GetNetConnections("net_1")

GetPolygon (Layout Editor)

Use: Get a [Polygon Object](#) for the specified object name, or None if the object does not exist. Returned polygon is as rendered, in SI units

Syntax: oLayout.GetPolygon(<Name>)

Return Value: Polygon object.

VB Example: oPolygon = oLayout.GetPolygon('line_37')

GetPolygonDef (Layout Editor)

Use: Get a [Polygon Object](#) for the specified object name, or None if the object does not exist. Returned polygon is as rendered, in SI units. For trace types, this corresponds to the center line.

Syntax: oLayout.GetPolygonDef(<Name>)

Return Value: Polygon object.

VB Example: oPolygon = oLayout.GetPolygonDef('line_37')

GetPortInfo (Layout Editor)

Use: Request information on a port or pin.

Command: None.

Syntax: GetPortInfo<"name">

Return Value: an array of text as follows:

For pins and ports that are not edge ports:

Name=<name>

Type=Pin, Padstack: <padstack definition name>

X=<X coordinate>

Y=<Y coordinate>

ConnectionPoints=<connection description>; <connection description>,...

NetName=<net name>

For edge ports:

Name=<name>

Type=EdgePort

X=<X coordinate>

Y=<Y coordinate>

ConnectionPoints=<connection description>; <connection description>,...

NetName=<net name>

<name>

Text containing the name of the pin or port

<padstack definition name>

Text containing the name of the associated padstack definition

<X coordinate>

Double indicating the X location of the pin or port location

<Y coordinate>

Double indicating the Y location of the pin or port location

<connection description>

< X coordinate> <Y coordinate> Dir:<direction> Layer: <layer name>

<direction>

Either NONE or a double giving the angle in degrees for the connection direction

<layer name>

Name of the layer for the connection point being described.

Example returned values:

Name=Pin1

Type=Pin, Padstack: Padstack

X=0.000744

Y=0.015537

ConnectionPoints= 0.000744 0.015537 Dir:NONE Layer: Top; 0.000744 0.015537

Dir:NONE Layer: Ground; 0.000744 0.015537 Dir:NONE Layer: Bottom

NetName=Pin1


```
Name=Port1
Type=EdgePort
X=-0.008120
Y=0.004264
ConnectionPoints= -0.008120 0.004264 Dir:90.000000 Layer: Bottom
NetName=Port1
```

Parameters: <"name">

Text that contains the name of the port or pin for which information is being requested.

VB Example:

```
Dim conns
conns = oEditor.GetPortInfo("Port1")
```

GetProperties (Layout Editor)

Use: 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 (Layout Editor)

Use: 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.GetPropertyValue("BaseElementTab", _  
"rect_1", "Name")
```

GetSelections [Design]

This script serves no function at the Design level. See: [GetSelections \(Layout Editor\)](#), [GetSelections \(Model Editor\)](#), or Get Selections (Schematic Editor).

Group (Layout Editor)

Use: Groups a set of primitives into a coordinate system.

Command: None.

Syntax: oEditor.Group <CS name>, Array("NAME:elements", <element1>, <element2>, ...)

Return Value: None.

```
VB Example: oEditor.Group "CS_7", Array("NAME:elements", "rect_4", "rect_6")
```

Heal (Layout Editor)

Heals imprecise geometry.

UI Access	From the Layout ribbon, click Healing .		
Parameters	Name	Type	Description
	<HealArray>	Array	Structured array containing these parameters in order: • <NAME:category> • <Type> • <OptionalArgs> • <Tol>
	<NAME:category>	String	Required first parameter which identifies the category of healing (i.e., " Snap ", " Feature ", or " Repair ", such as <NAME:Snap>.
	<Type>	String	Dependent on <NAME:category>. Identifies the type of healing, per category: Snap — "Point", "Arc", or "Grid" Feature — "Voids" or "FloatingBodies" Repair — "Colinear" or "SelfIntersecting"
	<OptionalArgs>	String	Optional parameters (i.e., "Antipads", "ModifyArcs", and/or "InsertPointsInEdges". Multiple <OptionalArgs> can be entered, depending on <NAME:category> and <Type>.
	<Antipads>	Boolean	If <NAME:Feature> and <Type>="Voids", then this optional argument is available: True — voids are converted to antipads False — voids are not converted to antipads
	<ModifyArcs>	Boolean	If <NAME:Snap> and <Point>="Voids", then this optional argument is available: 0 — arcs do not snap to close primitive edges 1 — arcs snap to close primitive edges
	<InsertPointsInEdges>	Boolean	If <NAME:Snap> and <Point>="Voids", then this optional argument is available:

			0 — new points will not appear when healed primitives snap together 1 — new points will appear at midpoints between existing edges and at the new edges created when healed primitives snap together
	<Tol>	Integer	Primitives at or within this tolerance are healed.
Return Value	None		

Python Syntax	oEditor.Heal([<NAME:<category>, <Type>, <OptionalArgs>, <Tol>])
Python Example	<pre> oProject = oDesktop.SetActiveProject("GeometryHealing") oDesign = oProject.SetActiveDesign("TestDesign") oEditor = oDesign.SetActiveEditor("Layout") oEditor.Heal(["NAME:Snap", "Type:=" , "Point", "ModifyArcs:=" , 1, "InsertPointsInEdges:=" , 1, "Tol:=" , "2mm", "Tol:=" , "2mm"]) </pre>

VB Syntax	oEditor.Heal(<NAME:<category>, <Type>, <OptionalArgs>, <Tol>)

VB Example	<pre> oProject = oDesktop.SetActiveProject("GeometryHealing") oDesign = oProject.SetActiveDesign("TestDesign") oEditor = oDesign.SetActiveEditor("Layout") oEditor.Heal Array("NAME:Snap", "Type:=", "Point", "ModifyArcs:=", 1, "InsertPointsInEdges:=", 1, "Tol:=", "2mm", "Tol:=" , "2mm") </pre>

HighlightNet (Layout Editor)

Highlights or removes the highlight from all elements in a net.

UI Access	None.		
Parameters	Name	Type	Description
	<requiredParameterKeyword>	String	Required first parameter which must have the value "Name:Args".
	<requiredNetKeyword>	String	Required second parameter which must have the value "Name:=". Must be the first parameter for each net.
	<netName>	String	Name of the net.
	<requiredFlagKeyword>	String	Required fourth parameter which must have the value "Hi:=". Must be the third parameter for each net.
	<highlightFlag>	Boolean	True if the net should be highlighted. False if it should not.
Return Value	None.		

Python Syntax	HighlightNet(["Name:Args", "Name:=", " <netName>", "Hi:=", <highlightFlag>, ...])
Python Example	<pre>oEditor.HighlightNet(["NAME:Args", "Name:=", "net_0", "Hi:=", True]) </pre>

VB Syntax	HighlightNet Array("Name:Args", "Name:=", " <netName>", "Hi:=", <highlightFlag>, ...)
VB Example	<pre>Set oDesign = oProject.SetActiveDesign("HFSS3D1") Set oEditor = oDesign.SetActiveEditor("Layout") oEditor.HighlightNet Array("NAME:Args", "Name:=", "net_0", "Hi:=", false) </pre>

ImportDXF [Modeler]

Imports a DXF model file.

UI Access	Modeler > Import.		
Parameters	Name	Type	Description
	<Parameters>	Array	Structured array. <pre>Array("NAME:options", "FileName:=" , <string>, "Scale:=" , <float>,</pre>

		<pre>"AutoDetectClosed:=" , <boolean>, "SelfStitch:=" , <boolean>, "DefeatureGeometry:=" , <boolean>, "DefeatureDistance:=" , <integer>, "RoundCoordinates:=" , <boolean>, "RoundNumDigits:=" , <integer>, "WritePolyWithWidthAsFilledPoly:=", <boolean>, "ImportMethod:=" , <integer>, "2DSheetBodies:=" , <boolean>, <layersArray>)</pre>
<layersArray>	Array	Structured array. <pre>Array("NAME:LayerInfo", <layer>, <layer>, <layer>,...)</pre>
<layer>	Array	Structured array. <pre>Array("NAME:<layerName>", "source:=" , <string>, "display_source:=" , <string>, "import:=" , <boolean>, "dest:=" , <string>, "dest_selected:=" , <boolean>, "layer_type:=" , <string>)</pre>

Return Value	None.

Python Syntax	ImportDXF(<Parameters>)
Python Example	<pre> oEditor.ImportDXF(["NAME:options", "FileName:=", "C:/Users/jdoe/Desktop/export.dxf", "Scale:=" , 0.001, "AutoDetectClosed:=" , True, "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, </pre>


```
"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"
],
["NAME:LAYER_2",
"source:="              , "LAYER_2",
"display_source:="      , "LAYER_2",
"import:="              , True,
"dest:="                , "LAYER_2",
"dest_selected:="       , False,
"layer_type:="          , "signal"
]
```

	]
	])

VB Syntax	ImportDXF <Parameters>
VB Example	<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, "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")))</pre>

ImportGDSII

Imports a GDSII file into a new project.

UI Access	File > Import > GDSII.		
Parameters	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. Full path of “technology” (corresponding to <code>-t=technologyfile</code> argument in anstranlator). Pass empty string if none.
	<mapFileName>	String	If technology file was used in place of the control file. Full path to either gdsii mapping file (corresponding to <code>-g=gdsMappingFile</code> argument in anstranlator) or XML control file. Otherwise, pass empty string.
Return Value	None.		

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

VB Syntax	ImportGDSII <gdsiiFileName>, <outputPathName>, <controlFileName>, <mapFileName>
VB Example	<pre>Set oTool = oDesktop.GetTool("ImportExport")</pre>

oTool.ImportGDSII "C:/Files/test.gds", "C:/Files/test.aedb.edb", _ "C:/Files/test.xml"

Intersect (Layout Editor)

Use: Causes Boolean intersecting of 2 or more *primitive* (polygons, rectangles, lines, or circles) objects.

Syntax: Intersect Array ("NAME: *primitives*",

<object_name>, // 1st primitive name

<object_name>, // 2nd primitive, if any

...)// etc

VB Example:

```
oEditor.Intersect Array("NAME:primitives", "circle_0", "rect_2")
```

ModifyNetClass

Use: Modify an existing net class in a layout.

Command: ModifyNetClass

Syntax: ModifyNetClass (<name> <new name> <new description> <new nets name list>)

Return Value: None

VB Example: oEditor.ModifyNetClass "power", "power", "new power nets", Array("A_D2D_VDD_0", "A_D2D_VDD_1", "A_D2D_VSS_0", "VDD_DIG_0", "VDD_DIG_1")

Move (Layout Editor)

Use: Translate this Polygon by the vector specified in the provided Point. Return this Polygon.

Syntax: oPolygon.Move(<oPoint>)

Return Value: This Polygon object.

VB Example: oPolygon = oPolygon.Move(oLayout.Point().Set(1,1))

PageSetup (Layout Editor)

Use: Specifies page setup for printing.

Command: **File>Page Setup**

Syntax: PageSetup <ArgArray>

Return Value: None.

Parameters: <Margins>: Page margins in implicit units of inches.

 <Border>: Integer value indicating to draw border (1) or not to draw (0).

 <DesignVars>: Integer value indicating to draw design vars (1) or not to draw (0).

VB Example: Set oProject = oDesktop.GetActiveProject()

Set oDesign = oProject.GetActiveDesign()

Set oEditor = oDesign.GetActiveEditor()

oEditor.PageSetup Array("NAME:PageSetupData", "margins:=", Array("left:=", 550, "right:=", 550, "top:=", 500, "bottom:=", 500), "border:=", 1, "DesignVars:=", 0)

Point (Layout Editor)

Use: Create and return a [Point Object](#).

Syntax: oLayout.Point()

Return Value: Point object.

VB Example: oPoint = oLayout.Point()

Polygon (Layout Editor)

Use: Create and return a [Polygon Object](#).

Syntax: oLayout.Polygon()

Return Value: Polygon object.

VB Example: oPolygon = oLayout.Polygon()

PositionRelative (Layout Editor)

Use: Position a coordinate system (CS), including 3D structures and components, relative to another object.

Command: Draw > Position Relative

Syntax: There are two forms available:

PositionRelative

Array("NAME:Contents",

```
"RelPos:=", Array(<edge description>)),  
Array("NAME:elements", <coordinate system name>, < coordinate system name >, ...)
```

```
PositionRelative Array("NAME:Contents",  
"RelPos:=", Array("t:=", "pr", "from:=", <edge port or pin name>)),  
Array("NAME:elements", <coordinate system name>, <coordinate system name>, ...)
```

Return Value: None.

Parameters:

```
Array("NAME:elements", "CS_23")
```

<edge description> for primitive edges

"t:=", "pe", "from:=", <"prim">, "pos:=", <edge position>, "et:=", "pe", "prim:=", <"prim">, "edge:=", <edge#>

<"prim">: text that is the primitive name

<edge position>

double between 0 and 1, inclusive

0 indicates the start of the edge

1 indicates the end of the edge

values in between are positions along the edge

<edge#>: integer that is the edge number on the primitive

<edge description> for via edges

"t:=", "pe", "from:=", <"via">, "pos:=", <edge position>, "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)

< coordinate system name >

Text that contains the name of the coordinate system

<edge port or pin name>

Text that contains the name of the edge port or via

VB Example:

```
oEditor.PositionRelative Array("NAME:Contents", "RelPos:=", Array("t:=", "pe",  
"from:=", "poly_0", "pos:=", 0, "et:=", "pe", "prim:=", "poly_0", "edge:=", 3)),  
Array("NAME:elements", "CS_23")
```

```
oEditor.PositionRelative Array("NAME:Contents", "RelPos:=", Array("t:=", "pe",  
"from:=", "via_0", "pos:=", 0, "et:=", "pse", "sel:=", "via_0", "layer:=", 10, "sx:=",  
0.0015, "sy:=", 0.0015, "ex:=", -0.0015, "ey:=", 0.0015, "h:=", 0, "rad:=", 0)),  
Array("NAME:elements", "CS_201")
```

PushExcitations

Allows access to computed excitations for transient and linear frequency solutions. The script command can be accessed from three locations.

UI Access	• Layout Editor • Schematic Editor • Select a Nexxim solution in a 3D Layout design, right click, and choose Push Excitations.		
Parameters	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 =" , <true false>, "Sol:=", "<solution name>")</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", "transient:=", Array("start:=", <start time>, "stop:=", <stop time>, "maxHarmonics:=", <max harmonics>,</pre>

			<pre>"winType:=", <window>, ["widthPct:=", <width percentage>, ["kaiser:=", <Kaiser value>, ["correctCoherentGain:=", true]], "Sol:=", "<solution name>")</pre> winType can have the following values: • Rectangular • Bartlett • Blackman • Hamming • Hanning • Kaiser • Welch • Weber • Lanczos
Return Value	None		

Note:

For more information about the component name, schematic IDs, and formats, see the section [Editor Scripting IDs](#).

VB Syntax	<pre>PushExcitations "<refdes>", Array("NAME:options", "transient:=", ["CalcThevenin =", <true>], Array("start:=", <start time>, "stop:=", <stop time>, "maxHarmonics:=", <max harmonics>, "winType:=", <window>, ["widthPct:=", <width percentage>,"kaiser:=", <Kaier value>," ["correctCoherentGain:=", true]), "Sol:=", "<solution name>")</pre>
VB Example	For a transient solution: <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") 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:=", _ true), "Sol:=", "Transient Setup 1") For a linear frequency solution: Set oEditor = oDesign.SetActiveEditor("SchematicEditor") oEditor.PushExcitations "S1", Array("NAME:options", _</pre>

	"CalcThevenin:=", false, "Sol:=", "LinearFrequency")

RemoveLayer (Layout Editor)

Use: Removes a layer or stackup layer.

Command: Remove Layer from a layout or footprint definition

Syntax: RemoveLayer (<LayerName>)

Return Value: None.

Parameters: <LayerName>

Type: <String>

VB Example: oEditor.RemoveLayer ("T3 C1 sub")

oDefinitionEditor.RemoveLayer("Top3 Footprint")

Note As with other Layout scripting interface commands that modify the layout, this command is not intended for use within scripts that define footprints. The command behavior from within such a script is undefined and may be unexpected. Use the LayoutHost scripting interface commands within scripts that define footprints. Users can, however, execute the **RemoveLayer** command from both general and **IC** modes, although the command functions differently in each environment. Refer to "RemoveLayer (Layout Editor)" on page 25-152.

RemovePortsFromAllNets (Layout Editor)

Use: Removes ports from all the pins in all the nets.

Command: Layout tab under Nets right-click and click **Remove Ports**.

Syntax: RemovePortsFromAllNets

Return Value: None

Parameters: None.

VB Example: oEditor.RemovePortsFromAllNets

RemovePortsFromNet (Layout Editor)

Removes ports from all the pins on the designated nets.

UI Access	In the Layout window, right click the net and click Port > Remove Ports from Net or on the Nets tab of the Nets window, highlight and right click one or more nets and choose Remove Ports .		
Parameters	Name	Type	Description
	< <i>requiredKeyword</i> >	String	Required parameter that must proceed all element names with the value "NAME:Nets".
	< <i>netName</i> >	String	Name of a net. Ports are removed from the port instances of the selected nets.
Return Value	None.		

Python Syntax	RemovePortsFromNet(["NAME:Nets", < <i>netName</i> >, ...])
Python Example	<pre>oEditor.RemovePortsFromNet (["NAME:Nets", "NET_1", "NET_2", "NET_3"])</pre>

VB Syntax	RemovePortsFromNet Array("NAME:Nets", <netName>, ...)
VB Example	<code>oEditor.RemovePortsFromNet Array("NAME:Nets", "NET_1", "NET_2", "NET_3")</code>

RemovePortsOnComponents (Layout Editor)

Remove ports on port instances of selected components. Multiple nets can be listed.

UI Access	Right click and choose Port > Remove Ports From Component .		
Parameters	Name	Type	Description
	< <i>requiredKeyword</i> >	String	Required parameter that must proceed all element names with the value "NAME:elements".
	< <i>elementName</i> >	String	Name of a component. Ports are removed from the port instances of the selected components.
Return Value	None.		

Python Syntax	RemovePortsOnComponents (["NAME:elements", <elementName>,...])
Python Example	<pre>oEditor.RemovePortsOnComponents (["NAME:elements", "0"</pre>

	])

VB Syntax	RemovePortsOnComponents Array("NAME:elements", "<element-name>",...)
VB Example	oEditor.RemovePortsOnComponents Array("NAME:elements", "0")

Rotate (Layout Editor)

Use: Rotate this Polygon counter clockwise (CCW) about a center specified by the provided Point. Angle is in radians. Return this Polygon.

Syntax: oPolygon.Rotate(<AngRad>, <oPoint>)

Return Value: This Polygon object.

VB Example: oPolygon = oPolygon.Rotate(pi/2, oLayout.Point())

SelectAll (Layout Editor)

Use: Select all elements in the layout editor.

Command: None.

Syntax: SelectAll()

Parameters: None

VB Example: Dim removedDefs

removedDefs = oDefinitionEditor.SelectAll()

SetCS (Layout Editor)

Use: Activate a coordinate system (CS).

Command: None.

Syntax: oEditor.SetCS <CS name or blank>

Return Value: None.

```
VB Example: oEditor.SetCS "CS_7" // activate CS_7
oEditor.SetCS "" // activate the 'global CS', i.e. no CS is active
```

SetLayerMapping (Layout Editor)

Use: Set layer mapping.

Command: None.

Syntax: SetLayerMapping <component name>, <design layer>, <footprint layer>

Return Value: None.

Parameters: <component name> is a string that specifies the name of the component

<design layer> is a string that specifies the name of the design layer

<footprint layer> is a string that specifies the name of the footprint layer

```
VB Example: oEditor.SetLayerMapping "2", "Bottom Signal", "Top"
```

SetNetVisible (Layout Editor)

Shows and highlights or hides specified nets in the Layout Editor. Multiple nets can be highlighted or hidden at once.

UI Access	On the Nets tab of the Nets window, highlight and right click on one or more nets. Click Hide or any of the Show options.		
Parameters	Name	Type	Description
	<requiredParameterKeyword>	String	Required first parameter which must have the value "Name:Args".
	<requiredNameKeyword>	String	Required second parameter which must have the value "Name:=". Must be the first parameter for each net.
	<netName>	String	Name of the net
	<requiredFlagKeyword>	String	Required fourth parameter which must have the value "Vis:=". Must be the third parameter for each net.
	<visibilityFlag>	Boolean	True if the net should be visible. False if it should be hidden.
Return Value	None.		

Python Syntax	SetNetVisible(["Name:Args", "Name:=", "<netName>", "Vis:=", <highlightFlag>, ...])
Python Example	<pre> oEditor.SetNetVisible(["NAME:Args", "Name:=", "Net_1", "Vis:=", True]) oEditor.SetNetVisible(["NAME:Args", </pre>

	<pre> "Name:=", "<NO-NET>", "Vis:=", False, "Name:=", "Ground", "Vis:=", False, "Name:=", "Chassis", "Vis:=", True])</pre>

VB Syntax	SetNetVisible Array ("Name:Args", "Name:=", "<netName>", "Vis:=", <visibilityFlag>, ...)
VB Example	<pre> oEditor.SetNetVisible Array("NAME:Args", "Name:=", "NET_1", "Vis:=", true) oEditor.SetNetVisible Array("NAME:Args", "Name:=", "<NO-NET>", "Vis:=", false, "Name:=", "Ground", "Vis:=", false, "Name:=", "Chassis", "Vis:=", true)</pre>

StitchLines (Layout Editor)

Use: Stitch together connected or crossing lines into polygons and lines.

Command: StitchLines

Syntax: StitchLines Array("NAME:elements", <"line">, ...)

Return Value: None

Parameters: <"line">

Type: text

Description: line name.

Example:

```
Set oDesign = oProject.SetActiveDesign("HFSS3D1")
Set oEditor = oDesign.SetActiveEditor("Layout")
oEditor.StitchLines Array("NAME:elements", "line_1", "line_2", "line_3")
```

Subtract (Layout Editor)

Use: Causes boolean subtracting of one or more *primitive* (polygons, rectangles, lines, or circles) object(s) from another one.

Syntax: Subtract Array ("NAME: *primitives*",

<object_name>, // Primitive to subtract from

<object_name>, // 1nd primitive to subtract, if any

...) // etc

VB Example:

```
oEditor.Intersect Subtract ("NAME:primitives", "circle_0", "rect_2")
```

ToggleViaPin (Footprint Editor)

Use: Selected pins are changed to vias. Selected vias are changed to pins.

Syntax: ToggleViaPin <NAME:elements", <object_name> ... // objects to be toggled

Return Value: None.

Parameters: <object_name>

Type: <String>

VB Example: oEditor.ToggleViaPin Array("NAME:elements", "via_195")

Ungroup (Layout Editor)

Use: Reverses a group operation; deletes a coordinate system but leaves the primitives.

Command: None.

Syntax: oEditor.Ungroup Array("NAME:elements", <CS1>, <CS2>, ...)

Return Value: None.

VB Example: oEditor.Ungroup Array("NAME:elements", "CS_7")

Unite (Layout Editor)

Use: Causes Boolean uniting of 2 or more *primitive* (polygons, rectangles, lines, or circles) objects.

Syntax: **Unite** Array ("NAME: *primitives*",

<object_name>, // 1st primitive name

<object_name>, // 2nd primitive, if any

...) // etc

VB Example:

```
oEditor.Unite Array("NAME:primitives", "circle_0", "rect_2")
```

UnselectAll (Layout Editor)

Use: Unselect all active selections in the layout editor.

Command: Edit > Unselect All

Syntax: UnselectAll()

Parameters: None

VB Example: oLayout.UnselectAll()

ZoomToFit (Layout Editor)

Use: Set the current schematic zoom to fit the contents of the currently visible page

Command: None

Syntax: ZoomToFit()

Return Value: None

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.

[ChangeProperty \(Symbol Editor\)](#)

[CreateCircle \(Symbol Editor\)](#)

[CreateLine \(Symbol Editor\)](#)

[CreatePin \(Symbol Editor\)](#)

[CreatePolygon \(Symbol Editor\)](#)

[CreateRectangle \(Symbol Editor\)](#)

[CreateText \(Symbol Editor\)](#)

[Delete \(Symbol Editor\)](#)

[FlipHorizontal \(Symbol Editor\)](#)

[FlipVertical \(Symbol Editor\)](#)

[GetProperties \(Symbol Editor\)](#)

[GetPropertyValue \(Symbol Editor\)](#)

[Move \(Symbol Editor\)](#)

[Paste \(Symbol Editor\)](#)

[RemovePort \(Symbol Editor\)](#)

[Rotate \(Symbol Editor\)](#)

[SelectAll \(Symbol Editor\)](#)

[SetPropertyValue \(Symbol Editor\)](#)

[Undo \(Symbol Editor\)](#)

[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
```


Python Syntax	<code>AlignHorizontal(["NAME:Selections", "Selections:=", [<selections>]], 0)</code>
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _ "Simplorer 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
```

Python Syntax	<code>AlignVertical(["NAME:Selections", "Selections:=", [<selections>]], 0)</code>

Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSymbolEditor.AlignVertical (["NAME:Selections","Selections:=", ["SchObj@11"]], 0)</pre>

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>,  
<NewPropArray>,  
<ChangedPropsArray>,  
<DeletedPropsArray>)
```

```
<PropServersArray>  
Array("Name:PropServers", <PropServer>,  
<PropServer>, ...)
```

```
<NewPropArray>  
Array("Name:NewProp", <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:=", "lnH"))))
```

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

Python Syntax	CloseEditor()
Python Example	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)
```

Python Syntax	CreateArc()
Python Example	<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)
```

Python Syntax	CreateCircle()
Python Example	<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)
```

Python Syntax	CreateLine()
Python Example	<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)
```

Python Syntax	CreatePin()
Python Example	<pre>oDefinitionEditor.CreatePin (["NAME:PinData", _</pre>

	<pre>"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)
```

Python Syntax	CreatePolygon()
Python Example	<pre>oDefinitionEditor.CreatePolygon (["NAME:PolygonData", _ "X1:=", -0.001524, "Y1:=", 0.000508, _ "X2:=", 0.002286, "Y2:=", -0.002032, "Id:=", 18], _ ["NAME:Attributes", "Page:=", 1])</pre>

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)
```

Python Syntax	CreateRectangle()
Python Example	<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)
```

Python Syntax	CreateText()
Python Example	<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)
```

Python Syntax	<code>Cut(["NAME:Selections", "Selections:=", [<selections>]])</code>
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", _ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSymbolEditor.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.GetPropertyValue("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.

UI Access	View > Pan.		
Parameters	Name	Type	Description
	<argSelections>	Array of doubles	First double pans along x-axis; second double pans along y-axis.
Return Value	None.		

Python Syntax	Pan (<argSelections>)
Python Example	<pre>oDefinitionEditor.Pan([-0.00922653869663931, 0.00686839904222147])</pre>

VB Syntax	Pan <argSelections>
VB Example	<pre>oDefinitionEditor.Pan Array(-0.00922653869663931, 0.00686839904222147)</pre>

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.

UI Access	Tools > Edit Libraries > Symbols. From Symbol editor, click View > Levels > Remove Level.

Parameters	Name	Type	Description
	</levels>	Array of integers	The levels to be deleted.
Return Value	None.		

Python Syntax	RemoveLevels (</levels>)
Python Example	<code>oSymbolEditor.RemoveLevels ([3,4])</code>

VB Syntax	RemoveLevels </levels>
VB Example	<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",  
    "Simplorer 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()
```

Python Syntax	SelectAll()
Python Example	removedDefs = oSymbolManager.SelectAll()

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"))

Python Syntax	SendToBack(["NAME:Selections", "Selections:=", [<Object>, <Object>, ...]])
Python Example	oDefinitionEditor.SendToBack(["NAME:Selections", "Selections:=", ["SchObj@10"]])

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	Tools > Edit Libraries > Symbols. From Symbol editor, click View > Levels. Select the Default check box for desired levels.		
Parameters	Name	Type	Description
	<levels>	Array of integers	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

	<pre>Set oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor", - "Simplorer Elements\basic Elements\Circuit\Passive Elements:R") oldLevel = oSymbolEditor.SetInitialLevels(Array(3,4))</pre>

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.

UI Access	Tools > Edit Libraries > Symbols. From Symbol editor, click View > Levels . Select the Visible check box for the desired levels.

Parameters	Name	Type	Description
	</levels>	Array of integers	The levels to be made visible.
Return Value	None.		

Python Syntax	SetVisibleLevels (</levels>)
Python Example	<code>oSymbolEditor.SetVisibleLevels ([3, 4])</code>

VB Syntax	SetVisibleLevels </levels>
VB Example	<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)
```

ToggleViaPin (Symbol Editor)

Selected pins are changed to vias. Selected vias are changed to pins.

Syntax: ToggleViaPin <NAME:elements", <object_name> ... // objects to be toggled

Return Value: None.

Parameters: <object_name>

Type: <String>

VB Example: oEditor.ToggleViaPin Array("NAME:elements", "via_195")

Undo (Symbol Editor)

Undo the last operation

Command: Edit>Undo

Syntax: Undo

Return Value: None

Parameters: None

VB Example: oDesign.Undo

Python Syntax	Undo()
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor",_ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSymbolEditor.Undo()</pre>

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

Python Syntax	ZoomToFit()
Python Example	<pre>oSymbolEditor = oProject.SetActiveDefinitionEditor("SymbolEditor",_ "Simplorer Elements\basic Elements\Circuit\PassiveElements:R") oSymbolEditor.ZoomToFit()</pre>

Footprint Editor Scripts

Footprint editor script commands are accessed with a definition editor.

```
Set oDefinitionEditor = oProject.SetActiveDefinitionEditor("FootprintEditor", "MyFootprint")
```

The footprint editor script commands are listed below.

[AddLayer \(Footprint Editor\)](#)

[AddStackupLayer \(Footprint Editor\)](#)

[ChangeLayers \(Footprint Editor\)](#)

[ChangeOptions \(Footprint Editor\)](#)

[CreateCircle \(Footprint Editor\)](#)

[CreateCircleVoid \(Footprint Editor\)](#)

[CreateEdgePort \(Footprint Editor\)](#)

[CreateLine \(Footprint Editor\)](#)

[CreateLineVoid \(Footprint Editor\)](#)

[CreateMeasure \(Footprint Editor\)](#)

[CreatePolygonVoid \(Footprint Editor\)](#)

[CreatePin \(Footprint Editor\)](#)

[CreatePolygon \(Footprint Editor\)](#)

[CreateRectangle \(Footprint Editor\)](#)

[CreateText \(Footprint Editor\)](#)

[CreateVia \(Footprint Editor\)](#)

[Duplicate \(Footprint Editor\)](#)

[Edit \(Footprint Editor\)](#)

[EraseMeasurements \(Footprint Editor\)](#)

[FlipHorizontal \(Footprint Editor\)](#)

[FlipVertical \(Footprint Editor\)](#)

[GetAllLayerNames \(Footprint Editor\)](#)

[GetLayerInfo \(Footprint Editor\)](#)

[GetProperties \(Footprint Editor\)](#)

[GetStackupLayerNames \(Footprint Editor\)](#)

[Intersect \(Footprint Editor\)](#)

[Move \(Footprint Editor\)](#)

[PageSetup \(Footprint Editor\)](#)

[RemoveLayer \(Footprint Editor\)](#)

[RemovePort \(Footprint Editor\)](#)

[Rotate \(Footprint Editor\)](#)[SetPropertyValue \(Footprint Editor\)](#)[Subtract \(Footprint Editor\)](#)[ToggleViaPin \(Footprint Editor\)](#)[Unite \(Footprint Editor\)](#)[ZoomToFit \(Footprint Editor\)](#)

AddLayer (Footprint Editor)

Add a layer in a footprint definition. This command can take all parameters available in [ChangeLayers \(Footprint Editor\)](#) but cannot be recorded.

UI Access	None		
Parameters	Name	Type	Description
	<LayerArray>	Array	An array that contains these parameters in order: • <Layername> • <Type> • <TopBottom> • <Color> • <Transparency> • <Pattern> • <VisFlag> • <Locked> • <Draw>
	<LayerName>	String	Name of layer.
	<Type>	String	Type of layer, such as "user", "signal", "dielectric", "soldermask", "solder-paste", "silkscreen", "assembly", "wirebond", "dielectric", "glue", "user", "Post-

			processing", or "outline".
	<TopBottom>	String	Optional. Indicates whether the layer is the top or bottom. Acceptable values are "top", "bottom", or "neither".
	<Color>	Integer	Optional. A code to indicate the color of the layer. It is an RGB code in the hex format <i>bbggrr</i> . For example red (#0000ff) is 255, green (#00ff00) is 65280, and blue (#ff0000) is 16711680.
	<Transparency>	Integer	Optional. A percentage that indicates the transparency of the layer between 0 (opaque) and 100 (completely transparent and not visible).
	<Pattern>	Integer	Optional. A code for the pattern of the layer: 0 is hollow, 1 is solid, and the other patterns 3-8 are various hatch patterns.
	<VisFlag>	Integer	Sets visibility for all objects on the layer using the following additive code: • 1 makes primitives (rectangles, circles, polygons) visible. • 2 makes lines (also known as paths or traces) visible. • 4 makes pads (from padstacks, aka vias) visible. • 8 makes holes (or vias) visible. • 16 makes components visible. • 32 makes the mesh overlay visible if they are plotted. • 64 makes the mesh for the background material visible. For example, a value of 25 (1+8+16) makes primitives, vias, and components visible. Use 127 to show everything and 0 to hide all.
	<Locked>	Boolean	Optional. Whether the layer is locked.
	<Draw>	Integer	Optional. Allows drawing on the layer as wireframe, outlines, or solid. Acceptable values are -1 for wireframe, 0 for normal, and 1 for solid.
Return Value	None		

Python Syntax	<pre>oEditor.AddLayer(["NAME:layers", "Name:=", "<LayerName>", "Type:=", "<Type>", "Top Bottom:=", "<TopBottom>", "Color:=", "<Color>", "Transparency:=", "<Transparency>", "Pattern:=", "<Pattern>", "VisFlag:=", "<VisFlag>", "Locked:=", "<Locked>", "DrawOverride:=", "<Draw>",])</pre>
Python Example	<pre>oEditor = oDesign.SetActiveEditor("Footprint") oEditor.AddLayer(["NAME:layers", "Name:=", "signal", "Type:=", "signal", "Top Bottom:=", "neither", "Color:=", 16711680,])</pre>

	<pre>"Transparency:=", 0, "Pattern:=", 1, "VisFlag:=", 127, "Locked:=", False, "DrawOverride:=", 0,])</pre>

VB Syntax	<pre>oEditor.AddLayer Array("NAME:stackup layer", "Name:=", "<LayerName>", "Type:=", "<Type>", "Top Bottom:=", "<TopBottom>", "Color:=", <Color>, "Transparency:=", <Transparency>, "Pattern:=", <Pattern>, "VisFlag:=", <VisFlag>, "Locked:=", <Locked>, "DrawOverride:=", <Draw>,)</pre>
VB Example	<pre>Set oEditor = oDesign.SetActiveEditor("Footprint") oEditor.AddLayer Array("NAME:stackup layer", "Name:=", _ "NewLayer", "Type:=", "dielectric", "Top Bottom:=", "neither", "Color:=", _ 3361318, "Transparency:=", 60, "Pattern:=", 1, "VisFlag:=", 127, "Locked:=", _ false, "DrawOverride:=", 0,)</pre>

AddStackupLayer (Footprint Editor)

Adds a stackup layer in a footprint definition. This command can take all parameters available in [ChangeLayers \(Footprint Editor\)](#) but cannot be recorded.

UI Access	None

Parameters	Name	Type	Description
	<LayerArray>	Array	An array that contains these parameters in order: • <Layername> • <Type> • <TopBottom> • <Color> • <Transparency> • <Pattern> • <VisFlag> • <Locked> • <Draw> • and an array with the parameters: • <Thickness> • <LowerElevation> • <Material>
	<LayerName>	String	Name of layer.
	<Type>	String	Type of layer, such as "user", "signal", "dielectric", "soldermask", "solderpaste", "silkscreen", "assembly", "wirebond", "dielectric", "glue", "user", "Post-processing", or "outline".
	<TopBottom>	String	Optional. Indicates whether the layer is the top or bottom. Acceptable values are "top", "bottom", or "neither".
	<Color>	Integer	Optional. A code to indicate the color of the layer. It is an RGB code in the hex format <i>bbggrr</i> . For example red (#0000ff) is 255, green (#00ff00) is 65280, and blue (#ff0000) is 16711680.
	<Transparency>	Integer	Optional. A percentage that indicates the transparency of the layer between 0 (opaque) and 100 (completely transparent and not visible).
	<Pattern>	Integer	Optional. A code for the pattern of the layer: 0 is hollow, 1 is solid, and the other patterns 3-8 are various hatch patterns.
	<VisFlag>	Integer	Sets visibility for all objects on the layer using the following additive code:

			• 1 makes primitives (rectangles, circles, polygons) visible. • 2 makes lines (also known as paths or traces) visible. • 4 makes pads (from padstacks, aka vias) visible. • 8 makes holes (or vias) visible. • 16 makes components visible. • 32 makes the mesh overlay visible if they are plotted. • 64 makes the mesh for the background material visible. For example, a value of 25 (1+8+16) makes primitives, vias, and components visible. Use 127 to show everything and 0 to hide all.
	<Locked>	Boolean	Optional. Whether the layer is locked.
	<Draw>	Integer	Optional. Allows drawing on the layer as wireframe, outlines, or solid. Acceptable values are -1 for wireframe, 0 for normal, and 1 for solid.
	<Thickness>	String	Thickness of the layer, contains the measurement and units.
	<LowerElevation>	String	The elevation of the lower edge of the layer, contains the measurement and units.
	<Material>	String	The material the layer is made of.
Return Value	None		

Python Syntax	<pre>oEditor.AddStackupLayers(["NAME:stackup layer", "Name:=", "<LayerName>", "Type:=", "<Type>",</pre>

	<pre> "Top Bottom:=", "<TopBottom>", "Color:=", <Color>, "Transparency:=", <Transparency>, "Pattern:=", <Pattern>, "VisFlag:=", <VisFlag>, "Locked:=", <Locked>, "DrawOverride:=", <Draw>, ["NAME:sublayer", "Thickness:=", "<Thickness>", "LowerElevation:=", "<LowerElevation>", "Material:=", "<Material>",]]) </pre>
Python Example	<pre> oEditor = oDesign.SetActiveEditor("Footprint") oEditor.AddStackupLayer(["NAME:stackup layer", "Name:=", "Cover", "Type:=", "signal", "Top Bottom:=", "neither", </pre>

	<pre>"Color:=", 16711680, "Transparency:=", 0, "Pattern:=", 1, "VisFlag:=", 127, "Locked:=", False, "DrawOverride:=", 0, ["NAME:Sublayer", "Thickness:=", "0mil", "LowerElevation:=", "20mil", "Material:=", "gold",]])</pre>

VB Syntax	<code>oEditor.AddStackupLayers Array("NAME:stackup layer", "Name:=", "<LayerName>", "Type:=", "<Type>", "Top Bottom:=", "<TopBottom>", "Color:=", <Color>, "Transparency:=", <Transparency>, "Pattern:=", <Pattern>, "VisFlag:=", <VisFlag>, "Locked:=", <Locked>, "DrawOverride:=", <Draw>, Array("NAME:Sublayer", "Thickness:=", "<Thickness>", "LowerElevation:=", "<LowerElevation>", "Material:=", <Material>"))</code>
VB Example	<pre>Set oEditor = oDesign.SetActiveEditor("Footprint") oEditor.AddStackupLayers Array("NAME:stackup layer", "Name:=", _ "NewLayer", "Type:=", "dielectric", "Top Bottom:=", "neither", "Color:=", _</pre>

```
3361318, "Transparency:=", 60, "Pattern:=", 1, "VisFlag:=", 127, "Locked:=", _
false, "DrawOverride:=", 0, Array("NAME:Sublayer", "Thickness:=", _
"1.6mm", "LowerElevation:=", "0mm", "Material:=", "FR4_epoxy"))
```

ChangeLayer (Footprint Editor)

Changes one or more attributes or parameters of a single layer.

Note: To execute the command, all of the following parameters must be entered in the command line even if the user only intends to change one attribute.

UI Access	Create or change a layer in the Edit Layers window.		
Parameters	Name	Type	Description
	<LayerArray>	Array	An array that contains these parameters in order: • <mode> • <Layername> • <ID> • <Type> • <TopBottom> • <Color> • <Transparency> • <Pattern> • <VisFlag> • <Locked>

		• <Draw> • <Subarray>
<mode>	String	Type of stackup. Can be "Laminate", "Multizone", or "Overlap".
<LayerName>	String	Name of layer.
<ID>	Integer	A numerical identification for the layer.
<Type>	String	Type of layer, such as "user", "signal", "dielectric", "soldermask", "solder-paste", "silkscreen", "assembly", "wirebond", "dielectric", "glue", "user", "Post-processing", or "outline".
<TopBottom>	String	<i>Optional.</i> Indicates whether the layer is the top or bottom. Acceptable values are "top", "bottom", or "neither".
<Color>	Integer	<i>Optional.</i> A code to indicate the color of the layer. It is an RGB code in the hex format <i>bbggrr</i> . For example red (#0000ff) is 255, green (#00ff00) is 65280, and blue (#ff0000) is 16711680.
<Transparency>	Integer	<i>Optional.</i> A percentage that indicates the transparency of the layer between 0 (opaque) and 100 (completely transparent and not visible).
<Pattern>	Integer	<i>Optional.</i> A code for the pattern of the layer: 0 is hollow, 1 is solid, and the other patterns 3-8 are various hatch patterns.
<VisFlag>	Integer	Sets visibility for all objects on the layer using the following additive code: • 1 makes primitives (rectangles, circles, polygons) visible. • 2 makes lines (also known as paths or traces) visible. • 4 makes pads (from padstacks, aka vias) visible. • 8 makes holes (or vias) visible. • 16 makes components visible. • 32 makes the mesh overlay visible if they are plotted. • 64 makes the mesh for the background material visible. For example, a value of 25 (1+8+16) makes primitives, vias, and components visible. Use 127 to show everything and 0 to hide all.
<Locked>	Boolean	<i>Optional.</i> Whether the layer is locked.

	<DrawOverride>	Integer	<i>Optional.</i> Allows drawing on the layer as wireframe, outlines, or solid. Acceptable values are -1 for wireframe, 0 for normal, and 1 for solid.
	<Zones>		
	<Subarray>	Array	<i>Optional.</i> An array that contains these parameters in order: • <Thickness> • <LowerElevation> • <Roughness> • <BotRoughness> • <SideRoughness> • <Material> • <FillMaterial> This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.
	<Thickness>	String	Thickness of the layer, contains the measurement and units.
	<LowerElevation>	String	The elevation of the lower edge of the layer, contains the measurement and units.
	<Roughness>	String	The roughness of the top of the layer, contains the measurement and units.
	<BotRoughness>	String	The roughness of the bottom of the layer, contains the measurement and units.
	<SideRoughness>	String	The roughness of the side of the layer, contains the measurement and units.
	<Material>	String	The material the layer is made of.
	<FillMaterial>	String	The material to use to fill in around geometry on the layer.
Return Value	None		

Python Syntax	<code>oEditor.ChangeLayer(<LayerArray>)</code>
Python Example	<pre>oEditor = oDesign.SetActiveEditor("Footprint") oEditor.ChangeLayer(</pre>


```
[ "NAME:stackup layer",  
  "Name:="          , "MQ",  
  "ID:="            , 20,  
  "Type:="          , "signal",  
  "Top Bottom:="    , "neither",  
  "Color:="         , 6136303,  
  "Transparency:="  , 60,  
  "Pattern:="       , 1,  
  "VisFlag:="       , 96,  
  "Locked:="        , False,  
  "DrawOverride:="  , 0,  
  "Zones:="         , [],  
  [ "NAME:Sublayer",  
    "Thickness:="    , "0.745um",  
    "LowerElevation:=" , "737.155um",  
    "Roughness:="     , "0um",  
    "BotRoughness:="  , "0um",  
    "SideRoughness:=" , "0um",  
    "Material:="      , "MQ",  
    "FillMaterial:="  , "FR4_epoxy"
```

	<pre>]]) </pre>

VB Syntax	oEditor.ChangeLayer <LayerArray>
VB Example	<pre> Set oEditor = oDesign.SetActiveEditor("Footprint") oEditor.ChangeLayer Array("NAME:stackup layer","Name:=" , "MQ", "ID:=" , 20, "Type:=" , "signal", "Top Bottom:=" , "neither", "Color:=" , 6136303, "Transparency:=" , 60, "Pattern:=" , 1, "VisFlag:=" , 96, "Locked:=" , False, "DrawOverride:=" , 0, "Zones:=" , Array(), Array("NAME:Sublayer", "Thickness:=" , "0.745um", "LowerElevation:=" , "737.155um", "Roughness:=" , "0um", "BotRoughness:=" , "0um", "SideRoughness:=" , "0um", "Material:=" , "MQ", "FillMaterial:=" , "FR4_epoxy")) </pre>

ChangeLayers (Footprint Editor)

Changes one or more attributes or parameters of a layer.

Note: To execute the command, all of the following parameters must be entered in the command line even if the user only intends to change one attribute.

UI Access	Create or change a layer in the Footprint editor with the Edit Layers window.

Parameters	Name	Type	Description
	<LayerArray>	Array	An array that contains these parameters in order: • <Layername> • <ID> • <Type> • <TopBottom> • <Color> • <Transparency> • <Pattern> • <VisFlag> • <Locked> • <Draw> • <Subarray> • <Neg>
	<mode>	String	Type of stackup. Can be "Laminate", "Multizone", or "Overlap" .
	<LayerName>	String	Name of layer.
	<ID>	Integer	A numerical identification for the layer.
	<Type>	String	Type of layer, such as "user", "signal", "dielectric", "soldermask.", "solder-paste", "silkscreen", "assembly", "wirebond", "dielectric", "glue", "user", "Post-processing", or "outline".
	<TopBottom>	String	Optional. Indicates whether the layer is the top or bottom. Acceptable values are "top", "bottom", or "neither".
	<Color>	Integer	Optional. A code to indicate the color of the layer. It is an RGB code in the hex format <i>bbggrr</i> . For example red (#0000ff) is 255, green (#00ff00) is 65280, and blue (#ff0000) is 16711680.
	<Transparency>	Integer	Optional. A percentage that indicates the transparency of the layer between 0 (opaque) and 100 (completely transparent and not visible).

<Pattern>	Integer	Optional. A code for the pattern of the layer: 0 is hollow, 1 is solid, and the other patterns 3-8 are various hatch patterns.
<VisFlag>	Integer	Sets visibility for all objects on the layer using the following additive code: • 1 makes primitives (rectangles, circles, polygons) visible. • 2 makes lines (also known as paths or traces) visible. • 4 makes pads (from padstacks, aka vias) visible. • 8 makes holes (or vias) visible. • 16 makes components visible. • 32 makes the mesh overlay visible if they are plotted. • 64 makes the mesh for the background material visible. For example, a value of 25 (1+8+16) makes primitives, vias, and components visible. Use 127 to show everything and 0 to hide all.
<Locked>	Boolean	Optional. Whether the layer is locked.
<Draw>		Optional. Allows drawing on the layer as wireframe, outlines, or solid. Acceptable values are -1 for wireframe, 0 for normal, and 1 for solid.
<Subarray>	Array	An array that contains these parameters in order: • <Thickness> • <LowerElevation> • <Roughness> • <BotRoughness> • <SideRoughness> • <Material> • <FillMaterial> This array's placeholder does not appear in the syntax or examples below but instead has its contents listed.
<Thickness>	String	Thickness of the layer, contains the measurement and units.
<LowerElevation>	String	The elevation of the lower edge of the layer, contains the measurement and units.

	<Roughness>	String	The roughness of the top of the layer, contains the measurement and units.
	<BotRoughness>	String	The roughness of the bottom of the layer, contains the measurement and units.
	<SideRoughness>	String	The roughness of the side of the layer, contains the measurement and units.
	<Material>	String	The material the layer is made of.
	<FillMaterial>	String	The material to use to fill in around geometry on the layer.
	<Neg>	Boolean	Whether the geometry on the layer is cut away from the layer.
Return Value	None		

Python Syntax	<pre>oEditor.ChangeLayers(["NAME:layers", "Mode:=", "<mode>", ["NAME:pps"], ["NAME:stackup layer", "Name:=", "<LayerName>", "ID:=", <ID>, "Type:=", "<Type>", "Top Bottom:=", "<TopBottom>",</pre>		

	<pre> "Color:=", <Color>, "Transparency:=", <Transparency>, "Pattern:=", <Pattern>, "VisFlag:=", <VisFlag>, "Locked:=", <Locked>, "DrawOverride:=", <Draw>, ["NAME:Sublayer", "Thickness:=", "<Thickness>", "LowerElevation:=", "<LowerElevation>", "Roughness:=", "<Roughness>", "BotRoughness:=", "<BotRoughness>", "SideRoughness:=", "<SideRoughness>", "Material:=", "<Material>", "FillMaterial:=", "<FillMaterial>"], "Neg:=", <Neg>]]) </pre>
Python Example	<pre> oEditor = oDesign.SetActiveEditor("Footprint") oEditor.ChangeLayers(</pre>

```
[  
  "NAME:layers",  
  "Mode:=", "Laminate",  
  [  
    "NAME:pps"  
  ],  
  [  
    "NAME:stackup layer",  
    "Name:=", "Cover",  
    "ID:=", 10,  
    "Type:=", "signal",  
    "Top Bottom:=", "neither",  
    "Color:=", 16711680,  
    "Transparency:=", 0,  
    "Pattern:=", 1,  
    "VisFlag:=", 127,  
    "Locked:=", False,  
    "DrawOverride:=", 0,  
    [  
      "NAME:Sublayer",
```

	<pre> "Thickness:=", "0mil", "LowerElevation:=", "20mil", "Roughness:=", "0um", "BotRoughness:=", "0um", "SideRoughness:=", "0um", "Material:=", "gold", "FillMaterial:=", "FR4_epoxy"], "Neg:=", True]]) </pre>

VB Syntax	<pre> oEditor.ChangeLayers Array("NAME:layers","Mode:=", "<mode>", Array("NAME:pps"), Array("NAME:stackup layer", "Name:=", "<LayerName>", "ID:=", <ID>, "Type:=", "<Type>", "Top Bottom:=", "<TopBottom>", "Color:=", <Color>, "Trans- parency:=", <Transparency>, "Pattern:=", <Pattern>, "VisFlag:=", <VisFlag>, "Locked:=", <Locked>, "DrawOverride:=", <Draw>, Array("NAME:Sublayer", "Thickness:=", "<Thickness>", "LowerElevation:=", "<LowerElevation>", "Rough- ness:=", "<Roughness>", "BotRoughness:=", "<BotRoughness>", "SideRoughness:=", "<SideRoughness>", "Material:=", <Material>,"FillMaterial:=", "<FillMaterial>"),"Neg:=", <Neg>)) </pre>
VB Example	<pre> Set oEditor = oDesign.SetActiveEditor("Footprint") oEditor.ChangeLayers Array("NAME:layers", "Mode:=", "Laminate", Array("NAME:pps"), Array ("NAME:stackup layer", "Name:=", _ "NewLayer", "ID:=", 7, "Type:=", "dielectric", "Top Bottom:=", "neither", "Color:=", _ 3361318, "Transparency:=", 60, "Pattern:=", 1, "VisFlag:=", 127, "Locked:=", _ </pre>

<pre>false, "DrawOverride:=", 0, Array("NAME:Sublayer", "Thickness:=", _ "1.6mm", "LowerElevation:=", "0mm", "Roughness:=", "0um", "BotRoughness:=", _ "0um", "SideRoughness:=", "0um", "Material:=", "FR4_epoxy"), "Neg:=", True))</pre>

ChangeOptions (Footprint Editor)

Use: Changes options for an existing layout. (Does not change global options specified in the registry.) Only those options being changed need to be specified. Options not specified are not affected.

Command: None.

Syntax: ChangeOptions Array("NAME:options", <OptionData>, ...)

Return Value: None

Parameters: <OptionData> can be of varying forms:

 Type: <String>

 Type: integer

 Type: Array(float, float, float, float)

VB Example:

```
oEditor.ChangeOptions Array("NAME:options", _  
"MajorSize:=", "20", _  
"MinorSize:=", "1", "MajorColor:=", 8421376, _  
"MinorColor:=", 16776960, _  
"ShowGrid:=", false, "PageExtent:=", _
```

```
Array( -0.3, -0.1, 0.1, 0.1), _  
"background color:=", 4194368, _  
"DefaultToSketchMode:=", true, _  
"fillMode:=", false, "PixelSnapTolerance:=", 22, _  
"SnapTargetVertex_on:=", _  
false, "SnapTargetEdgeCenter_on:=", false, _  
"SnapTargetObjCenter_on:=", _  
true, "SnapTargetEdge_on:=", true, _  
"SnapTargetElecConnection_on:=", true, _  
"SnapTargetIntersection_on:=", true, _  
"SnapTargetGrid_on:=", false, _  
"SnapSourceVertex_on:=", false, _  
"SnapSourceEdgeCenter_on:=", false, _  
"SnapSourceObjCenter_on:=", true, _  
"SnapSourceEdge_on:=", true, _  
"SnapSourceElecConnection_on:=", true, _  
"ConstrainToGrid:=", false _  
"defaultholesize:=", "5mil", _  
"show connection points:=", true, _  
"display vertex labels:=", true, _  
"NetColor:=", 8421440, _
```

```
"rectangle description:=", 1, "snaptoport:=", false, _
"sym footprint scaling:=", _
0.385, "primary selection color:=", 32768, _
"secondary selection color:=", _
22784, "preview selection:=", true, "anglesnap:=", _
"59deg", "AllowDragOnFirstClick:=", true, _
"useFixedDrawingResolution:=", true, _
"DrawingResolution:=", "0.002mm", "Tol:=", _
Array(3E-009, 1.5E-008, 1E-012))
```

Note:

An error will be returned if this script command is not used at the top-level hierarchy.

- Global layout defaults are stored in the registry (Layout\Preferences\)
- Names of registry items were selected for backwards compatibility and are consistent with adsn, technology file, and scripting naming
- Local options are both script-able and undo-able
- Global options are neither script-able nor undo-able

Option	Description	Installed default	Registry, Scripting Names, Data Type
Arc Drawing	Controls drawing of segments for	Dynamic	"useFixedDrawingResolution" ¹

Resolution	arcs - either dynamic and based on current zoom or fixed to specified value		"DrawingResolution" ²
Major and Minor Grid lines	Spacing, color, and visibility	The grid is displayed. Spacing based on current default length units. Major grid lines are 10 minor grid lines apart. The line colors are light blue grays, with major grid lines being darker.	"MajorColor" ³ "MinorColor" ³ "MajorSize" ² "MinorSize" ² "ShowGrid" ¹
Drawing Extent	Specifies size and coordinates of the layout	Lower left of the layout is (-0.1, -0.1) and the upper right is (0.1, 0.1)	"PageExtent" ⁴
Background color	Color of the layout background	white	"background color" ³
Sketch Mode	Fill patterns and center lines are not drawn.	off	"DefaultToSketchMode" ¹
Solid Mode	Objects are filled in with solid color.	off	"fillMode" ¹
Draw Connection Points	Display pin symbols in the layout.	off	"show connection points" ¹
Draw Rats	Display lines indicating missing physical connections in nets	on	"draw rats" ¹
Label Vertices	Display property text labeling the vertices of selected polygons and lines.	off	"display vertex labels" ¹
Net Color	Color used for highlighting nets	light yellow	"NetColor" ³
Rectangle Description	Specifies if rectangles are described with two points or cen-	2 points	"rectangle description" ⁵

	ter point, width, and height		
Default Hole Size	Default size for actual holes in a PC board	25 mil	"defaultholesize" ²
Align Microwave Components	Snaps components on entry.	on	"snaptoport" ¹
Symbol Footprint Scaling	Used to size symbol footprints placed in layout.	Based on data extent and current length units.	"sym footprint scaling" ⁶
Primary and Secondary Selection Colors	Colors used to indicate the first item selected and other items currently selected.	Primary color is red, secondary color is a darker red	"primary selectioncolor" ³ "secondary selection color" ³
Preview Selection	Highlight the object that would be selected with a left mouse button click in the current location.	off	"preview selection" ¹
Snapping options	Choose snapping sources and targets and the maximum distance (in pixels) for snapping to occur. Constrain edits to grid if desired. Note: Off-grid objects are ignored if ConstrainToGrid is asserted.	Snap distance is 20 pixels. Snapping sources are vertex, edge center, object center, & elec. conn. Snapping targets are vertex, edge center, object center, elec. conn, & grid. Edits are constrained to grid.	"PixelSnapTolerance" ⁷ "SnapTargetVertex_on" ¹ "SnapTargetEdgeCenter_on" ¹ "SnapTargetObjCenter_on" ¹ "SnapTargetEdge_on" ¹ "SnapTargetElecConnection_on" ¹ "SnapTargetIntersection_on" ¹ "SnapTargetGrid_on" ¹ "SnapSourceVertex_on" ¹ "SnapSourceEdgeCenter_on" ¹ "SnapSourceObjCenter_on" ¹ "SnapSourceEdge_on" ¹ "SnapSourceElecConnection_on" ¹ "ConstrainToGrid" ¹

Always Show Merge Layers Dialog	Controls display of the layer merging dialog.	off – Only show dialog when the user must be involved.	"AlwaysShowLayerMergeDlg" ¹
Rotation Increment	During rotation, objects are rotated by multiples of this amount.	Based on current angle units.	"anglesnap" ²
Allow Drag on first click	Selection and move with one left mouse button click.	false	"AllowDragOnFirstClick" ¹

The footnotes in the table above correspond to the following:

¹ = Integer with a value of 1 for on/true and 0 for off/false

² = String containing a amount and units

³ = Integer representing a Color

⁴ = An Array containing (lower left x, lower left y, upper right x, upper right y)

⁵ = Integer with a value of 0 for two points, and 1 for center/width/height

⁶ = String holding a double.

⁷ = Integer

CloseEditor (Footprint Editor)

Use: Closes the active definition editor.

Command: None

Syntax: CloseEditor

Return Value: None

Parameters: None

VB Example: oDefinitionEditor.CloseEditor

CreateCircle (Footprint Editor)

Use: Draws a circle in the footprint editor

Command: None

Syntax: CreateCircle

Return Value: None

VB Example:

```
oDefinitionEditor.CreateCircle Array("NAME:Contents",
"circleGeometry:=", Array("Name:=", _
"circle_118", "LayerName:=", "Top", "lw:=", "0mm", _
"x:=", "-3.23020108044147mm", "y:=", "1.49086199235171mm", _
"r:=", "0.605222899964955mm"))
```

CreateCircleVoid (Footprint Editor)

Use: Creates a circle void and adds it to a particular parent primitive. Returns the name of the newly created object.

Syntax: CreateCircleVoid <circle_void_description>

```
<circle_void_description>:
Array("NAME:Contents",
"owner:=", <object_name>, // parent primitive name
"circle voidGeometry:=", <circle_geometry>)//definition
```

VB Example:

```
oEditor.CreateCircleVoid
Array("NAME:Contents",
"owner:=", "rect_4",
"circle voidGeometry:=",
Array ("Layer:=", 6,
"Name:=", "circle void_10",
"LayerName:=", "Top",
"lw:=", "0mm",
"x:=", "26mm",
"y:=", "11mm",
"r:=", "1.41421356237309mm"))
```

CreateEdgePort (Footprint Editor)

Use: Creates an edge port using the specified edges.

Command: Draw > Port > Create

Right-click > Port > Create

Also available through Tool Bar icon

Syntax: CreateEdgePort

```
Array("NAME:Contents",
"edge:=", Array(<edge description>), "edge:=, Array(<edge description>), ...
```


"external:=", <flag>)

Return Value: Text containing the name of the created edge port. (Returns an empty name if the edge port is not created.)

Parameters: <edge description> for primitive edges

"et:=", "pe", "prim:=", <"prim">, "edge:=", <edge#>

<"prim">: text that is the primitive name

<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)

<flag>

true if the port is an external port

false if the port is an internal port

VB Example:

```
oDefinitionEditor.CreateEdgePort Array("NAME:Contents", "edge=", Array("et=",  
"pe", "prim=", "rect_6", "edge=", 1), "edge=", Array("et=", "pe", "prim=",  
"rect_6", "edge=", 2), "external=", true)
```

```
oDefinitionEditor.CreateEdgePort Array("NAME:Contents", "edge=", Array("et=",  
"pse", "sel=", "via_4", "layer=", 4, "sx=", -0.0015, "sy=", 0.0015, "ex=", -0.0015,  
"ey=", -0.0015, "h=", 0, "rad=", 0), "edge=", Array("et=", "pse", "sel=", "via_4",  
"layer=", 4, "sx=", 0.0015, "sy=", 0.0015, "ex=", -0.0015, "ey=", 0.0015, "h=", 0,  
"rad=", 0), "external=", true)
```

CreateLine (Footprint Editor)

Use: Draws a line in the footprint editor

Command: None

Syntax: CreateLine

Return Value: None

VB Example:

```
oDefinitionEditor.CreateLine Array("NAME:Contents", _  
"lineGeometry=", Array("Name=", _  
"line_160", "LayerName=", "Top", "lw=", "0.5mm", _  
"endstyle=", 0, "joinstyle=", 1, "n=", 3, _  
"x0=", "2.22709029912949mm", "y0=", "0.819053850136697mm", _  
"x1=", "-0.0527859515mm", "y1=", "0.8894385mm ",  
"x2=", "1.63943274461105mm", "y2=", "0.769682056256832mm"))
```

CreateLineVoid (Footprint Editor)

Use: Creates a line void and adds it to a specified as parameter parent primitive. Returns the name of the newly created object.

Syntax: CreateLineVoid<line_void_description>

<line_void_description>:

Array("NAME:Contents",

"owner:=", <object_name>, // parent primitive name

"line voidGeometry:=", <line_geometry>) // definition

VB Example:

```
oEditor.CreateLineVoid
Array("NAME:Contents",
"owner:=", "rect_4",
"line voidGeometry:=",
Array("Layer:=", 6,
"Name:=", "line void_14",
"LayerName:=", "Top",
"lw:=", "2mil",
"endstyle:=", 0,
"joinstyle:=", 0,
"n:=", 3,
"x0:=", "27mm", "y0:=", "5mm",
"x1:=", "35mm", "y1:=", "5mm",
"x2:=", "36mm", "y2:=", "9mm"))
```

CreateMeasure (Footprint Editor)

Use: Creates a measurement. Returns the name of the created object.

Syntax: CreateMeasure

```
Array("NAME:Contents",
```

```
"MeasurementGeometry:=",  
Array("LayerName:=", <layer_name>, // layer  
"lw:=", <value>, // line width  
"sx:=", <value>, // start X coordinate  
"sy:=", <value>, // start Y coordinate  
"ex:=", <value>, // end X coordinate  
"ey:=", <value>, // end Y coordinate  
<text_style>)
```

<text_style> :

```
"name:=", <quoted string>, // its name  
"isPlot:=", <bool>,  
"font:=", <font_name>,  
"size:=", double, // size in current units  
"angle:=", <value>,  
"weight:=", <text_weight>,  
"just:=", <text_justification>,  
"mirror:=", <bool>,  
"scales:=", <bool>))
```

VB Example:

```
oEditor.CreateMeasure  
Array("NAME:Contents",  
"MeasurementGeometry:=",  
Array("Layer:=", 0,  
"Name:=", "Measurement_2",  
"LayerName:=", "Measures",  
"lw:=", "0mm",  
"sx:=", "-32mm",  
"sy:=", "-13mm",  
"ex:=", "32mm",  
"ey:=", "-11mm",  
"name:=", "<DefaultAnnotation>",  
"isPlot:=", false,  
"font:=", "Arial",  
"size:=", 10,  
"angle:=", "0deg",  
"weight:=", 3,  
"just:=", 4,  
"mirror:=", false,  
"scales:=", false))
```

CreatePolygonVoid (Footprint Editor)

Use: Creates a polygon void and adds it to a specified as parameter parent primitive.

Syntax: CreatePolygonVoid<polygon_void_description>

<polygon_void_description>:

```
Array("NAME:Contents",  
"owner:=", <object_name>, // owner name  
"poly voidGeometry:=", <polygon_geometry>) // definition
```

Return Value: Returns the name of the newly created object.

VB Example:

```
oEditor.CreatePolygonVoid  
Array("NAME:Contents",  
"owner:=", "rect_4",  
"poly voidGeometry:=",  
Array("Layer:=", 6,  
"Name:=", "poly void_18",  
"LayerName:=", "Top",  
"lw:=", "0mm",  
"n:=", 5,  
"x0:=", "21mm", "y0:=", "9mm",  
"x1:=", "21mm", "y1:=", "5mm",
```

```
"x2:=", "24mm", "y2:=", "7mm",  
"x3:=", "24mm", "y3:=", "7mm",  
"x4:=", "24mm", "y4:=", "7mm"))
```

CreatePin (Footprint Editor)

Use: Creates a pin in the footprint editor

Command: None

Syntax: CreatePin

Return Value: None

VB Example:

```
oDefinitionEditor.CreatePin Array("NAME:Contents", _  
Array("NAME:Port", "Name:=", "Pin_123"), "Rotation:=", Array( _  
"0deg"), "Offset:=", Array("x:=", "-0.310625357087702mm", "y:=", _  
"1.4226520434022mm"), "Padstack:=", "NoPad SMT East")
```

CreatePolygon (Footprint Editor)

Use: Draws a polygon in the footprint editor

Command: None

Syntax: CreatePolygon

Return Value: None

VB Example:


```
oDefinitionEditor.CreatePolygon Array("NAME:Contents", _  
"polyGeometry:=", Array("Name:=", _  
"poly_164", "LayerName:=", "Top", "lw:=", "0mm", "n:=", 4,  
"x0:=", "2.20868387259543mm", "y0:=", "-1.80972274392843mm", _  
"x1:=", "1.435644445897937mm", "y1:=", "-2.56435642950237mm", _  
"x2:=", "3.22099728509784mm", "y2:=", "-2.02138815075159mm", _  
"x3:=", "2.40194355137646mm", "y3:=", "-2.07660533487797mm"))
```

CreateRectangle (Footprint Editor)

Use: Draws a rectangle in the footprint editor

Command: None

Syntax: CreateRectangle

Return Value: None

VB Example:

```
oDefinitionEditor.CreateRectangle Array("NAME:Contents", _  
"rectGeometry:=", Array("Name:=", "rect_120", _  
"LayerName:=", "Top", "lw:=", "0mm", "Ax:=", _  
"-3.28541826456785mm", "Ay:=", _  
"0.37731695920229mm", "Bx:=", "-2.42035021074116mm", "By:=", _  
"-0.294491270324215mm", "ang:=", "0deg"))
```

CreateText (Footprint Editor)

Use: Draws text in the footprint editor

Command: None

Syntax: CreateText

Return Value: None

VB Example:

```
oDefinitionEditor.CreateText Array("NAME:Contents", _  
"textGeometry:=", Array("Name:=", _  
"text_165", "LayerName:=", "Top", _  
"x:=", "-2.25469819270074mm", "y:=", "-2.09501106292009mm", _  
"ang:=", "0deg", "isPlot:=", true, "font:=", _  
"RomanSimplex", "size:=", "5mm", "weight:=", 3, "just:=", 4, _  
"mirror:=", false, "text:=", "My text"))
```

CreateVia (Footprint Editor)

Use: Creates a via in the footprint editor

Command: None

Syntax: CreateVia

Return Value: Name of the created via.

VB Example:

```
oDefinitionEditor.CreateVia Array("NAME:Contents", "name:=", _
```

```
"", "vposition:=", _  
Array("x:=", "2.28299549780786mm", _  
"y:=", "1.18763139471412mm"), "rotation:=", 0, "overrides hole:=", _  
false, "hole diameter:=", Array("0.50038mm"), _  
"ReferencedPadstack:=", "Template 1mm/0.5mm", _  
"highest_layer:=", "Top", "lowest_layer:=", "Top")
```

Duplicate (Footprint Editor)

Use: Duplicates footprint objects.

Command: The specified objects are duplicated by the given count with the specified offset.

Syntax: Duplicate Array("NAME:options", "count:=", <count data>), Array("NAME:elements", <element names>), Array(<x>, <y>)

Return Value: None

Parameters: <count data> is an integer

 <element names> is one or more strings with the names of footprint objects

 <x> is the x value for the offset

 <y> is the y value for the offset

VB Example:

```
oEditor.Duplicate Array("NAME:options", "count:=", 2), Array("NAME:elements", "rect_56"), Array(  
-0.018, 0.017)
```

Edit (Footprint Editor)

Use: Edits an object in the footprint editor

Command: None

Syntax: Edit

Return Value: None

VB Example:

```
oDefinitionEditor.Edit Array("NAME:items", Array("NAME:item", _  
"name:=", "poly118", Array("NAME:contents", _  
"polyGeometry:=", Array("Name:=", _  
"poly118", "LayerName:=", "Top", "lw:=", "0mm", "n:=", 5, "x0:=", _  
"-9.58323911802614mil", "y0:=", "d + 0.000193052692338824", _  
"x1:=", "-9.58323911802614mil", "y1:=", _  
"W+d + 0.000193052692338824", "x2:=", "W+d", "y2:=", _  
"0mil", "x3:=", "d", "y3:=", "0mil", "x4:=", "d", "y4:=", "d"))))
```

EraseMeasurements (Footprint Editor)

Use: Causes erasing of ALL measurements currently present.

Command: None.

Syntax: EraseMeasurements

Return Value: None.

Parameters: None.

VB Example: oEditor.EraseMeasurements

FlipHorizontal (Footprint Editor)

Use: Flips the selected object horizontally

Command: None

Syntax: FlipHorizontal

Return Value: None

VB Example:

```
oDefinitionEditor.FlipHorizontal Array("NAME:elements", "poly118"), _  
Array(0.00082250994243756, 0.00101975944723128)
```

FlipVertical (Footprint Editor)

Use: Flips the selected object vertically

Command: None

Syntax: FlipVertical

Return Value: None

VB Example:

```
oDefinitionEditor.FlipVertical Array("NAME:elements", "poly118"), _  
Array(0.00082250994243756, 0.00101975944723128)
```

GetAllLayerNames (Footprint Editor)

Use: Informational.

Command: None.

Syntax: GetAllLayerNames

Return Value: Array of strings which are the names of all layers in the layout, blackbox, or footprint.

Parameters: None.

GetLayerInfo (Footprint Editor)

Retrieves information on a specified layer in the Footprint editor.

UI Access	None		
Parameters	Name	Type	Description
	<LayerName>	String	Name of the layer
Return Value	An array of strings, as follows: • Type: typename • TopBottomAssociation: "Top" "Neither" "Bottom" "Template" "Invalid" • Color: integer [representing rgb in hex] • IsVisible: "true" "false" • IsVisible: "true" "false". This is true if any these types of objects are visible: • IsVisibleShape: "true" "false"[for stackup layer only] • IsVisiblePath: "true" "false"[for stackup layer only] • IsVisiblePad: "true" "false"[for stackup layer only] • IsVisibleHole: "true" "false"[for stackup layer only] • IsVisibleComponent: "true" "false"[for stackup layer only]		

- IsLocked: "true"|"false"
- LayerId: integer [the ID for the layer]
- IsLocked: "true"|"false"
- LayerId: integer [the ID for the layer]

The following are also in the array if the layer is a stackup layer:

- Index: integer [the stackup order index]
- LayerThickness: double [total layer thickness, in meters]
- EtchFactor: double [This will not appear if the layer has no etch factor defined.]
- IsIgnored: "true"|"false"
- NumberOfSublayers: 1 [This is always 1.]
- Material0: material name
- FillMaterial0: material name
- Thickness0: expression with units
- LowerElevation0: expression with units
- If roughness is defined, some or all of the following appear:
 - Roughness0Type: Groiss | Huray
 - Roughness0: value and units or surface ratio value for Huray
 - BottomRoughness0 Type: Groiss | Huray
 - BottomRoughness0: value and units or surface ratio value for Huray
 - SideRoughness0 Type: Groiss | Huray
 - SideRoughness0: value and units or surface ratio value for Huray

For example:

```
['Type: signal', 'TopBottomAssociation: Neither', 'Color: 32512d', 'IsVisible: true', ' IsVisibleShape: true', ' IsVisiblePath: true', ' IsVisiblePad: true', ' ]
```

	<pre>IsVisibleHole: true', ' IsVisibleComponent: true', 'IsLocked: false', 'LayerId: 7', 'Index: 0', 'LayerThickness: 0', 'IsIgnored: false', 'NumberOfSublayers: 1', 'Material0: copper', 'FillMaterial0: FR4_epoxy', 'Thickness0: 0mil', 'Lower- Elevation0: 62mil', 'Roughness0 Type: Huray', 'Roughness0: 0mil, 2.9', 'Bot- tomRoughness0 Type: Huray', 'BottomRoughness0: 0mil, 2.9', 'SideRoughness0 Type: Huray', 'SideRoughness0: 0mil, 2.9']</pre>

Python Syntax	<code>GetLayerInfo(<layer_name>)</code>
Python Example	<pre>oModule = oDesign.GetModule("Footprint") oModule.GetLayerInfo ("TopLayer")</pre>

VB Syntax	<code>GetLayerInfo <layer_name></code>
VB Example	<pre>Set oModule = oDesign.GetModule("Footprint") oModule.GetLayerInfo "TopLayer"</pre>

GetProperties (Footprint Editor)

Use: 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")
```

GetStackupLayerNames (Footprint Editor)

Use: Informational.

Command: None.

Syntax: GetStackupLayerNames

Return Value: array of strings which are the names of all layers in the layout, blackbox, or footprint.

Parameters: None.

Intersect (Footprint Editor)

Use: Causes Boolean intersecting of 2 or more *primitive* (polygons, rectangles, lines, or circles) objects.

Syntax: Intersect Array ("NAME: *primitives*",
 <object_name>, // 1st primitive name
 <object_name>, // 2nd primitive, if any
 ...) // etc

VB Example:

```
oEditor.Intersect Array("NAME:primitives", "circle_0", "rect_2")
```

Move (Footprint Editor)

Use: Moves an object in the footprint editor

Command: None

Syntax: Move

Return Value: None

VB Example:

```
oDefinitionEditor.Move Array("NAME:elements", "poly118"), _  
Array(-0.000352530973032117, -0.000276988605037332)
```

PageSetup (Footprint Editor)

Use: Specifies page setup for printing.

Command: File>Page Setup

Syntax: PageSetup <ArgArray>

Return Value: None.

Parameters: <Margins>: Page margins in implicit units of inches.

 <Border>: Integer value indicating to draw border (1) or not to draw (0).

 <DesignVars>: Integer value indicating to draw design vars (1) or not to draw (0).

VB Example:

```
Set oProject = oDesktop.GetActiveProject()  
Set oDefinitionEditor = oProject.SetActiveDefinitionEditor("FootprintEditor", "Footprint")  
oDefinitionEditor.PageSetup Array("NAME:PageSetupData", "margins:=", Array("left:=", _  
500, "right:=", 800, "top:=", 500, "bottom:=", 500), "border:=", 1, "DesignVars:=", _  
1)
```

RemoveLayer (Footprint Editor)

Use: Removes a layer or stackup layer.

Command: Remove Layer from a layout or footprint definition

Syntax: RemoveLayer (<LayerName>)

Return Value: None.

Parameters: <LayerName>

Type: <String>

VB Example:

```
oEditor.RemoveLayer ("T3 C1 sub")  
oDefinitionEditor.RemoveLayer("Top3 Footprint")
```

Note As with other Layout scripting interface commands that modify the layout, this command is not intended for use within scripts that define footprints. The command behavior from within such a script is undefined and may be unexpected. Use the LayoutHost scripting interface commands within scripts that define footprints.

RemovePort (Footprint Editor)

Use: Selected pins are changed to vias. Selected edge 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 [Footprint Editor]

Use: Rotates the selected object

Command: None

Syntax: Rotate

Return Value: None

VB Example:

```
oDefinitionEditor.Rotate Array("NAME:elements", "poly118"),  
Array( 0.000469978969405443, 0.000742770842193945)
```

Save (Footprint Editor)

Use: Saves changes during editing of symbols and footprints

Command: None

Syntax: Save

Return Value: None

Parameters: None

VB Example:

```
oDefinitionEditor.Save
```

SetActiveDefinitionEditor (Footprint Editor)

Use: Selects the symbol or footprint editor.

Command: None

Syntax: SetActiveDefinitionEditor < EditorName >,<Name>

Return Value: None

Parameters: <EditorName>

Type: <string>

Possible Values: SymbolEditor; FootprintEditor

<Name>

Type: <string>

Name of symbol or footprint to be modified

VB Example:

```
oProject.SetActiveDefinitionEditor("SymbolEditor", "tqtrx_cap")
```

SetPropertyValue (Footprint Editor)

Use: Sets the value of one property. This is not supported for properties of the following types: **ButtonProp**, **PointProp**, and **VPointProp**. Only the **ChangeProperty** command can be used to modify these properties. This can be executed by the **oProject**, **oDesign**, or **oEditor** objects.

Command: None

Syntax: SetPropertyValue <PropTab>, <PropServer>, <PropName>, <PropValue>

Return Value: None

Parameters: <PropValue>

Type: String

Contains the value to set the property. The formatting is different depending on what type of property is being edited. Use

GetPropertyValue for the desired property to see the expected format.

```
VB Example: oEditor.SetPropertyValue _  
"BaseElementTab", "rect_1", _  
"LineWidth", "3mm"
```

Subtract (Footprint Editor)

Use: Causes boolean subtracting of one or more *primitive* (polygons, rectangles, lines, or circles) object(s) from another one.

```
Subtract Array ("NAME: primitives",  
<object_name>, // Primitive to subtract from  
<object_name>, // 1st primitive to subtract, if any  
...) // etc
```

VB Example:

```
oEditor.Intersect Subtract ("NAME:primitives", "circle_0", "rect_2")
```

ToggleViaPin (Footprint Editor)

Use: Selected pins are changed to vias. Selected vias are changed to pins.

Syntax: ToggleViaPin <NAME:elements", <object_name> ... // objects to be toggled

Return Value: None.

Parameters: <object_name>

Type: <String>

VB Example: oEditor.ToggleViaPin Array("NAME:elements", "via_195")

Unite (Footprint Editor)

Use: Causes Boolean uniting of 2 or more *primitive* (polygons, rectangles, lines, or circles) objects.

Syntax: Unite Array ("NAME: *primitives*",
 <object_name>, // 1st primitive name
 <object_name>, // 2nd primitive, if any
 ...) // etc

VB Example:

```
oEditor.Unite Array("NAME:primitives", "circle_0", "rect_2")
```

ZoomToFit (Footprint Editor)

Use: Set the current zoom to fit the contents of the currently visible page

Command: None

Syntax: ZoomToFit()

Return Value: None

This page intentionally
left blank.

28 - Design Verification Script Commands

The Design Verification (DV) module controls the use of DV rule sets and runs in a Circuit project. The DV module is accessed via the design script object.

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

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

The topics for this section include:

[AddRuleSet](#)

[AddRun](#)

[DeleteRuleSet](#)

[DeleteRun](#)

[EditRuleSet](#)

[EditRun](#)

[RenameRuleSet](#)

[RenameRun](#)

[RunAllDV](#)

[RunAllRuleSetDV](#)

[RunDV](#)

AddRuleSet

Use: Adds a rule set to the design

Command: Right-click **Design Verification** in the **Project Tree** and choose **Add Rule Set**

Syntax: AddRuleSet Array("NAME:<RuleSetName>",

```
"ScriptNames:=", <ScriptInfo>,
"ScriptActiveFlags:=", <ScriptFlags>)
```

Return Value: None

Parameters: <RuleSetName>:

```
<string> // name of the rule set to create
<ScriptInfo>:
Array(<string>, <string>,...) // names of scripts to be in the rule set
<ScriptFlags>:
sequence of "t" and "f" characters
t indicates a script that is active (used in when the rule set is run)
f indicates a script is not currently active (used when the rule set is run)
applied to the scripts as ordered in <ScriptInfo>
<string>
```

VB Example:

```
oModule.AddRuleSet Array("NAME:Rule Set 9", _
"ScriptNames:=", Array("And", _
"Find Shorts"), _
"ScriptActiveFlags:=", "tf")
```

AddRun

Use: Adds a run to a rule set already in the design

Command: Right-click on a rule set item under **Design Verification** in the **Project Tree** and choose **Add Run**

Syntax: AddRun <RuleSetName>,
 Array("NAME:<RunName>",
 "TargetType:=", <TargetTypeInfo>, // optional
 "TargetObjects:=", <TargetObjectsInfo>, // optional
 "IgnoredObjects:=", <IgnoredObjectsInfo>, // optional
 "ArcTolerance:=", <ToleranceString>) // optional

Return Value: None

Parameters: <RuleSetName>:
 <string> // name of an existing rule set
 <RunName>:
 <string> // name of the run to create
 <TargetTypeInfo>:
 <int> // 0 for entire layout (default if not specified)
 // 1 for specified portion of layout
 <TargetObjectsInfo>:
 objects on which to perform the check
 may specify if TargetTypeInfo is 1 (specified portion of layout)
 "<ObjectName>, <ObjectName>, ..."
 <ObjectName>:
 <string> // name of a layout object

<IgnoredObjectsInfo>:

objects which to ignore when performing the check

may specify if TargetTypeInfo is 0 (entire layout)

"<ObjectName>, <ObjectName>, ..."

<ToleranceString>:

<string> // real number and units

// if not specified, the current layout arc tolerance is used

VB Example:

```
oModule.AddRun "Rule Set 10", _  
Array("NAME:All", _  
"TargetType:=", 0)  
oModule.AddRun "Rule Set 10", _  
Array("NAME:Selected Objs2", _  
"TargetType:=", 1, _  
"TargetObjects:=", "line_975,line_1015")  
oModule.AddRun "Rule Set 10", _  
Array("NAME:Objects to Ignore2", _  
"TargetType:=", 0, _  
"IgnoredObjects:=", "via_209,line_736")
```

DeleteRuleSet

Use: Deletes a rule set from the design

Command: Right-click on a rule set item under **Design Verification** in the **Project Tree** and choose **Delete**

Syntax: DeleteRuleSet <RuleSetName>

Return Value: None

Parameters: <RuleSetName>:

<string> // name of the rule set to delete

VB Example: oModule.DeleteRuleSet "Rule Set 9"

DeleteRun

Use: Deletes a run from an existing a rule set

Command: Right-click on a run item under **Design Verification** in the **Project Tree** and choose **Delete**

Syntax: DeleteRun <RuleSetName>, <RunName>

Return Value: None

Parameters: <RuleSetName>:

<string> // name of an existing rule set

<RunName>:

<string> // name of the run to delete

VB Example: oModule.DeleteRun "Rule Set 10", "All

EditRuleSet

Use: Edits a existing rule set

Command: Right-click on a rule set item under **Design Verification** in the **Project Tree** and choose **Properties**.

Syntax: EditRuleSet <ExistingRuleSetName>,
 Array("NAME:<RuleSetName>",
 "ScriptNames:=", <ScriptInfo>,
 "ScriptActiveFlags:=", <ScriptFlags>)

Return Value: None

Parameters: <ExistingRuleSetName>:
 <string> // name of the rule set to change
 <RuleSetName>:
 <string> // name of the rule set after change
 <ScriptInfo>:
 Array(<string>, <string>,...) // names of scripts to be in the rule set
 <ScriptFlags>:
 sequence of "t" and "f" characters
 t indicates a script that is active (used in when the rule set is run)
 f indicates a script is not currently active (used when the rule set is run)
 applied to the scripts as ordered in <ScriptInfo>
 <string>

VB Example: oModule.EditRuleSet "Rule Set 9", _
 Array("NAME:Rule Set 10", _

```
"ScriptNames:=", Array("And", _  
"Find Shorts"), _  
"ScriptActiveFlags:=", "tt")
```

EditRun

Use: Edits an existing run.

Command: Right-click on a run item under **Design Verification** in the **Project Tree** and choose **Properties**.

Syntax: EditRun <RuleSetName>,
 <ExistingRunName>,
 Array("NAME:<RunName>",
 "TargetType:=", <TargetTypeInfo>, // optional
 "TargetObjects:=", <TargetObjectsInfo>, // optional
 "IgnoredObjects:=", <IgnoredObjectsInfo>, // optional
 "ArcTolerance:=", <ToleranceString>) // optional

Return Value: None

Parameters: <RuleSetName>:
 <string> // name of an existing rule set
 <ExistingRunName>:
 <string> // name of the run to change
 <RunName>:
 <string> // name of the run after the changes
 <TargetTypeInfo>:

<int> // 0 for entire layout (default if not specified)
 // 1 for specified portion of layout
 <TargetObjectsInfo>:
 objects on which to perform the check
 may specify if TargetTypeInfo is 1 (specified portion of layout)
 "<ObjectName>, <ObjectName>, ..."
 <ObjectName>:
 <string> // name of a layout object
 <IgnoredObjectsInfo>:
 objects which to ignore when performing the check
 may specify if TargetTypeInfo is 0 (entire layout)
 "<ObjectName>, <ObjectName>, ..."
 <ToleranceString>:
 <string> // real number and units
 // if not specified, the current layout arc tolerance is used

VB Example:

```
oModule.EditRun "Rule Set 10", _
    "All", _
    Array("NAME:New All", _
    "TargetType:=", 0)
```

```
oModule.EditRun "Rule Set 10", _  
"New All", _  
Array("NAME:Selected Objs2", _  
"TargetType:=", 1, _  
"TargetObjects:=", "line_975,line_1015")  
oModule.EditRun "Rule Set 10", _  
"Selected Objs2", _  
Array("NAME:Objects to Ignore2", _  
"TargetType:=", 0, _  
"IgnoredObjects:=", "via_209,line_736")
```

RenameRuleSet

Use: Renames a existing rule set

Command: Right-click on a rule set item under **Design Verification** in the **Project Tree** and choose **Rename**

Syntax: RenameRuleSet <ExistingRuleSetName>, <NewRuleSetName>

Return Value: None

Parameters: <ExistingRuleSetName>:

 <string> // name of the rule set to change

 <NewRuleSetName>:

 <string> // new name for the rule set

VB Example: oModule.RenameRuleSet "Rule Set 12", "Rule Set 30"

RenameRun

Use: Renames an existing run.

Command: Right-click on a run item under **Design Verification** in the **Project Tree** and choose **Rename**

Syntax: RenameRun <RuleSetName>, <ExistingRunName>, <NewRunName>

Return Value: None

Parameters: <RuleSetName>:

<string> // name of an existing rule set

<ExistingRunName>:

<string> // name of the run to change

<NewRunName>:

<string> // new name for the run

VB Example: oModule.RenameRun "Rule Set 30", "New All", "Sel Objects"

RunAllDV

Use: Executes all runs in the Design Verification Project Tree item.

Command: Right-click on **Design Verification** in the **Project Tree** and choose **Run All**

Syntax: RunAllDV

Return Value: None

Parameters: None

VB Example: oModule.RunAllDV

RunAllRuleSetDV

Use: Executes all runs in an existing rule set

Command: Right-click on a rule set item under **Design Verification** in the **Project Tree** and choose **Run All**

Syntax: RunAllRuleSetDV <RuleSetName>

Return Value: None

Parameters: <RuleSetName>:

<string> // name of an existing rule set

VB Example: oModule.RunAllRuleSetDV "Rule Set 30"

RunDV

Use: Executes the specified run in an existing rule set

Command: Right-click on a run item under **Design Verification** in the **Project Tree** and choose **Run**

Syntax: RunDV <RuleSetName>, <RunName>

Return Value: None

Parameters: <RuleSetName>:

<string> // name of an existing rule set

<RunName>:

<string> // name of the run to execute

VB Example: oModule.RunDV "Rule Set 30", "Run 2"

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.

UI Access	N/A		
Parameters	Name	Type	Description
	<message>	String	Error message.
Return Value	None.		

Python Syntax	AddErrorMessage(<message>)
Python Example	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.AddErrorMessage('My error message.')</pre>

VB Syntax	N/A
VB Example	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.

UI Access	N/A		
Parameters	Name	Type	Description
	<message>	String	Error message.
Return Value	None.		

Python Syntax	AddFatalMessage(<message>)
Python Example	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.AddFatalMessage('My fatal error message.')</pre>

VB Syntax	N/A
VB Example	N/A. Script cannot be run in VBScript.

AddInfoMessage

Adds an informational message to the **Message Manager** window. AddInfoMessage is a function of CoreGlobalScriptContextFunctions.

UI Access	N/A		
Parameters	Name	Type	Description
	<message>	String	Informational message.
Return Value	None.		

Python Syntax	AddInfoMessage(<message>)
Python Example	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.AddInfoMessage('My info.')</pre>

VB Syntax	N/A
VB Example	N/A. Script cannot be run in VBScript.

AddWarningMessage

Adds a warning message to the **Message Manager** window. AddWarningMessage is a function of CoreGlobalScriptContextFunctions.

UI Access	N/A		
Parameters	Name	Type	Description
	<message>	String	Warning message.
Return Value	None.		

Python Syntax	AddWarningMessge(<message>)
Python Example	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.AddWarningMessage('My warn- ing.')</pre>

VB Syntax	N/A
VB Example	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.

UI Access	N/A		
Parameters	Name	Type	Description
	<message>	String	Debug message.
Return Value	None.		

Python Syntax	LogDebug(<message>)

Python Example	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.LogDebug('My debug message.')</pre>

VB Syntax	N/A
VB Example	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.

UI Access	N/A		
Parameters	Name	Type	Description
	<error>	String	Error to log.
Return Value	None.		

Python Syntax	LogError(<error>)
Python Example	<pre>import CoreGlobalScriptContextFunctions CoreGlobalScriptContextFunctions.LogError('My error.')</pre>

VB Syntax	N/A
VB Example	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](#)

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" & swncfreq, "")
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
```

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")
```

GetPropNames and GetPropValues for Layered Impedance Boundary Script

The following example script used Object Oriented commands to GetPropName for a Layered Impedance Boundary, and then use the names to use GetPropValue for those properties.

```
' -----
' Script Recorded by Ansys Electronics Desktop Version 2022.1.0
' 14:55:18 Oct 25, 2022
' -----

Option Explicit

test_layered_impedance()

Sub test_layered_impedance()

    Dim oAnsoftApp
    Dim oDesktop
    Dim oProject
    Dim oDesign
```

```
Dim oEditor
Dim oModule
Dim oBoundaries
Dim Layered_Boundaries
Dim single_boundary
Dim oBoundary
Dim Get_Boundary_Properties
Dim Layer_Name
Dim Layer_Type
Dim Layer_Roughness
Dim Layer_Inf_Ground_Plane
Dim Layer_Two_Sided
Dim Layer_Shell_Element
Dim Layer_Number_of_Layers
Dim Layer_L1_Thickness
Dim Layer_L1_Material
Dim Layer_L2_Thickness
Dim Layer_L2_Material

Set oAnsoftApp = CreateObject("Ansoft.ElectronicsDesktop.2022.1")
Set oDesktop = oAnsoftApp.GetAppDesktop()
oDesktop.RestoreWindow

Set oProject = oDesktop.SetActiveProject("test1")
Set oDesign = oProject.SetActiveDesign("HFSSDesign1")
Set oModule = oDesign.GetModule("BoundarySetup")

Set oBoundaries = oDesign.GetChildObject("Boundaries")
Layered_Boundaries = oModule.GetBoundariesOfType("Layered Imped-
ance")

For Each single_boundary In Layered_Boundaries
    Set oBoundary = oBoundaries.GetChildObject(single_boundary)
```

```
'Get_Boundary_Properties = oBoundary.GetPropNames()  
Layer_Name = oBoundary.GetPropValue("Name")  
Layer_Type = oBoundary.GetPropValue("Type")  
Layer_Roughness = oBoundary.GetPropValue("Roughness")  
Layer_Inf_Ground_Plane = oBoundary.GetPropValue("Inf Ground  
Plane")  
Layer_Two_Sided = oBoundary.GetPropValue("Two sided")  
Layer_Shell_Element = oBoundary.GetPropValue("Shell Element")  
Layer_Number_of_Layers = oBoundary.GetPropValue("Number of Lay-  
ers")  
Layer_L1_Thickness = oBoundary.GetPropValue("Layer 1/Thickness")  
Layer_L1_Material = oBoundary.GetPropValue("Layer 1/Material")  
Layer_L2_Thickness = oBoundary.GetPropValue("Layer 2/Thickness")  
Layer_L2_Material = oBoundary.GetPropValue("Layer 2/Material")  
MsgBox ("Name = " & Layer_Name & vbNewLine _  
& "Type = " & Layer_Type & vbNewLine _  
& "Roughness = " & Layer_Roughness & vbNewLine _  
& "Inf Ground = " & Layer_Inf_Ground_Plane & vbNewLine _  
& "Two-Sided = " & Layer_Two_Sided & vbNewLine _  
& "Shell Element = " & Layer_Shell_Element & vbNewLine _  
& "Number of Layers = " & Layer_Number_of_Layers & vbNewLine _  
& "Layer 1 Thickness = " & Layer_L1_Thickness & vbNewLine _  
& "Layer 1 Material = " & Layer_L1_Material & vbNewLine _  
& "Layer 2 Thickness = " & Layer_L2_Thickness & vbNewLine _  
& "Layer 2 Material = " & Layer_L2_Material)
```

Next

End Sub

IronPython Example Scripts

IronPython Examples:

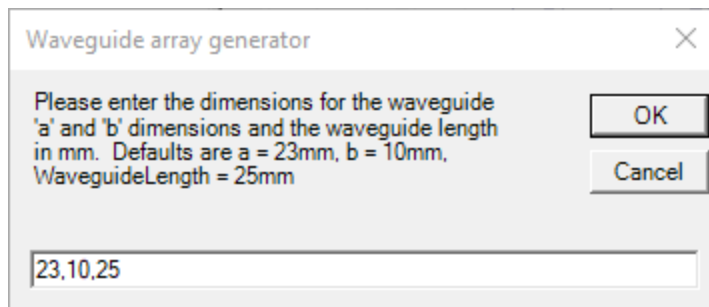
- [BH Coordinates Python Script](#)
- [Equation Based Curve Python Script](#)

HFSS Waveguide Array Python Script

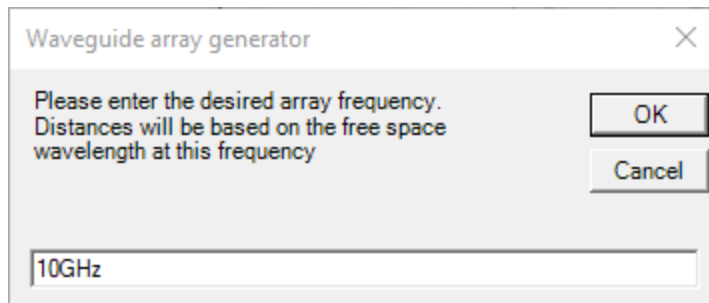
This sample Python script creates an HFSS Waveguide array. The script includes comment lines, which are preceded by an apostrophe ('), that offer explanations for each subsequent line or lines. The script includes examples of creating a 3D model, boundaries and excitation, solution setup and sweep, as well as reports.

You must insert an HFSS project before running the script. Running the script causes a series of input dialogs to appear that lets you set parameters.

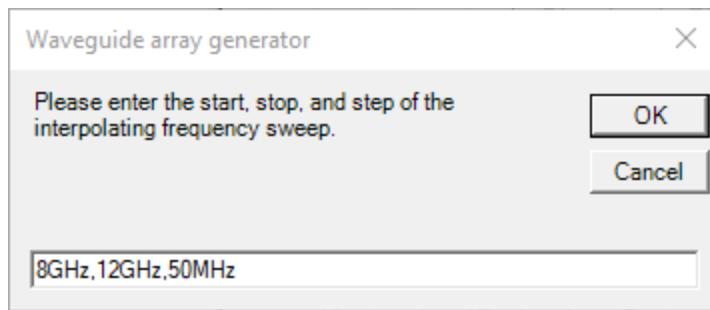
The first dialog asks for input for a and b dimensions and the waveguide length.



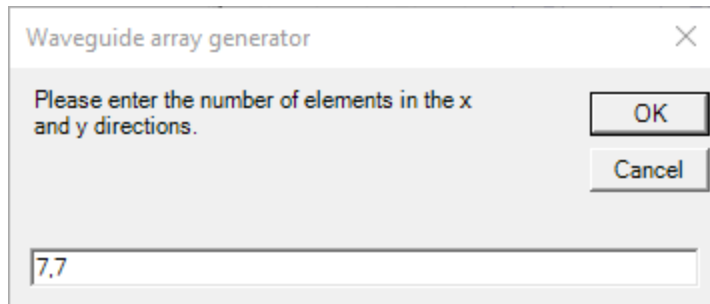
The next asks for the array frequency.



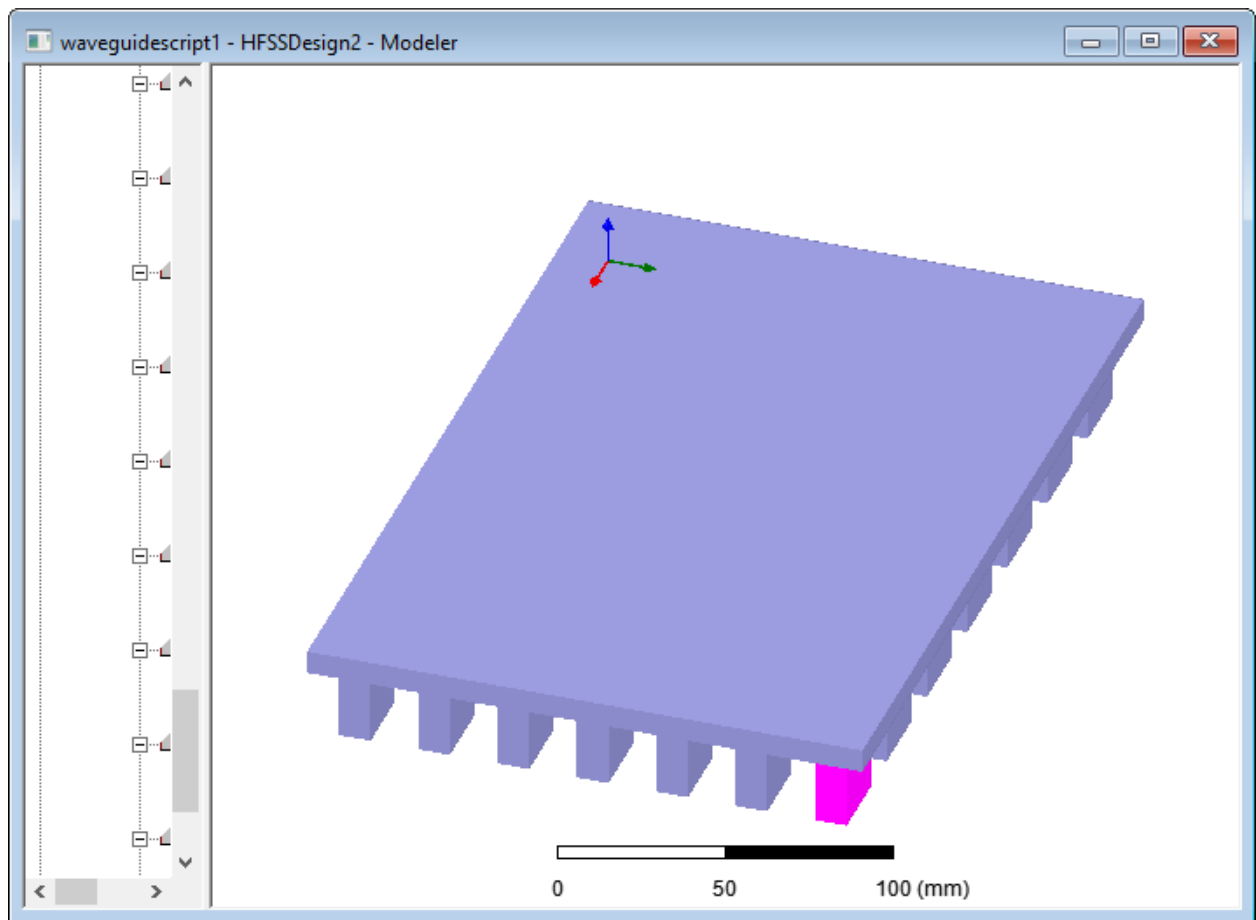
The next asks for start, stop and step values for in an interpolating frequency sweep.



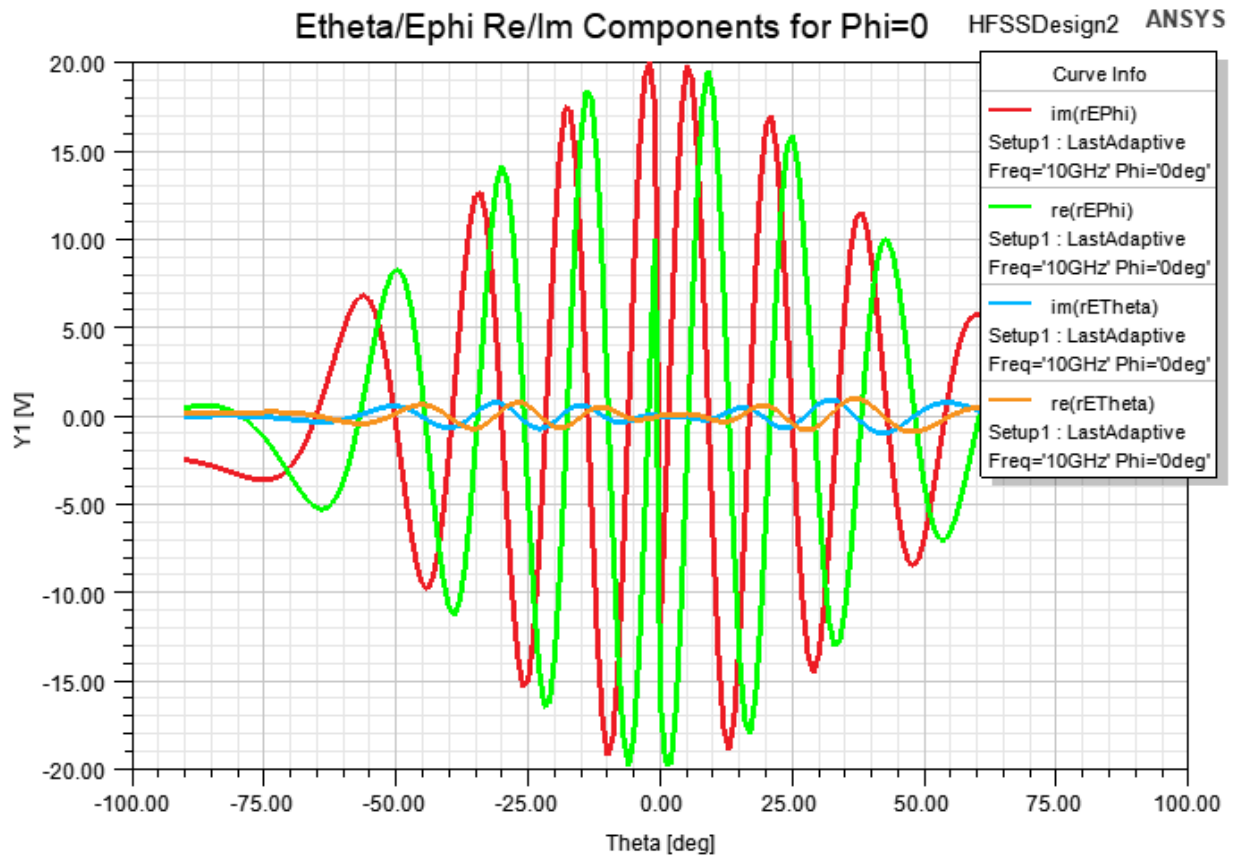
The last one asks for the array definition.

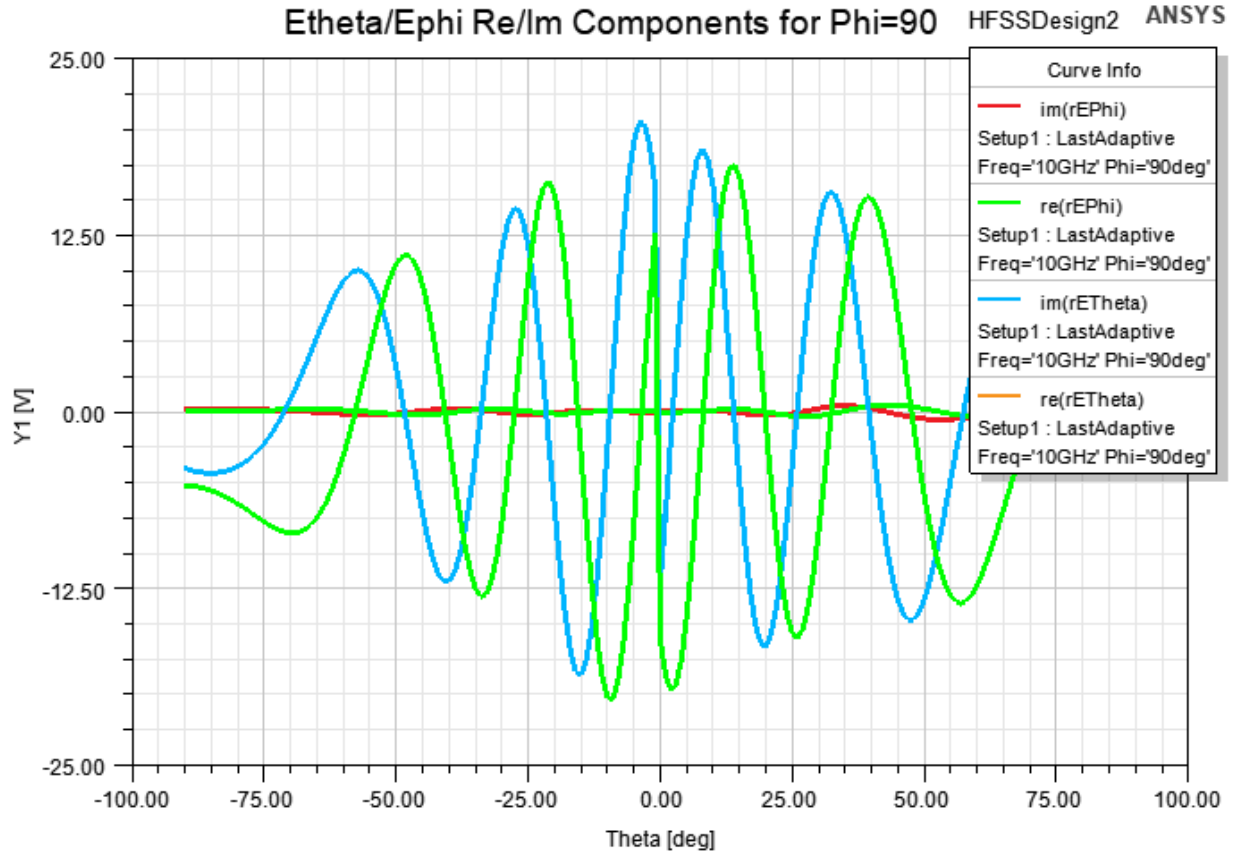


The following figure shows the waveguide array generated by the defaults.



After running the simulation, the plots defined by the script shows the following results.





The waveguide script in Python follows.

```
import clr

clr.AddReferenceByPartialName("Microsoft.VisualBasic")

from Microsoft.VisualBasic.Constants import
vbOKOnly, vbOKCancel, vbAbortRetryIgnore, vbYesNoCancel, vbYesNo, vbRetryC-
ancel

from Microsoft.VisualBasic.Constants import
vbOK, vbCancel, vbAbort, vbRetry, vbIgnore, vbYes, vbNo

from Microsoft.VisualBasic.Interaction import InputBox, MsgBox

oProject = oDesktop.GetActiveProject()

oDesign = oProject.GetActiveDesign()

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



```
# Ask for dimensions for waveguide dimensions
# -----

dim = InputBox("Please enter the dimensions for the waveguide 'a' and
'b' dimensions "

    + "and the waveguide length in mm. Defaults are a = 23mm, b = 10mm,
    WaveguideLength = 25mm",

    "Waveguide array generator", "23,10,25")
Dimensions = dim.split(',')

a_str = Dimensions[0] + "mm"
b_str = Dimensions[1] + "mm"
b_over2 = float(Dimensions[1])/2

WaveguideLength_str = Dimensions[2] + "mm"

#Ask for the frequency of operation
#-----

Frequency = InputBox("Please enter the desired array frequency.
Distances will " +

    "be based on the free space wavelength at this frequency",

    "Waveguide array generator", "10GHz")

#Ask for the start, stop, and step of the interpolating frequency
sweep
#-----
--

inputText = InputBox("Please enter the start, stop, and step of the
interpolating " +

    "frequency sweep.", "Waveguide array generator",
    "8GHz,12GHz,50MHz")

StartFrequency, StopFrequency, StepFrequency = inputText.split(',')

```

```
#Ask for the number of elements in the x and y directions
#-----

inputText = InputBox("Please enter the number of elements in the x
and y directions.",

    "Waveguide array generator", "7,7")
numX, numY = [float(elem) for elem in inputText.split(',')]
TotalElements = numX*numY

# Make variables and set them equal to the values from the user input
# -----

oDesign.ChangeProperty(
    [
        "NAME:AllTabs",
        [
            "NAME:LocalVariableTab",
            [
                "NAME:PropServers",
                "LocalVariables"
            ],
            [
                "NAME:NewProps",
                [
                    "NAME:a", "PropType:=", "VariableProp", "UserDef:=", True,
                    "Value:=", a_str
                ],
            ],
            [
                "NAME:b", "PropType:=", "VariableProp", "UserDef:=", True,
                "Value:=", b_str
            ],
            [
```

```
"NAME:NumX", "PropType:=", "VariableProp", "UserDef:=", True,
"Value:=", numX
],
[
"NAME:NumY", "PropType:=", "VariableProp", "UserDef:=", True,
"Value:=", numY
],
[
"NAME:WaveguideLength", "PropType:=", "VariableProp", "User-
Def:=", True,
"Value:=", WaveguideLength_str
],
[
"NAME:Frequency", "PropType:=", "VariableProp", "UserDef:=",
True,
"Value:=", Frequency
],
[
"NAME:Lambda", "PropType:=", "VariableProp", "UserDef:=", True,
"Value:=", "c0/" + Frequency
],
[
"NAME:RadBoundDist", "PropType:=", "VariableProp", "UserDef:=",
True,
"Value:=", "Lambda/4"
]
]
]
])
#Create the radiation box
#-----
```

```

oEditor.CreateBox(
[
    "NAME:BoxParameters",
    "XPosition:=" , "-a/2-RadBoundDist",
    "YPosition:=" , "-b/2-RadBoundDist",
    "ZPosition:=" , "0mm",
    "XSize:=" , "NumX*a+(NumX-1)*Lambda/2+2*RadBoundDist",
    "YSize:=" , "NumY*b+(NumY-1)*Lambda/2+2*RadBoundDist",
    "ZSize:=" , "RadBoundDist"
],
[
    "NAME:Attributes",
    "Name:=" , "RadiationBox",
    "Flags:=" , "",
    "Color:=" , "(132 132 193)",
    "Transparency:=" , 0.8,
    "PartCoordinateSystem:=" , "Global",
    "UDMId:=" , "",
    "MaterialValue:=" , "\"vacuum\"",
    "SurfaceMaterialValue:=" , "\"\"",
    "SolveInside:=" , True,
    "IsMaterialEditable:=" , True,
    "UseMaterialAppearance:=" , False,
    "IsLightweight:=" , False
])

```

```

oEditor.FitAll() # Zoom out
#Create first element
#-----
oEditor.CreateBox(

```

```
[
    "NAME:BoxParameters",
    "XPosition:=" , "-a/2",
    "YPosition:=" , "-b/2",
    "ZPosition:=" , "0mm",
    "XSize:=" , "a",
    "YSize:=" , "b",
    "ZSize:=" , "-WaveguideLength"
],
[
    "NAME:Attributes",
    "Name:=" , "Element1",
    "Flags:=" , "",
    "Color:=" , "(132 132 193)",
    "Transparency:=" , 0.8,
    "PartCoordinateSystem:=" , "Global",
    "UDMId:=" , "",
    "MaterialValue:=" , "\"vacuum\"",
    "SurfaceMaterialValue:=" , "\"\"",
    "SolveInside:=" , True,
    "IsMaterialEditable:=" , True,
    "UseMaterialAppearance:=" , False,
    "IsLightweight:=" , False
])

# Define the port
# -----
# Get the numeric value of half of the b dimension of the port.
# The values fed in to the "Start" and "End" arrays cannot have mathematical operators. For example, if HFSS
```

has a variable 'b', and a desired coordinate is 'b/2', that value cannot be entered here as "b/2". The number

has to be computed explicitly in script and entered as a string such as "-200mm".

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

```
# get bottom face ID
```

```
element1FaceID = oEditor.GetFaceByPosition(
```

```
[
    "NAME:Parameters",
    "BodyName:=", "Element1",
    "XPosition:=", "-a/2",
    "YPosition:=", "-b/2",
    "ZPosition:=", "-WaveguideLength"
])
```

```
oModule.AssignWavePort(
```

```
[
    "NAME:WavePort1",
    "Faces:=" , [element1FaceID],
    "NumModes:=" , 1,
    "RenormalizeAllTerminals:=", True,
    "UseLineModeAlignment:=", False,
    "DoDeembed:=" , False,
    [
        "NAME:Modes",
        [
            "NAME:Model",
            "ModeNum:=" , 1,
            "UseIntLine:=" , True,
            [
                "NAME:IntLine",
```

```
"Start:=" , ["0mm", "-" +str(b_over2) + "mm", "-" + Wave-
guideLength_str],
"End:=" , ["0mm", str(b_over2) + "mm", "-" + WaveguideLength_str]
],
"AlignmentGroup:=" , 0,
"CharImp:=" , "Zpi"
]
],
"ShowReporterFilter:=" , False,
"ReporterFilter:=" , [True],
"UseAnalyticAlignment:=" , False
])

#Set the radiation boundary
#-----
# Get face IDs for further assignment
top_face_id = oEditor.GetFaceByPosition(["NAME:FaceParameters",
    "BodyName:=", "RadiationBox",
    "XPosition:=", "0mm",
    "YPosition:=", "NumY*b-b/2+(NumY-1)*Lambda/2+RadBoundDist",
    "ZPosition:=", "0mm"])

faceIDs = [int(elem) for elem in oEditor.GetFaceIDs("RadiationBox")
if elem != str(top_face_id)]
oModule.AssignRadiation(
[
    "NAME:Radiation",
    "Faces:=" , faceIDs,
    "IsFssReference:=" , False,
    "IsForPML:=" , False
])
```

```
# Copy and paste elements/ports into a rectangular array.
# Duplicate boundaries with geometry" must be turned on under Tools-
>Options->HFSS Options
# -----
-----

ElementNum = 1
for i in range(1, int(numX)+1):
    for j in range(1, int(numY)+1):
        if ElementNum == 1:
            pass

elif ElementNum <= numY: #If in the first column, only
    oEditor.Copy(["NAME:Selections", "Selections:=", "Element1"])
    oEditor.Paste()
    oEditor.Move(["NAME:Selections", "Selections:=", "Element" + str
(ElementNum)],
["NAME:TranslateParameters", "CoordinateSystemID:=", -1,
"TranslateVectorX:=", "0mm",
"TranslateVectorY:=", str(j-1) + "*" + (b+Lambda/2)",
"TranslateVectorZ:=", "0mm"]])

elif ElementNum > numY:
    oEditor.Copy(["NAME:Selections", "Selections:=", "Element1"])
    oEditor.Paste()
    oEditor.Move(["NAME:Selections", "Selections:=", "Element" + str
(ElementNum)],
["NAME:TranslateParameters", "CoordinateSystemID:=", -1,
"TranslateVectorX:=", str(i-1) + "*" + (a+Lambda/2)",
"TranslateVectorY:=", str(j-1) + "*" + (b+Lambda/2)",
"TranslateVectorZ:=", "0mm"]])
```



```
ElementNum += 1

#Create the setup and interpolating sweep
#-----
oModule = oDesign.GetModule("AnalysisSetup")
oModule.InsertSetup("HfssDriven",
[
    "NAME:Setup1",
    "AdaptMultipleFreqs:" , False,
    "Frequency:" , Frequency,
    "MaxDeltaS:" , 0.02,
    "PortsOnly:" , False,
    "UseMatrixConv:" , False,
    "MaximumPasses:" , 15,
    "MinimumPasses:" , 1,
    "MinimumConvergedPasses:" , 1,
    "PercentRefinement:" , 50,
    "IsEnabled:" , True,
    "BasisOrder:" , 1,
    "DoLambdaRefine:" , True,
    "DoMaterialLambda:" , True,
    "SetLambdaTarget:" , False,
    "Target:" , 0.3333,
    "UseMaxTetIncrease:" , False,
    "PortAccuracy:" , 2,
    "UseABConPort:" , False,
    "SetPortMinMaxTri:" , False,
    "UseDomains:" , False,
    "UseIterativeSolver:" , False,
    "SaveRadFieldsOnly:" , False,
```

```
"SaveAnyFields:=" , True,
"IESolverType:=" , "Auto",
"LambdaTargetForIESolver:=", 0.15,
"UseDefaultLambdaTgtForIESolver:=", True
])

oModule.InsertFrequencySweep("Setup1",
[
  "NAME:InterpolatingSweep",
  "IsEnabled:=" , True,
  "RangeType:=" , "LinearStep",
  "RangeStart:=" , StartFrequency,
  "RangeEnd:=" , StopFrequency,
  "RangeStep:=" , StepFrequency,
  "Type:=" , "Interpolating",
  "SaveFields:=" , False,
  "InterpTolerance:=" , 0.5,
  "InterpMaxSolns:=" , 50,
  "InterpMinSolns:=" , 0,
  "InterpMinSubranges:=" , 1,
  "ExtrapToDC:=" , False,
  "InterpUseS:=" , True,
  "InterpUsePortImped:=" , False,
  "InterpUsePropConst:=" , True,
  "UseDerivativeConvergence:=", False,
  "InterpDerivTolerance:=", 0.2,
  "UseFullBasis:=" , True,
  "EnforcePassivity:=" , True,
  "PassivityErrorTolerance:=", 0.0001
])
```

```
#Create a relative coordinate system centered on the array for radiation pattern calculations

#-----

oEditor.CreateRelativeCS(

[
    "NAME:RelativeCSParameters",
    "Mode:=" , "Axis/Position",
    "OriginX:=" , "-a/2+NumX*a/2+(NumX-1)*Lambda/4",
    "OriginY:=" , "-b/2+NumY*b/2+(NumY-1)*Lambda/4",
    "OriginZ:=" , "0mm",
    "XAxisXvec:=" , "1mm",
    "XAxisYvec:=" , "0mm",
    "XAxisZvec:=" , "0mm",
    "YAxisXvec:=" , "0mm",
    "YAxisYvec:=" , "1mm",
    "YAxisZvec:=" , "0mm"
],
[
    "NAME:Attributes",
    "Name:=" , "RelativeCS1"
])

#Create an infinite sphere with fine theta resolution and phi cuts at 0 and 90 degrees

#-----

oModule = oDesign.GetModule("RadField")
oModule.InsertFarFieldSphereSetup(

[
    "NAME:Infinite Sphere1",
    "UseCustomRadiationSurface:=", False,
```

```

"ThetaStart:=" , "-90deg",
"ThetaStop:=" , "90deg",
"ThetaStep:=" , "1deg",
"PhiStart:=" , "0deg",
"PhiStop:=" , "90deg",
"PhiStep:=" , "90deg",
"UseLocalCS:=" , True,
"CoordSystem:=" , "RelativeCS1"
])

```

```

#Create output plots for Ephi/Etheta real and imaginary components
for Phi = 0 and separately for Phi = 90

```

```

#-----
-----

```

```

oModule = oDesign.GetModule("ReportSetup")

```

```

#For phi = 0

```

```

oModule.CreateReport("Etheta/Ephi Re/Im Components for Phi=0", "Far
Fields",

```

```

    "Rectangular Plot", "Setup1 : LastAdaptive",

```

```

    [

```

```

        "Context:=" , "Infinite Sphere1"

```

```

    ],

```

```

    [

```

```

        "Theta:=" , ["All"],

```

```

        "Phi:=" , ["0deg"],

```

```

        "Freq:=" , ["All"],

```

```

        "a:=" , ["Nominal"],

```

```

        "b:=" , ["Nominal"],

```

```

        "NumX:=" , ["Nominal"],

```

```

        "NumY:=" , ["Nominal"],

```

```

        "WaveguideLength:=" , ["Nominal"],

```

```
"Frequency:=" , ["Nominal"]
],
[
  "X Component:=" , "Theta",
  "Y Component:=" , ["im(rEPhi)", "re(rEPhi)", "im(rETheta)", "re
    (rETheta)"]
], [])

#For phi = 90
#-----

oModule.CreateReport("Etheta/Ephi Re/Im Components for Phi=90", "Far
Fields",
  "Rectangular Plot", "Setup1 : LastAdaptive",
  [
    "Context:=" , "Infinite Sphere1"
  ],
  [
    "Theta:=" , ["All"],
    "Phi:=" , ["90deg"],
    "Freq:=" , ["All"],
    "a:=" , ["Nominal"],
    "b:=" , ["Nominal"],
    "NumX:=" , ["Nominal"],
    "NumY:=" , ["Nominal"],
    "WaveguideLength:=" , ["Nominal"],
    "Frequency:=" , ["Nominal"]
  ],
  [
    "X Component:=" , "Theta",
    "Y Component:=" , ["im(rEPhi)", "re(rEPhi)", "im(rETheta)", "re
      (rETheta)"]
  ]
]
```

], [])

This page intentionally
left blank.

Index

2

2.5D Via Commands 23-1

3

3D Modeler editor commands 8-1

AlignFaces 8-125

AssignMaterial 8-126

AssignSurfaceMaterial 8-250

Chamfer 8-129

ChangeProperty 6-12, 8-253,
13-9, 27-24

CleanUpModel 8-131

CloseAllWindows 8-258

Connect 8-132

Copy 8-104

CoverLines 8-133

CoverSurfaces 8-134

Create3DComponent 8-6

CreateBondwire 8-11

CreateBox 8-15

CreateCircle 8-17

CreateCone 8-20

CreateCutplane 8-23

CreateCylinder 8-25

CreateEllipse 8-27

CreateEntityList 8-135

CreateEquationCurve 8-30

CreateEquationSurface 8-34

CreateFaceCS 8-136, 8-142

CreateGroup 8-141

CreateHelix 8-37

CreateObjectFromEdges 8-149

CreateObjectFromFace 8-151

CreateObjectFromFaces 8-153

CreatePoint 8-39

CreatePolyline 8-41

CreateRectangle 8-46

CreateRegion 8-49

CreateRegularPolygon 8-52

CreateRegularPolyhedron 8-55

CreateRelativeCS 8-154

CreateSphere 8-58

CreateSpiral 8-61

CreateTorus 8-63

CreateUserDefinedPart 8-65

Delete 8-258

DeleteEmptyGroups 8-156

DeleteLastOperation 8-157

DeleteOperation 8-158

DeletePolylinePoint 8-105	GetEdgeByPosition 8-273
DetachEdges 8-160	GetEdgeIDsfromFace 8-275
DetachFaces 8-161	GetEdgeIDsfromObject 8-275
DuplicateAlongLine 8-106	GetEdgeLength 8-276
DuplicateAroundAxi 8-108	GetEntityListIDByName 8-277
DuplicateMirror 8-111	GetFaceArea 8-277
EditFaceCS 8-163	GetFaceByPosition 8-279
EditObjectCS 8-169	GetFaceCenter 8-278
EditPolyline 8-78	GetFacelDs 8-280
EditRelativeCS 8-176	GetFacelDsOfSheet 8-281
Export 8-178	GetModelBoundingBox 8-282
ExportModelImagetoFile 8-180, 10-62	GetPartsForUserDefinedModel 8-292
Fillet 8-184	GetPoints 8-293
FlattenGroup 8-186	GetProperties 2-95, 6-51, 7-103
GenerateAllUserDefinedModels 8-260	GetPropertyValue 2-96, 6-52, 7-104, 8- 294, 10-86, 26-120
GenerateHistory 8-187	GetPropEvaluatedValue 6-47, 8-293
GenerateUserDefinedModel 8- 260	GetPropSIValue 6-49, 8-297
Get3DComponentParameters 8-85	GetRelativeCoordinateSystems 8-299
Get3DComponentPartNames 8- 86	GetUser Position 8-301
GetActiveCoordinateSystem 8- 188	GetVertexIDFromNameForFirstOperation 8-302
GetAct- iveCoordin- ateSystemTransform 8-189	GetVertexIDsFromEdge 8-303
GetBodyNamesByPosition 8- 263	GetVertexIDsFromFace 8-303
GetCoordinateSystems 8-189	GetVertexIDsFromObject 8-304
	GetVertexPosition 8-305
	GetWireBodyNames 8-305
	HealObject 8-190
	Import 8-194

ImportDXF 8-198, 27-125	SetTopDownViewDir- ectionForAllViews 8-313
ImportFromClipboard 8-202	SetWCS 8-224
ImportGDSII 8-203	Simplify 8-226
Imprint 8-206	Split 8-229
ImprintProjection 8-208	Stitch 8-231
Insert3DComponent 8-86	Subtract 8-233
InsertComponent 8-87	SweepAlongPath 8-91
Intersect 8-210	SweepAlongVector 8-93
Mirror 8-114	SweepAroundAxis 8-95
Move 8-116	SweepFacesAlongNormal 8-97, 8- 97, 8-234, 8-234
MoveCSToEnd 8-211	Sweep- FacesAlongNormalWithAt- tributes 8-99
MoveEntityToGroup 8-212	ThickenSheet 8-236
MoveFaces 8-213	UncoverFaces 8-238
OffsetFaces 8-118	Ungroup 8-241
PageSetup 8-306	Unite 8-239
ProjectSheet 8-215	UpdateComponentDefinition 8- 102
RemoveBadEdges 8-308	UpdatePriorityList 8-313
RemoveBadFaces 8-308	UpgradeVersion 8-314
RemoveBadVertices 8-309	Validate3DComponent 8-315
RenamePart 8-310	WrapSheet 8-242
Replacewith3DComponent 8- 217	WriteHistoryTreeLayoutForTest 8- 316
Rotate 8-120	3D Modeler editor object commands
Scale 8-121	GetNumObjects 8-283
Section 8-220	GetObjectIDByName 8-284
SeparateBody 8-222	GetObjectName 8-285
SetModelUnits 8-223	
SetPropValue Modeler 8-311	
SetTopDownViewDir- ectionForActiveView 8-312	

GetObjectByNameByEdgeID 8-285

GetObjectByNameByFaceID 8-286

GetObjectByNameByID 8-287

GetObjectByNameByVertexID 8-287

GetObjectShapeType 8-289

GetObjectVolume 8-291

GetObjPath 8-291

A

Aborting Scripts 1-11

Activate 25-55, 27-7

Activation and Deactivation Methods 25-55

Add 16-1, 16-8, 16-12, 16-18, 21-2, 22-2, 24-1, 26-3, 26-34, 26-40, 26-141, 26-215, 26-266, 26-302, 26-334

AddAddSolverOnDemandModel 26-165

AddAllEyeMeasurements 10-6

AddAntennaOverlay 15-11

AddCableToBundle 8-319

AddCartesianLimitLine 10-6

AddCartesianLimitLineFromCurve 10-8

AddCartesianLimitLineFromEquation 10-10

AddCartesianXMarker 10-12

AddCartesianYMarker 10-12

AddCartesianYMarkerToStack 10-13

AddCircuitRefPort 25-176, 27-8

AddDataset 6-4, 7-7, 26-22

AddDefinitionFromBlock 26-25

AddDefinitionFromLibFile 8-245

AddDeltaMarker 10-14

AddDesignVariablesForDynamicLink 7-10

AddDiffPair 19-4

AddDynamicLink 7-11

AddHole 25-74

Additional Property Scripting Example 2-102

AddLevel 27-150

AddMarker 10-16

AddMarkerToPlot 13-4

AddMaterial 6-7, 26-30, 26-67

AddMenuProp 2-56

AddMenuProp2 2-58

AddMessage 4-6

AddModelingProperties 7-13

AddNote 10-17

AddNPortData 26-159

AddPoint 25-71

AddPortsToAllNets 25-178, 25-178, 27-14, 27-14

AddPortsToNet 25-178, 27-14

AddProp 2-59

AddProp2 2-63

AddRadFieldSourceGroup 15-13

AddRefPort 22-2, 25-179, 27-14

AddRefPortUsingEdges 22-4
AddRuleSet 28-1
AddRun 28-3
AddSweep 21-19
AddTraceCharacteristics 10-19
AddTraces 10-20
AlignFaces 8-125
AlignHorizontal 27-151
AlignObjects 25-207, 27-17
AlignPorts 25-181, 27-18
AlignVertical 27-152
Analysis module commands
 Analyze 7-13
 AnalyzeAll 7-14
 AnalyzeAllNominal 7-15
 CopySetup 11-7, 11-35
Analyze 21-23, 21-23
 Analysis module command 7-13
AnalyzeDistributed 7-16
AnalyzeSweep 21-23
Ansoft Application Object commands 3-1
 GetAppDesktop 3-2
ApplyMeshOps 7-17, 7-81
ApplyReportTemplate 10-23
Area 25-78
AreMaterialPropertiesEqual 26-51
AreSurfaceMaterialPropertiesEqual 26-52
AreThereSimulationsRunning 4-7

arithmetic operators 1-6
array variables 1-4
AssignCircuitRefPort 25-182, 25-184, 27-19, 27-21
AssignMaterial 8-126
AssignSurfaceMaterial 8-250

B

BBoxLL 25-76
BBoxUR 25-76
Boolean Operations on Primitives 25-82
Boundary/Excitation module commands
 CreateNportCircuitElements 27-57
BreakUDMConnection 8-252
BringToFront 26-345, 27-153

C

Cable Modeling commands
 AddCableToBundle 8-319
 CreateCableBundle 8-320
 CreateCableHarness 8-322
 CreateClockSource 8-324
 CreatePWLSource 8-326
 CreateStraightWireCable 8-328, 8-329
 ExportCableLibrary 8-331
 ImportCableLibrary 8-332
 RemoveCables 8-332
 UpdateCableHarness 8-333

- Cable Modeling Commands 8-318
- CalcOp 13-6, 14-3
- CalcRead (deprecated) 14-4
- CalcStack 13-7, 14-6
- CalculatorRead 13-7, 14-5
- CalculatorWrite 13-8, 14-6
- Cascade 19-5
- Cavity Commands 24-1
- Chamfer 8-129
- ChangeGeomSettings 13-14, 14-7
- ChangeOptions 25-208, 27-216
- ChangeProperty 6-12, 7-18, 8-253, 10-23, 13-9, 26-53, 27-24, 27-153
- CircleIntersectsPolygon 25-79
- ClcEval 13-14, 14-8
- ClcMaterial 13-15, 14-9
- ClcMaterialValue 14-10
- CleanUpModel 8-131
- ClearAllMarkers 10-33
- ClearAllNamedExpr 13-16, 14-10
- ClearAllTraceCharacteristics 10-33
- ClearDiffPairs 19-6
- ClearLayerMappings 25-144, 27-28
- ClearMessages 4-7, 6-16
- ClearRefPort 25-186, 27-29
- ClearRelative 25-91, 27-29
- ClearSolutionCache 26-165
- ClipPlane 25-223
- Clone 19-7
- CloneMaterial 6-17, 26-66
- CloneReportsFromDatasetSolution 10-34
- Close 6-18, 19-8
- CloseAllWindows 4-11, 8-258
- CloseEditor 27-166, 27-221
- CloseProject 4-12
- CloseProjectNoForce 4-13
- ClosestPointOnLine 25-69
- Combine 19-9
- comment lines 1-12, 1-14
- comparison operators 1-7
- CompInstance Functions 62
- CompInstance Script Commands (multiple)
- Component Manager Script Commands 26-140
- conditional statements
 - If...Then... Else 1-8
 - Select Case 1-8
 - types of 1-8
- Connect 8-132, 25-48, 27-29
- conventions
 - scripting help 1-1
- converting data types 1-10
- ConvertPlaneToTrace 22-4, 27-30
- ConvertPrimitives 23-1, 25-82
- ConvertToDynamic 26-277
- ConvertToParametric 26-277
- ConvertTraceToPlane 22-5, 27-30

- Coordinate System Methods 25-86
- Copy 8-104, 25-48, 27-30, 27-167
- CopyDesign 6-19
- CopyItemCommand 7-27
- CopyNamedExprToStack 13-17, 14-11
- CopyPlotSettings 10-35
- CopyRadFieldSetup 15-2
- CopyReportDefinition 10-35
- CopyReportsData 10-36
- Copyright and Trademark Information 2
- CopySetup
 - Analysis module command 11-7, 11-35
 - Optimetrics module command 11-7, 11-35
- CopyToPlanarEM 25-224, 27-31
- CopyTraceDefinitions 10-37
- CopyTracesData 10-38
- Core Global Script Context Commands 29-1
- Count 4-14
- CoupleEdgePorts 22-5
- CoverLines 8-133
- CoverSurfaces 8-134
- CPython 1-71
- Create Primitives 25-3
- Create Voids in Primitives 25-20
- Create3DComponent 8-6
- Create3DStructure 25-91, 27-31
- CreateArc 27-167
- CreateBondwire 8-11
- CreateBox 8-15
- CreateCableBundle 8-320
- CreateCableHarness 8-322
- CreateCircle 8-17, 25-6, 27-36, 27-168, 27-222
- CreateCircleVoid 25-20, 27-38, 27-222
- CreateCircuitPort 25-187, 27-39
- CreateClockSource 8-324
- CreateComponent 25-27, 27-43
- CreateCone 8-20
- CreateCS 25-87, 27-32
- CreateCutplane 8-23
- CreateCylinder 8-25
- CreateEdgePort 27-223
- CreateEllipse 8-27
- CreateEntityList 8-135
- CreateEquationCurve 8-30
- CreateEquationSurface 8-34
- CreateFaceCS 8-136, 8-142
- CreateFieldPlot 13-18
- CreateGroundPortComponent 22-5, 27-47
- CreateGroup 8-141
- CreateGroupSelected 25-92, 27-47
- CreateHelix 8-37
- CreateHole 25-30, 27-48
- CreateImage 27-169

CreateInterfaceGround 22-6, 27-49
 CreateInterfacePort 22-6, 27-50
 CreateInterfacePortComponent 22-7, 27-50
 CreateLine 25-8, 27-51, 27-170, 27-226
 CreateLineFromPolygon 25-58, 27-53
 CreateLineVoid 25-21, 27-53, 27-226
 CreateMeasure 25-31, 27-54, 27-227
 CreateNetClass 25-158, 27-59
 CreateNPort 25-192
 CreateNportCircuitElements 27-57
 CreateObjectFromEdges 8-149
 CreateObjectFromFace 8-151
 CreateObjectFromFaces 8-153
 CreateObjectFromPolygon 25-58, 27-59
 CreateOutputVariable 9-1
 CreatePin 25-33, 27-60, 27-171, 27-231
 CreatePinGroup 25-34, 27-61
 CreatePinGroupPort 25-35, 27-61
 CreatePoint 8-39
 CreatePolygon 25-14, 27-63, 27-172, 27-231
 CreatePolygonVoid 25-22, 27-64, 27-230
 CreatePolyline 8-41
 CreatePortInstancePorts 25-195, 27-65

CreatePortsOnComponents 25-195, 27-66
 CreatePWLSource 8-326
 CreateRectangle 8-46, 25-15, 27-66, 27-173, 27-232
 CreateRectangleVoid 25-24, 27-68
 CreateRegion 8-49
 CreateRegularPolygon 8-52
 CreateRegularPolyhedron 8-55
 CreateRelativeCS 8-154
 CreateReport 7-27, 10-39
 CreateReportFromTemplate 10-48
 CreateReportofAllQuantities 10-49
 CreateSphere 8-58
 CreateSpiral 8-61
 CreateText 25-17, 27-69, 27-173, 27-233
 CreateTorus 8-63
 CreateTrace 25-36, 27-72
 CreateUserDefinedPart 8-65
 CreateUserDefinedSolution 18-1
 CreateVia 25-42, 27-78, 27-233
 Cross 25-68
 Cut 27-174
 CutDesign 6-20
 CutOutSubDesign 25-224, 27-79

D

dataset commands

AddDataset 6-4, 7-7, 26-22
 DeleteDataset 6-21, 7-37, 26-67

- EditDataset 6-23, 7-44, 26-95
- ExportDataset 6-34, 7-61, 26-113
- ImportDataset 6-56, 7-109, 26-121
- DeactivateOpen 25-55, 27-84
- DeactivateShort 25-56, 27-84
- DecoupleEdgePorts 22-7
- Deembed 19-10
- DeembedBack 19-11
- DeembedFront 19-13
- DefeatureObjects 25-229, 27-85
- Definition Editor Script
 - Commands 27-1
- Definition Manager commands
 - AddDefinitionFromBlock 26-25
 - AddDefinitionFromLibFile 8-245
 - GetProjectMaterialNames 26-119
 - UpdateDefFromBlock 26-134, 26-136
- Definition Manager Script Commands 26-1
- Delete 8-258, 21-23, 22-7, 22-8, 23-1, 24-1, 25-49, 25-49, 25-56, 25-56, 27-86, 27-86, 27-175
- Delete [Interface Port] 22-7
- DeleteAllReports 10-51
- DeleteDataset 6-21, 7-37, 26-67
- DeleteDesign 6-21
- DeleteDesignInstance 7-38, 7-38
- DeleteEmptyGroups 8-156
- DeleteFarFieldSetup 15-3
- DeleteFieldPlot 13-29
- DeleteFieldVariation 7-38
- DeleteFullVariation 7-39
- DeleteImportData 12-1
- DeleteLastOperation 8-157
- DeleteMarker 10-50
- DeleteMarker[Fields Reporter] 13-30
- DeleteNamedExpr 13-31, 14-12
- DeleteNearFieldSetup 15-3
- DeleteOperation 8-158
- DeleteOutputVariable 7-40, 9-3
- DeletePinGroup 25-47, 27-87
- DeletePolylinePoint 8-105
- DeleteProbePortAndVia 22-8
- DeleteProject 4-15
- DeleteReport 10-52
- DeleteRuleSet 28-5
- DeleteRun 28-5
- DeleteSetup 15-4
- DeleteSetups
 - Optimetrics module command 11-8, 11-36
- DeleteSolutionVariation 7-41, 12-2
- DeleteSource [Interface Source] 22-8
- Deletetraces 10-52
- DeleteUneditablePlot 13-32
- DeleteUserDefinedSolutions 18-3
- DelNetClass 25-160, 27-86

- Design object commands 10-73, 10-74, 10-76, 10-77, 10-77, 10-92
- AddCartesianLimitLine 10-6
- AddCartesianXMarker 10-12
- AddCartesianYMarker 10-12
- AddDeltaMarker 10-14
- AddDesignVariablesForDynamicLink 7-10
- AddDynamicLink 7-11
- AddMarker 10-16
- AddMarkerToPlot 13-4
- AddNote 10-17
- AddTraceCharacteristics 10-19
- AddTraces 10-20
- AnalyzeDistributed 7-16
- ApplyMeshOps 7-17, 7-81
- CalculatorRead 13-7, 14-5
- ClearAllMarkers 10-33
- ClearAllTraceCharacteristics 10-33
- CloseAllWindows 4-11
- CopyReportDefinition 10-35
- CreateOutputVariable 9-1
- CreateReportFromTemplate 10-48
- DeleteAllReports 10-51
- DeleteOutputVariable 7-40, 9-3
- DeleteReport 10-52
- DeleteTraces 10-52
- DistributeAnalysis 7-42
- DoesOutputVariableExist 9-4
- Edit Layout 7-49
- EditMarker 10-56
- EditOutputVariable 7-51, 9-5
- ExportConvergence 7-60
- ExportMeshStats 7-66
- ExportModelMeshToFile 8-183, 10-65
- ExportPlot3DToFile 10-66
- ExportPlotImageToFile 13-52
- ExportPlotImageWithViewToFile 13-53
- ExportProfile 7-74
- ExportReport 7-75, 10-67
- ExportToFile 10-70
- Fast Transformation 7-132, 7-132
- GeometryCheckAndAutofix 8-261, 27-108
- GetDesignValidationInfo 7-88
- GetDisplayType 10-81
- GetLibraryDirectory 4-31
- GetMatchedObjectName 8-281
- GetModelUnits 8-283
- GetModule 7-93, 7-94, 7-96
- GetName 7-97, 10-85, 11-21, 11-53
- GetObjectsByMaterial 8-288
- GetObjectsInGroup 8-290
- GetObjPath 7-99, 10-86, 11-22, 11-54
- GetOutputVariableValue 7-99, 9-8
- GetProjectDirectory 4-36
- GetProperties 2-95, 6-51, 7-103

GetPropertyValue 2-96, 6-52, 7-104, 8-294, 10-86, 26-120

GetPropEvaluatedValue 6-47, 8-293

GetPropNames 7-106, 15-7

GetPropSIValue 6-49, 8-297

GetRegistryInt 4-38, 4-81

GetRegistryString 4-39, 4-82

GetSelections 8-300

GetSubGroupsInGroup 8-300

GetTempDirectory 4-43

GetVariables 2-98, 6-55, 7-107

GetVariableValue 2-98, 6-54, 7-108

GetVersion 4-44

ImportIntoReport 10-110

IsFeaturEnabled 4-45

PasteDesign 7-114

PasteReports 10-116

PasteReportsWithLegacyNames 10-116

PasteTracesWithLegacyNames 10-118

Redo 7-119

RenameDesignInstance 7-121

RenameReport 10-118

RenameTraces 10-119

RunToolkit 6-67, 7-123

SavePlotSettingsAsDefault 10-121

SetActiveEditor 7-102, 7-124

SetLibraryDirectory 4-69

SetProjectDirectoryVBCommand> 4-69

SetPropertyValue 2-99, 6-75, 7-136, 26-133

SetTempDirectory 4-73

SetVariableValue 2-101, 6-77, 7-138

Show Layout 7-137

ShowWindow 8-225

Solve 7-139, 7-140

StopSimLink 7-140

UpdateAllReports 10-124

Design Object Script Commands 7-1

Design Object SCripting commands

ExportNetworkData 7-67

Design Verification Script Commands 28-1

Desktop Commands For Registry Values 4-78

Desktop object commands

AddMessage 4-6

AreThereSimulationsRunning 4-7

ClearMessages 4-7

CloseProject 4-12

CloseProjectNoForce 4-13

Count 4-14

DeleteProject 4-15

DownloadJobResults 4-16

EnableAutosave 4-19

ExportOptionsFiles 4-19

- GetActiveProject 4-20
- GetActiveScheduler 4-21
- GetActiveSchedulerInfo 4-22
- GetAutosaveEnabled 4-23
- GetBuildTimeDateString 4-23
- GetChildNames 6-38, 7-82, 15-5
- GetChildNames Modeler 8-265
- GetChildTypes 6-40, 7-84, 8-271, 15-6
- GetChildTypes Optimetrics 11-20, 11-53
- GetCustomMenuSet 4-24
- GetDesigns 6-43
- GetDesktopConfiguration 4-27
- GetDistributedAnalysisMachines 4-28
- GetDis-trib-utedAna-lysisMachinesForDesignType 4-28
- GetMonitorData 4-34
- GetProjectList 4-37
- GetProjects 4-33, 4-38
- GetPropNames 6-48, 11-25, 11-56
- GetSchematicEnvironment 4-41
- KeepDesktopResponsive 4-47
- LaunchJobMonitor 4-47
- NewProject 4-48
- OpenMultipleProjects 4-50
- OpenProject 4-50
- PageSetup 4-52, 25-257, 27-132, 27-241
- PauseRecording 4-52
- PauseScript 4-53
- Print 4-54
- QuitApplication 4-55
- RefreshJobMonitor 4-55
- ResetLogging 4-56
- RestoreWindow 4-58
- ResumeRecording 4-59
- RunProgram 4-60
- RunScript 4-61
- SelectScheduler 4-64
- SetActiveProject 4-65
- SetActiveProjectByPath 4-66
- SetCustomMenuSet 4-67
- SetDesktopConfiguration 4-68
- SetSchematicEnvironment 4-73
- Sleep 4-75
- StopSimulations 4-76
- SubmitJob 4-76
- TileWindows 4-78
- Desktop Object Script Commands 4-1
- Desktop Scripting Conventions 1-85
- DetachEdges 8-160
- DetachFaces 8-161
- DisableDiffPairs 19-14
- Disconnect 25-49, 27-88
- DisplayPinNames 27-175

Distance 25-67
DistanceFromLine 25-69
DistributeAnalysis 7-42
DistributedAnalyzeSetup, Opti-
metrics module command 11-
9, 11-37
DoesNamedExpressionExists 13-
32, 14-12
DoesOutputVariableExist 9-4
DoesSupportTraceCharacteristics
10-53
Draw Menu commands 8-4
Duplicate 25-231, 27-88, 27-234
DuplicateAcrossLyrs 25-144, 27-89
DuplicateAlongLine 8-106
DuplicateAroundAxis 8-108
DuplicateMirror 8-111
DynamicMeshOverlays 21-25

E

Edit 7-47, 16-5, 16-10, 16-15, 16-
22, 21-25, 21-25, 22-8, 22-8,
23-2, 24-2, 25-49, 26-68, 26-
106, 26-166, 26-238, 26-278,
26-309, 26-345, 27-89
Edit Menu Commands 8-103
Edit3DComponent 8-73
Edit3DComponentDefinition 8-76,
25-232, 27-91
EditAntennaArraySetup 15-56
EditAntennaOverlay 15-14
EditBoxSetup 15-16
EditCartesianXMarker 10-54

EditCartesianYMarker 10-55
EditCircuitExcitations 22-20
EditCircuitPort [Interface Port] 22-21
EditCoSimulationOptions 7-43
EditDataset 6-23, 7-44, 26-95
EditFaceCS 8-163
EditFarFieldSphereSetup 15-18
EditImportData 7-46
EditInfiniteArray 7-47
EditInfiniteSphereSetup 15-21
EditLayoutForLayoutComponent 7-
49
EditLineSetup 15-23
EditMarker 10-56
EditMaterial 6-25, 26-97
EditNearFieldBoxSetup 15-24
EditNearFieldLineSetup 15-26
EditNearFieldRectangleSetup 15-28
EditNearFieldSphereSetup 15-29
EditNotes 7-50, 7-50
EditObjectCS 8-169
EditOptions 7-51
Editor object commands
 SetPropertyValue 2-99, 6-75, 7-
 136, 26-133
EditOutputVariable 7-51, 9-5
EditPolyline 8-78
EditQ3DSources 12-3
EditRadFieldSourceGroup 15-31

- EditRectangleSetup 15-33
- EditRelativeCS 8-176
- EditRuleSet 28-6
- EditRun 28-7
- EditSetup
 - optimization command 11-75
 - parametric command 11-65
 - sensitivity command 11-88
 - statistical command 11-95
- EditSolverOnDemandModel 26-192
- EditSources 12-7
- EditSphereSetup 15-34
- EditSurfaceMeshSummaryData 13-33
- EditUserDefinedSolution 18-4
- EditWithComps 26-193, 26-238, 26-278, 26-316, 26-345
- EMDesignOptions 7-53
- EnableAutoSave 4-19
- EnableDiffPairs 19-15
- EnableSetup 11-9, 11-38
- EnterComplex 13-37, 14-13
- EnterComplexVector 13-38, 14-14
- EnterCoord 13-39, 14-15
- EnterEdge 13-40, 14-16
- EnterLine 13-40, 14-16
- EnterOutputVar 13-41, 14-17
- EnterPoint 13-42, 14-18
- EnterQty 13-43, 14-18
- EnterScalar 13-43, 14-19
- EnterScalarFunc 13-44, 14-20
- EnterSurf 13-45, 14-20
- EnterVector 13-45, 14-21
- EnterVectorFunc 13-46, 14-22
- EnterVol 13-47, 14-22
- EraseMeasurements 25-238, 27-92, 27-235
- Event Callback Scripting 1-89
- Example Scripts 30-1
- Excitations Commands 22-1
- ExecuteScript 2-65
- Export 8-178, 26-114, 26-116, 26-118, 26-205, 26-256, 26-289, 26-323, 26-357
- ExportCableLibrary 8-331
- ExportCitiFile 19-16
- ExportConvergence 7-60
- ExportDataset 6-34, 7-61, 26-113
- ExportDOELocalSensitivity 11-49
- ExportDOELocalSensitivityCurve 11-50
- ExportDOEResponseCurve 11-46
- ExportDOEResponseCurveSlices 11-47
- ExportDOEResponseSurface 11-48
- ExportDXConfigFile, Optimetrics module command 11-10, 11-38
- ExportDXF 25-238, 27-94
- ExportEigenmodes 12-11
- ExportEignemodes 12-11
- ExportElementPatternToFile 15-62
- ExportEyeMaskViolation 10-57

ExportFieldPlot 13-47	ExportPlot3DToFile 10-66
ExportFieldsToFile 15-63	ExportPlotImageToFile 13-52
ExportFile 27-176	ExportPlotImageWithViewToFile 13-53
ExportForHSpice 7-61, 12-12	ExportProfile 7-74
ExportForHSpice (Layout Editor) 7-63	ExportRadiationFieldsToFile 15-67
ExportForSpice (Layout Editor) 7-65	ExportRadiationParametersToFile 15-69
ExportFullWaveSpice 19-17, 26-296	ExportReport 7-75, 10-67
ExportGDSII 25-145, 25-240, 27-97	ExportReportDataToFile 10-69
ExportGerber 25-243, 27-99	ExportRespSurfaceMinMaxTable 11-15, 11-43
ExportImageToFile 10-58	ExportRespSurfaceRefinePoints 11-16, 11-44
ExportMarkerTable 13-48	ExportRespSurfaceResponsePoints 11-17, 11-45
ExportMaterial 6-35, 26-115	ExportRespSurfaceVerificationPoints 11-17, 11-46
ExportMatlab 19-19	ExportScript 26-117, 26-330
ExportMesh Stats 7-66	ExportSpreadsheet 19-25
ExportModelMeshToFile 8-183, 10-65	ExportSurfaceMeshSummary 13-54
ExportNCDrill 25-245, 27-102	ExportTableToFile 10-70
ExportNMFData 12-16	ExportToFile 13-56
ExportOptimetricsProfile, Optimetrics module command 11-11, 11-39	ExportToHFSS 21-41, 27-105
ExportOptimetricsResults, Optimetrics module command 11-12, 11-40	ExportToQ3D 21-41, 27-105
ExportOptionsFiles 4-19	ExportTouchstone 19-28
ExportOutputVariables 9-6	ExportTouchstone2 19-30
ExportParametersToFile 15-65	ExportUniformPointsToFile 10-72
ExportParametricResults, Optimetrics module command 11-13, 11-41	Extract 19-33

F

FFTONReport 12-18

Field Calculator commands

DoesNamedExpressionExists
13-32, 14-12

Field Overlay module commands,
GetFieldPlotNames 13-58

field overlay script commands 13-1

Field Overlays module commands

CalcOp 13-6, 14-3

CalcStack 13-7, 14-6

ChangeGeomSettings 13-14,
14-7

ClcMaterial 13-15, 14-9

ClearAllNamedExpr 13-16, 14-
10

CopyNamedExprToStack 13-
17, 14-11

CreateFieldPlot 13-18

DeleteFieldPlot 13-29

DeleteNamedExpr 13-31, 14-12

DeleteUneditablePlot 13-32

EditSurfaceMeshSummaryData
13-33

EnterComplex 13-37, 14-13

EnterComplexVector 13-38, 14-
14

EnterCoord 13-39, 14-15

EnterEdge 13-40, 14-16

EnterLine 13-40, 14-16

EnterOutputVar 13-41, 14-17

EnterPoint 13-42, 14-18

EnterQty 13-43, 14-18

EnterScalar 13-43, 14-19

EnterScalarFunc 13-44, 14-20

EnterSurf 13-45, 14-20

EnterVector 13-45, 14-21

EnterVectorFunc 13-46, 14-22

EnteVol 13-47, 14-22

ExportFieldPlot 13-47

ExportOnGrid 13-49

ExportSurfaceMeshSummary 13-54

GetFieldFolderNames 13-57

GetFieldPlotQuantityName 13-58

GetMeshPlotNames 13-59, 21-42

HidePolarPlot 13-61

HideRadiatedPlotOverlay 13-62

ModifyFieldPlot 13-64

ReassignFieldPlot 13-66

RenameFieldPlot 13-69

RenamePlotFolder 13-70

SaveFieldsPlots 13-71

SetFieldPlotSettings 13-73

SetPlotFolderSettings 13-76

UpdateAllFieldsPlots 13-82

UpdateQuantityFieldsPlots 13-83

Fields Calculator commands

AddNamedExpr 13-5, 14-2

CalcOp 13-6, 14-3

- CalcStack 13-7, 14-6
 - CalculatorWrite 13-8, 14-6
 - ChangeGeomSettings 13-14, 14-7
 - ClcEval 13-14, 14-8
 - ClcMaterial 13-15, 14-9
 - ClcMaterialValue 14-10
 - ClearAllNamedExpr 13-16, 14-10
 - CopyNamedExprToStack 13-17, 14-11
 - DeleteNamedExpr 13-31, 14-12
 - EnterComplex 13-37, 14-13
 - EnterComplexVector 13-38, 14-14
 - EnterCoord 13-39, 14-15
 - EnterEdge 13-40, 14-16
 - EnterLine 13-40, 14-16
 - EnterOutputVar 13-41, 14-17
 - EnterPoint 13-42, 14-18
 - EnterQty 13-43, 14-18
 - EnterScalar 13-43, 14-19
 - EnterScalarFunc 13-44, 14-20
 - EnterSurf 13-45, 14-20
 - EnterVector 13-45, 14-21
 - EnterVectorFunc 13-46, 14-22
 - EnterVol 13-47, 14-22
 - ExportOnGrid 14-23
 - ExportToFile 14-26
 - Fields Calculator Script Commands 14-1
 - Fields reporter module commands
 - DeleteMarker[Fields Reporter] 13-30
 - ExportMarkerTable 13-48
 - Fillet 8-184
 - FilterObjectList 25-60, 27-105
 - FindObjects 25-59
 - FindObjectsByPoint 25-61, 27-106
 - FindObjectsByPolygon 25-61, 27-107
 - FlattenGroup 8-186
 - FlipHorizontal 25-53, 27-107, 27-177, 27-236
 - FlipVertical 25-53, 27-108, 27-177, 27-236
 - footprint
 - editor scripts 27-195
 - manager script commands 26-215
 - For...Next loop 1-9
 - functions
 - VBScript procedures 1-10
- G**
- GenerateAllUserDefinedModels 8-260
 - GenerateHistory 8-187
 - GenerateUserDefinedModel 8-260
 - GenerateVariationData Parametric, parametric command 11-18, 11-51, 11-67
 - GeometryCheckAndAutofix 8-261, 27-108

Get3DComponentDefinitionNames 8-82	GetAutoSaveEnabled 4-23
Get3DComponentInstanceNames 8-83	GetAvailableDisplayTypes 10-76
Get3DComponentMaterialNames 8-84	GetAvailableReportTypes 10-77
Get3DComponentMaterialProperties 8-84	GetAvailableSolutions 10-77
Get3DComponentParameters 8-85	GetAvailableVariations 12-19
Get3DComponentPartNames 8-86	GetBBox 25-62, 27-110
GetActiveCoordinateSystem 8-188	GetBodyNamesByPosition 8-263
GetActiveCoordinateSystemTransform 8-189	GetBoundingCircleCenter 25-78
GetActiveDesign 6-36	GetBoundingCircleRadius 25-78
GetActiveEditor (Layout Editor) 7-82	GetBuildDateTimeString 4-23
GetActiveProject 4-20	GetCallback 2-67
GetActiveScheduler 4-21	GetChangedProperty 2-67
GetActiveSchedulerInfo 4-22	GetChild Object (Report Setup), Report module command 10-78
GetAdaptiveFreq 12-19	GetChildNames 6-38, 7-82, 15-5
GetAllBoundariesList 22-23	GetChildNames (Optimetrics), Report module command 10-78
GetAllCategories 10-73	GetChildNames Modeler 8-265
GetAllLayerNames 25-148, 27-110, 27-237	GetChildObject Design 6-38, 7-83
GetAllPortsList 22-24	GetChildObject Modeler 8-268
GetAllQuantities 10-74	GetChildObject Radiation 15-6
GetAllReportNames 10-75	GetChildTypes 6-40, 7-84, 8-271, 15-6
GetAllSolutionNames 21-41	GetChildTypes Optimetrics 11-20, 11-53
GetApplication 2-66	GetChildTypes ReportSetup 10-80
GetArrayVariables 2-94, 6-37	GetClosestPoint 25-80
	GetClosestPoints 25-80
	GetCompInstanceFromRefDes 25-250, 27-111
	GetComponentInfo 25-249, 27-111
	GetComponentName 62

GetComponentPinInfo 25-251, 27-112	lysisMachinesForDesignType 4-28
GetComponentPins 25-250, 27-113	GetDocumentNames 17-8
GetCoordinateSystems 8-189	GetDynLinkIntrinsicVariables 10-82
GetCSObjects	GetDynLinkQtyValueState 10-82
Coordinate Systems 25-62, 25-91, 27-110	GetDynLinkTraces 10-83
GetCurvePropServerName 10-80	GetDynLinkVariableValues 10-84
GetData 26-206, 26-257, 26-261, 26-291, 26-324, 26-358	GetEdgeByPosition 8-273
GetDataExpressions 10-95	GetEdgeIDsFromFace 8-275
GetDataUnits 10-96	GetEdgeIDsFromObject 8-275
GetDefinitionManager 6-41	GetEdgeLength 8-276
GetDependentFiles 6-41	GetEditor 2-69, 7-89
GetDescription 2-68	GetEditor (Layout Editor) 7-89
GetDesign 2-69, 2-69	GetEditorName [Layout] 25-252, 27-114
GetDesignID 7-86	GetEntityListIDByName 8-277
GetDesignName 7-87	GetEvaluatedText 2-70
GetDesigns 6-43	GetExeDir 4-29
GetDesignType 7-87	GetFaceArea 8-277
GetDesignValidationInfo 7-88	GetFaceByPosition 8-279
GetDesignVariableNames 10-96	GetFaceCenter 8-278
GetDesignVariableUnits 10-97	GetFaceIDs 8-280
GetDesignVariableValue 10-98	GetFaceIDsOfSheet 8-281
GetDesignVariationKey 10-99	GetFieldFolderNames 13-57
GetDisplayType 10-81	GetFieldPlotNames, Field Overlay module command 13-58
GetDistributedAnalysisMachines 4-28	GetFieldPlotQuantityName 13-58
GetDis-trib-utedAna-	GetFileName 2-71
	GetFrequencies 19-34
	GetFrequencyCount 19-35

GetGeo- metryId- sForAllNetLay- erCombinations 7-91	GetNPortData 26-210
GetGeo- metryId- sForNetLayerCombinations 7-90	GetNumObjects 8-283
GetHidden 2-71	GetObjectIDByName 8-284
GetHoles 25-75	GetObjectName 8-285
GetImagDataValues 10-100	GetObjectNameByEdgeID 8-285
GetInstanceID 63	GetObjectNameByFaceID 8-286
GetInstanceName 64	GetObjectNameByID 8-287
GetIntersectionType 25-79	GetObjectNameByVertexID 8-287
GetLayerInfo 25-149, 27-237	GetObjectsByMaterial 8-288
GetLibraryDirectory 4-31	GetObjectShapeType 8-289
GetMatchedObjectName 8-281	GetObjectsIngroup 8-290
GetMaterialList 25-252, 27-115	GetObjectVolume 8-291
GetMeshOperations 21-42	GetObjPath 6-46, 7-99, 8-291, 10-86, 11- 22, 11-54
GetMeshPlotNames 13-59	GetOptimetricsResults, Optimetrics mod- ule command 11-24
GetModelBoundingBox 8-282	GetOutputVariableValue 7-99, 9-8
GetModelUnits 8-283	GetParentDesign 64
GetModule 7-93, 7-94, 7-96	GetPartsForUserDefinedModel 8-292
GetMonitorData 4-34	GetPath 6-47
GetName 6-45, 7-97, 7-98, 10-85, 11-21, 11-53, 19-36	GetPerQuantityPrimarySweepValues 10- 101
GetNames 26-207, 26-258, 26-262, 26-292, 26-325, 26-359	GetPersonalLibDirectory 4-35
GetNetClasses 25-160, 27-115	GetPoints 8-293, 25-74
GetNetClassNets 25-160, 27-115	GetPolygon 25-63, 27-117
GetNetConnections 25-161, 27- 116	GetPolygonDef 25-63, 27-117
	GetPortCount 19-36
	GetPortInfo 25-198, 27-117
	GetPortNumber 19-37
	GetPostProcSettings 19-39

GetProcessID 4-36	GetRegistryInt 4-38, 4-81
GetProgress 2-72, 2-72	GetRegistryString 4-39, 4-82
GetProjectDirectory 4-36	GetRelativeCoordinateSystems 8-299
GetProjectList 4-37	GetRe-
GetProjectMaterialNames 26-119	portSum-
GetProjects 4-33, 4-38	maryForRegressionTesting 10-91
GetProperties 2-95, 6-51, 7-103, 25-253, 26-264, 27-120, 27-178, 27-239	GetReportTraceNames 10-90
GetPropertyValue 2-96, 6-52, 7-104, 8-294, 10-86, 25-253, 26-120, 27-121, 27-178	GetRunStatus 2-75
GetPropEvaluatedValue 6-47, 8-293	GetSelections 8-300, 25-254, 25-254, 27-121
GetPropHost 2-73, 2-73, 7-102, 7-102, 65	GetSetupData 21-43
GetPropNames 6-48, 7-106, 10-88, 11-25, 11-56, 15-7	GetSetupNames 15-5, 15-8
GetPropNames Modeler 8-296	GetSetupNames (Optimetrics), Optimetrics module command 11-19, 11-19, 11-26, 11-51, 11-52, 11-57
GetPropServerName 65	GetSetupNamesByType (Optimetrics), Optimetrics module command 11-27, 11-58
GetPropServers 2-73	GetSolutionContexts 10-91
GetPropSIValue 6-49, 8-297	GetSolutionDataPerVariation 10-92
GetPropTabType 2-74	GetSolutionVariation 19-40
GetPropValue Modeler 8-298	GetSolutionVersionID 12-21
GetPropValue Optimetrics 11-25, 11-57	GetSolveRangeInfo 12-21
GetPropValue Project 6-50, 7-106, 15-7	GetSolverOnDemandData 26-210
GetPropValue Reporter 10-89	GetSolverOnDemandModelList 26-211
GetQtyExpressionsForSourceTrace 10-89	GetSourceContexts 12-22
GetReadOnly 2-75	GetSourceData 12-23
GetRealDataValues 10-102	GetStackupLayerNames 25-151, 27-240

- GetSubGroupsInGroup 8-300
 - GetSweepNames 10-103
 - GetSweepUnits 10-104
 - GetSweepValues 10-105
 - GetSysLibDirectory 4-42
 - GetTabTypeName 2-76
 - GetTempDirectory 4-43
 - GetText 2-76
 - GetTool 4-85
 - GetTopDesignList 6-54
 - GetTopEntryValue 13-60, 14-29
 - GetUserDefinedSolutionNames 18-6
 - GetUserDefinedSolutionProperties 18-7
 - GetUserLibDirectory 4-44
 - GetUserPosition 8-301
 - GetValidISolutionList 12-23
 - GetValue 2-77
 - GetVariables 2-98, 6-55, 7-107
 - GetVariableValue 2-98, 6-54, 7-108
 - GetVariation 19-41
 - GetVersion 4-44
 - GetVer-
texIDFromNameForFirstOper-
ation 8-302
 - GetVertexIDsFromEdge 8-303
 - GetVertexIDsFromFace 8-303
 - GetVertexIDsFromObject 8-304
 - GetVertexPosition 8-305
 - GetVisibleLevels 27-179
 - GetWireBodyNames 8-305
 - GetX (Layout Editor) 25-65
 - GetY 25-66
 - GridSetup 27-179
 - Group 25-92, 27-121
 - GroupPlotCurvesByGroupingStrategy 10-109
- H**
- HasArcs 25-75
 - HasFields 12-24
 - HasHoles 25-74
 - HasMatrixData 12-24
 - HasSameData 19-42
 - HasSelfIntersections 25-75
 - Heal (Layout Editor) 25-255, 27-121
 - HealObject 8-190
 - HideAntennaParametersOverlay 13-61
 - HidePolarPlot 13-61
 - HideRadiatedPlotOverlay 13-62
 - hierarchy of variables in HFSS 1-76
 - HighlightNet 25-162, 27-124
- I**
- Icepak boundary condition commands
 - ExportDOELocalSensitivity 11-49
 - ExportDOELocalSensitivityCurve 11-50
 - ExportDOEResponseCurve 11-46

- ExportDOEResponseCurveSlices 11-47
- ExportDOEResponseSurface 11-48
- ImportIDX 4-100
- If...Then... Else statement 1-8
- Import (3D Modeler command) 8-194
- ImportANF 4-86, 4-87
- ImportAutoCAD 4-89
- ImportAWRMicrowaveOffice 4-90
- ImportCableLibrary 8-332
- ImportDataset 6-56, 7-109, 26-121
- ImportDXF 8-198, 27-125
- ImportEDB 4-92
- ImportExport Tool 4-85
- ImportExtracta 4-93
- ImportFromClipboard 8-202
- ImportGDSII 4-94, 4-95, 8-203, 27-129
- ImportIDF 4-96, 4-99
- ImportIDX 4-100
- ImportIntoReport 10-110
- ImportIPC 4-103
- ImportODB 4-104
- ImportOutputVariables 9-9
- ImportReportDataIntoReport 10-111
- ImportSetup, Optimetrics module command 11-28, 11-59
- ImportSolution 12-26
- ImportTable 12-27
- ImportXFL 4-105
- Imprint 8-206
- ImprintProjection 8-208
- include files
 - scripts 1-10
- indentation, IronPython 1-21
- InputBox function 1-11
- Insert3DComponent 8-86
- InsertBoxSetup 15-37
- InsertComponent 8-87
- InsertDesign 6-58, 6-60, 7-111
- InsertDesignWithWorkflow 6-60
- InsertFarFieldSphereSetup 15-39
- InsertInfiniteSphereSetup 15-41
- InsertLineSetup 15-43
- InsertNearFieldBoxSetup 15-45
- InsertNearFieldLineSetup 15-47
- InsertNearFieldRectangleSetup 15-48
- InsertNearFieldSphereSetup 15-50
- InsertRectangleSetup 15-52
- InsertSetup
 - optimization command 11-80
 - parametric command 11-33, 11-64, 11-67, 11-74
 - statistical command 11-98
- InsertSphereSetup 15-53
- Intersect 8-210, 25-81, 25-81, 25-84, 25-84, 27-131, 27-131, 27-240

IronPython 1-15
 indentation in 1-21
 IsArc 25-67, 25-67, 25-77, 25-77
 IsBox 25-77
 IsCircle 25-78
 IsClosed 25-72
 IsConvex 25-76
 IsDataComplex 10-106
 IsEqual 25-67
 IsFeatureEnabled 4-45
 IsFieldAvailableAt 12-28
 IsParametric 25-76
 IsPerQuantityPrimarySweep 10-107
 IsPoint 25-77
 IsSegment 25-77
 IsUsed 26-211, 26-258, 26-264, 26-292, 26-325, 26-359
 IsValueConstant 2-78

K

KeepDesktopResponsive 4-47
 keywords, VBScript 1-12, 1-14

L

LaunchJobMonitor 4-47
 Layers Methods 25-128
 layout
 script interrogation methods 25-57
 Layout Interrogation 25-57

Layout Scripting 1-88, 25-1, 25-96, 25-205
 Layout3DMeshOverlay 21-44
 LayoutMeshOverlay 21-43
 ListMatchingVariations 12-29
 ListValuesOfVariable 12-30
 ListVariations 21-44
 LoadNamedExpressions 13-63, 14-30
 LoadSolution 19-43
 logical operators 1-7
 looping through code
 Do ... Loop 1-8
 For ... Next 1-8

M

Mag (Layout Editor) 25-67
 material commands
 AddDefinitionFromBlock 26-25
 AddDefinitionFromLibFile 8-245
 AddMaterial 6-7, 26-30, 26-67
 AreMaterialPropertiesEqual 26-51
 AreSurfaceMaterialPropertiesEqual 26-52
 CloneMaterial 6-17, 26-66
 EditMaterial 6-25, 26-97
 ExportMaterial 6-35, 26-115
 RemoveMaterial 6-64, 26-127
 UpdateDefFromBlock 26-134
 UpdateDefFromBlockEx 26-136
 Materials Scripting Support 2-22

Microsoft

VBScript user's guide 1-11

Mirror 8-114

MirrorX 25-74

Model Manager Script
Commands 26-266

Modeler Menu Commands 8-123

ModifyFieldPlot 13-64

ModifyLibraries 26-331

ModifyNetClass 25-165, 27-131

module script commands

field overlay 13-1

radiation 15-2

ModuleHasMesh 12-25

modules in HFSS scripting 1-76

Move 8-116, 25-54, 25-54, 25-54,
25-68, 25-68, 25-68, 25-72,
25-72, 25-72, 27-132, 27-132,
27-132, 27-181, 27-240

MoveCSToEnd 8-211

MoveEntityToGroup 8-212

MoveFaces 8-213

MovePlotCurveToGroup 10-112

MovePlotCurveToNewGroup 10-
113

MsgBox function 1-11

MultipleEdit 23-2

N

Named Arguments 1-85

NdExplorer

Script Commands 19-1

NetClass Methods 25-156

Network Data Explorer Manager Com-
mands 26-295

NewProject 4-48

Normalize 25-69

O

oAnsoftApp object 1-76

Object Identifiers and Script Record-
ing 25-2Object Movement and Modification
Methods 25-47Object Oriented Property Scripting 2-
1

oDesign object 1-76

oDesktop object 1-76

oEditor object 1-76

OffsetFaces 8-118

oModule object 1-76

Open 19-45

OpenAndConvertProject 4-49

OpenMultipleProjects 4-50

OpenProject 4-50, 27-241

OpenWindowForAllReports 10-113

OpenWindowForReports 10-114

operators

arithmetic 1-6

categories in VBScript 1-5

comparison 1-7

- concatenation 1-7
- logical 1-7
- precedence of 1-5
- oProject object 1-76
- Optimetrics module commands
 - DeleteSetups 11-8, 11-36
 - DistributeAnalyzeSetup 11-9, 11-37
 - EnableSetup 11-9, 11-38
 - ExportDXConfigFile 11-10, 11-38
 - ExportOptimetricsProfile 11-11, 11-39
 - ExportOptimetricsResults 11-12, 11-40
 - ExportParametricResults 11-13, 11-41
 - ExportRespSurfaceMinMaxTable 11-15, 11-43
 - ExportRespSurfaceRefinePoints 11-16, 11-44
 - ExportRespSurfaceResponsePoints 11-17, 11-45
 - ExportRespSurfaceVerificationPoints 11-17, 11-46
 - GetChildNames 11-19, 11-20, 11-52, 11-52
 - GetOptimetricResult 11-22, 11-55
 - GetOptimetricsResults 11-24
 - GetSetupNames 11-26, 11-58
 - GetSetupNamesByType 11-27, 11-58
 - ImportSetup 11-28, 11-59
 - RenameSetup 11-30, 11-61
 - SolveSetup 11-7, 11-31, 11-32, 11-35, 11-63, 11-64
- Optimetrics Module Script Commands 11-1
- optimization commands
 - EditSetup 11-75
 - InsertSetup 11-80
- Optimization Script Commands 11-75
- Other Creation Methods 25-25
- Other oEditor Commands 8-243
- output variable commands
 - CreateOutputVariable 9-1
 - DeleteOutputVariable 7-40, 9-3
 - DoesOutputVariableExist 9-4
 - EditOutputVariable 7-51, 9-5
 - GetOutputVariableValue 7-99, 9-8
- Output variable commands
 - ExportOutputVariables 9-6
 - ImportOutputVariables 9-9
- Output Variable Script Commands 9-1
- OverlayCurrents 7-113
- OverlayFarField 7-113
- OverlayMesh 7-114
- OverlayNearField 7-114

P

Padstack Manager Script Commands 26-302

PageSetup 8-306

PageSetup (Layout Editor) 4-52, 25-257, 27-132

Pan 27-182

parametric commands

- EditSetup 11-65
- GenerateVariationData Parametric 11-18, 11-51, 11-67
- InsertSetup 11-33, 11-64, 11-67, 11-74

Parametric Script Commands 11-65

Paste 6-62, 8-328, 8-329, 25-54, 27-182

Paste (Model Editor) 8-119

Paste (Project Object) 6-62

PasteDesign 7-114

PasteItemCommand 7-116

PastePlotSettings 10-115

PasteRadFieldSetup 15-8

PasteReports 10-116

PasteReportsWithLegacyNames 10-116

PasteSetup

- Optimetrics module command 11-29, 11-60

PasteTraces 10-117

PasteTracesWithLegacyNames 10-118

PauseRecording 4-52

PauseScript 4-53

Point (Layout Editor) 25-63, 27-133

Point Object 25-64

PointInPolygon 25-79

Polygon 25-63, 27-133

Polygon Object 25-69

Port Methods 25-175

PositionRelative 25-92, 27-133

Print 4-54

Project object commands

- AddDataset 6-4, 7-7, 26-22
- AddMaterial 6-7, 26-30, 26-67
- AreMaterialPropertiesEqual 26-51
- AreSurfaceMaterialPropertiesEqual 26-52
- ChangeProperty 26-53, 27-153
- CloneMaterial 6-17, 26-66
- Close 6-18
- CopyDesign 6-19
- CutDesign 6-20
- DeleteDataset 6-21, 7-37, 26-67
- DeleteDesign 6-21
- EditDataset 6-23, 6-34, 6-56, 7-44, 7-60, 7-109, 26-95, 26-113, 26-121
- EditMaterial 6-25, 26-97
- ExportMaterial 6-35, 26-115
- GetActiveDesign 6-36
- GetArrayVariables 2-94, 6-37

GetChildObject Design 6-38, 7-83	GetTopDesignList 6-54
GetChildObject Modeler 8-268	GetTopEntryValue 13-60, 14-29
GetChildObject Radiation 15-6	GetVariables 2-98, 6-55, 7-107
GetDependentFiles 6-41	GetVariableValue 2-98, 6-54, 7-108
GetDesign 2-69	InsertDesign 6-58, 7-111
GetDesignID 7-86	LoadSolution 19-43
GetDesignName 7-87	Paste 6-62, 6-62
GetDesignType 7-87	Redo 6-62
GetName 6-45	RemoveAllUnusedDefinitions 6-63
GetObjPath 6-46	RemoveMaterial 6-64, 26-127
GetPath 6-47	RemoveUnusedDefinitions 6-65, 26-131
GetProjectMaterialNames 26-119	Rename 6-66
GetProperties 2-95, 6-51, 7-103, 25-253, 27-120, 27-178, 27-239	RestoreProjectArchive 4-57
GetPropertyValue 2-96, 6-52, 7-104, 8-294, 10-86, 25-253, 26-120, 27-121, 27-178	Save 6-68
GetPropEvaluatedValue 6-47, 8-293	SaveAs 6-68
GetPropNames Modeler 8-296	SaveAsStandAloneProject 6-71
GetPropSIValue 6-49, 8-297	SaveProjectArchive 6-72
GetPropvalue Modeler 8-298	SetActiveDesign 6-74
GetPropvalue Optimetrics 11-25, 11-57	SetPropertyValue 2-99, 6-75, 7-136, 25-263, 26-133, 27-244
GetPropvalue Project 6-50, 7-106	SetPropValue Design 7-135, 15-10
GetPropvalue Radiation 15-7	SetPropValue Optimetrics 11-30, 11-62
GetPropvalue Reporter 10-89	SetPropValue Project 6-74, 10-122
GetSolutionVariation 19-40	SetVariableValue 2-101, 6-77, 7-138
	SimulateAll 6-11, 6-78
	Undo 6-78
	UpdateDefinitions 6-79, 26-139
	Project Object Script Commands 6-1

ProjectSheet 8-215

property commands

ChangeProperty 26-53, 27-153

GetArrayVariables 2-94, 6-37

GetProjectMaterialNames 26-119

GetProperties 2-95, 6-51, 7-103, 25-253, 27-120, 27-178, 27-239

GetPropertyValue 2-96, 6-52, 7-104, 8-294, 10-86, 25-253, 26-120, 27-121, 27-178

GetPropEvaluatedValue 6-47, 8-293

GetPropSIValue 6-49, 8-297

GetTopEntryValue 13-60, 14-29

GetVariables 2-98, 6-55, 7-107

GetVariableValue 2-98, 6-54, 7-108

SetPropertyValue 2-99, 6-75, 7-136, 25-263, 26-133, 27-244

SetVariableValue 2-101, 6-77, 7-138

Property Script Commands 2-30

Conventions used in this
Chapter 2-41

Object Script Property Function
Summary 2-32

PropertyExists 2-79

PropHost Functions 2-56

PullExcitations 21-44

PurgeHistory 8-216

PushExcitations 7-116, 25-258, 27-136

Layout Editor 7-116, 25-258, 27-136

Q

QuitApplication 4-55

R

Radiation boundary

Infinite Sphere 16-1, 16-5

Near Field Line 16-8, 16-10

Near Field Plane 16-12, 16-15

Near Field Sphere 16-18, 16-22

Radiation module commands

AddAntennaOverlay 15-11

AddRadFieldSourceGroup 15-13

CopyRadFieldSetup 15-2

DeleteFarFieldSetup 15-3

DeleteNearFieldSetup 15-3

DeleteSetup 15-4

EditAntennaArraySetup 15-56

EditAntennaOverlay 15-14

EditBoxSetup 15-16

EditFarFieldSphereSetup 15-18

EditInfiniteSphereSetup 15-21

EditLineSetup 15-23

EditNearFieldBoxSetup 15-24

EditNearFieldLineSetup 15-26

EditNearFieldRectangleSetup 15-28

- EditNearFieldSphereSetup 15-29
- EditRadFieldSourceGroup 15-31
- EditRadiatedPowerCalculationMethod 15-5
- EditRectangleSetup 15-33
- EditSphereSetup 15-34
- ExportElementPatternToFile 15-62
- ExportFieldsToFile 15-63
- ExportParametersToFile 15-65
- ExportRadiationFieldsToFile 15-67
- ExportRadiationParametersToFile 15-69
- GetSetupNames 15-8
- InsertBoxSetup 15-37
- InsertFarFieldSphereSetup 15-39
- InsertInfiniteSphereSetup 15-41
- InsertLineSetup 15-43
- InsertNearFieldBoxSetup 15-45
- InsertNearFieldLineSetup 15-47
- InsertNearFieldRectangleSetup 15-48
- InsertRectangleSetup 15-52
- InsertSphereSetup 15-50, 15-53
- PasteRadFieldSetup 15-8
- RenameSetup 15-9
- Radiation Module Script Commands 15-1, 16-1
- radiation script commands 15-2
- ReassignFieldPlot 13-66
- Record Script and Edit Properties 2-101
- Redo 7-120, 27-183
 - design-level command 7-119
 - project-level command 6-62
- references, for VBScript 1-11
- RefreshJobMonitor 4-55
- RefreshMeshOverlays 21-45
- ReleaseData 10-108
- Remove 26-124, 26-126, 26-128, 26-130, 26-212, 26-259, 26-293, 26-326, 26-360
- RemoveAllUnusedDefinitions 6-63
- RemoveBadEdges 8-308
- RemoveBadFaces 8-308
- RemoveBadVertices 8-309
- RemoveCables 8-332
- RemoveImportData 7-120
- RemoveLayer 27-140, 27-242
- RemoveLevels 27-183
- RemoveMaterial 6-64, 26-127
- RemoveModelingProperties 7-121
- RemovePort 25-200, 27-184, 27-242
- RemovePortsFromAllNets 25-201, 27-140
- RemovePortsFromNet 25-201, 27-141
- RemovePortsOnComponents 25-202, 27-142
- RemoveProp 2-79
- RemoveRefPort 22-25

- RemoveScript 26-129, 26-332
- RemoveSolverOnDemandModel 26-213
- RemoveUnused 26-213, 26-260, 26-265, 26-294, 26-327, 26-361
- RemoveUnusedDefinitions 6-65, 26-131
- Rename 6-66, 19-45, 21-45, 22-24, 23-2, 24-2
- Rename [Interface Port] 22-24
- RenameDesignInstance 7-121, 7-122
- RenameFieldPlot 13-69
- RenamelImportData 7-122
- RenamePart 8-310
- RenamePlotFolder 13-70
- RenameReport 10-118
- RenameRuleSet 28-9
- RenameRun 28-10
- RenameSetup
 - Optimetrics module command 11-30, 11-61
 - Radiation module command 15-9
- RenameSource [Interface Source] 7-123, 22-24
- RenameTraces 10-119
- Renormalize 19-47
- Reorder 19-48, 19-49
- Repeating a Statement Until a Condition Becomes True 1-9
- Repeating Statements While a Condition is True 1-9
- Replacewith3DComponent 8-217
- Report module commands
 - GetChildNames 10-78
- Reporter editor commands
 - AddAllEyeMeasurements 10-6
 - AddCartesianLimitLine 10-6
 - AddCartesianLimitLineFromCurve 10-8
 - AddCartesianLimitLineFromEquation 10-10
 - AddCartesianXMarker 10-12, 10-12
 - AddCartesianYMarkerToStack 10-13
 - AddDeltaMarker 10-14
 - AddMarker 10-16
 - AddMarkerToPlot 13-4
 - AddNote 10-17
 - AddTraceCharacteristics 10-19
 - AddTraces 10-20
 - ApplyReportTemplate 10-23
 - ChangeProperty 7-18, 10-23
 - ClearAllMarkers 10-33
 - ClearAllTraceCharacteristics 10-33
 - CloneReportsFromDatasetSolution 10-34
 - CopyPlotSettings 10-35
 - CopyReportDefinition 10-35

CopyReportsData 10-36	GetAvailableDisplayTypes 10-76
CopyTraceDefinitions 10-37	GetAvailableReportTypes 10-77
CopyTracesData 10-38	GetAvailableSolutionss 10-77
CreateReport 7-27, 10-39	GetChildTypes ReportSetup 10-80
CreateReportFromTemplate 10-48	GetCurvePropServerName 10-80
CreateReportOfAllQuantities 10-49	GetDisplayType 10-81
DeleteAllReports 10-51	GetDynLinkIntrinsicVariables 10-82
DeleteReport 10-52	GetDynLinkQtyValueState 10-82
DeleteTraces 10-52	GetDynLinkTraces 10-83
DoesSupportTraceCharacteristics 10-53	GetDynLinkVariableValues 10-84
EditCartesianXMarker 10-54	GetPropNames 10-88
EditCartesianYMarker 10-55	GetQtyExpressionsForSourceTrace 10-89
EditMarker 10-56	GetReportSummaryForRegressionTesting 10-91
ExportEyeMaskViolation 10-57	GetReportTraceNames 10-90
ExportImageToFile 10-58	GetSolutionContexts 10-91, 10-92
ExportPlot3DToFile 10-66	GroupPlotCurvesByGroupingStrategy 10-109
ExportPlotImageToFile 13-52, 13-53	ImportReportDataIntoReport 10-111
ExportReport 7-75, 10-67	MovePlotCurveToGroup 10-112
ExportReportDataToFile 10-69	MovePlotCurveToNewGroup 10-113
ExportTableToFile 10-70	OpenWindowForAllReports 10-113
ExportToFile 10-70, 10-110	OpenWindowForReports 10-114
ExportUniformPointsToFile 10-72	PastePlotSettings 10-115
FFTONReport 12-18	PasteReports 10-116
GetAllCategories 10-73	PasteReportsWithLegacyNames 10-116
GetAllQuantities 10-74	PasteTraces 10-117
GetAllReportNames 10-75	PasteTracesWithLegacyNames 10-118

- RenameReport 10-118
- RenameTraces 10-119
- ResetPlotSettings 10-120
- SavePlotSettingsAsDefault 10-121
- SetLinkOutputTraces 10-121
- UnGroupPlotCurvesInGroup 10-123
- UpdateAllReports 10-124
- UpdateReports 10-125
- UpdateTraces 10-125
- UpdateTracesContextAndSweeps 10-128
- Reporter Editor Script
 - Commands 10-1
- Reporter module commands
 - GetChildObject 10-79
- ReportTemplates 7-123
- Reset 19-50
- ResetLogging 4-56
- ResetPlotSettings 10-120
- RestoreWindow 4-58
- ResumeRecording 4-59
- Rotate 8-120, 25-54, 25-68, 25-73, 27-143, 27-185, 27-243
- RunAllDV 28-10
- RunAllRuleSetDV 28-11
- RunDV 28-11
- Running Instance Manager Script
 - Commands 5-1
- RunProgram 4-60
- RunScript 4-61
- RunScriptWithArguments 4-62
- RunToolkit 6-67, 7-123
- S**
 - sample scripts
 - simple HFSS 1-12
 - simple Q3D Extractor 1-14
 - waveguide 30-11
 - Save 6-68, 27-185, 27-243
 - SaveAs 6-68
 - SaveAsStandAloneProject 6-71
 - SaveFieldsPlots 13-71
 - SaveNamedExpressions 13-72, 14-31
 - SavePlotSettingsAsDefault 10-121
 - Scale 8-121, 25-73
 - Scope and Lifetime of Variables 1-4
 - Script and Library Scripts 26-328
 - script commands
 - field overlay 13-1
 - radiation 15-2
 - Script Commands for Creating and Modifying Radiation Setups 15-11
 - Script Commands for Exporting Antenna Parameters and Max Field Parameters 15-61
 - Script Commands for Modifying Antenna Array Setups 15-55

- scripts
 - CPython 1-71
 - IronPython 1-15
 - overview 1-1
 - pausing 1-81
 - recording 1-81
 - running 1-80
 - stopping 1-80
 - VBScript 1-2
- Scripts and Locked Layers 1-88
- Section 8-220
- Select Case statement 1-8
- SelectAll 25-262, 27-143, 27-186
- SelectInLayout 22-25, 23-3, 24-2
- SelectScheduler 4-16, 4-63
- SendToBack 27-187
- sensitivity commands
 - EditSetup 11-88
- Sensitivity Script Commands 11-88
- SeparateBody 8-222
- Set 25-65
- SetActiveDefinitionEditor 6-73, 27-244
- SetActiveDesign 6-74
- SetActiveEditor 7-124, 7-125, 7-127
- SetActiveLevel 27-187
- SetActiveProject 4-65
- SetActiveProjectByPath 4-66
- SetAllPortImpedance 19-51
- SetArc (Layout Editor) 25-66
- SetCallback 2-80
- SetClosed 25-72
- SetCS 25-96, 27-144
- SetDescription 2-81
- SetFastTransformationForLayoutComponent 7-132, 7-132
- SetFieldPlotSettings 13-73
- SetHidden 2-81
- SetInitialLevels 27-188
- SetLayerMapping 25-154, 27-144
- SetLibraryDirectory 4-69
- SetLinkOutputTraces 10-121
- SetModelUnits 8-223
- SetPlotFolderSettings 13-76
- SetPortDeembedDistance 19-52
- SetPortImpedance 19-53
- SetPostProcSettings 19-54
- SetProjectDirectory 4-69
- SetPropertyValue 2-99, 6-75, 7-136, 25-263, 26-133, 27-189, 27-244
- SetPropValue Design 7-135, 15-10
- SetPropValue Modeler 8-311
- SetPropValue Optimetrics 11-30, 11-62
- SetPropValue Project 6-74, 10-122
- SetReadOnly 2-82
- SetShowLayoutForLayoutComponent 7-137
- SetTempDirectory 4-73

- SetText 2-83
- Setting Numerical Values 1-87
- SetTopDownViewDirectionForActiveView 8-312
- SetTopDownViewDirectionForAllViews 8-313
- SetValue 2-84
- SetVariableValue 2-101, 6-77, 7-138
- SetVisibleLevels 27-189
- SetWCS 8-224
- SetX 25-65
- SetY 25-66
- SGetAppDesktop 3-1
- ShowAntennaParameterOverlay 13-81
- ShowWindow 8-225
- simple and composite names 1-3
- Simplify 8-226
- SimulateAll 6-11, 6-78
- Simulation Setup Commands 21-1
- SimValueContext 9-10
- Sleep 4-75
- Smooth 19-55
- Solution module commands
 - GetSourceContexts 12-22
- Solutions module commands
 - DeleteFieldVariation 7-38
 - DeleteFullVariation 7-39
 - DeleteImportData 12-1
 - DeleteSolutionVariation 7-41, 12-2
 - EditQ3DSources 12-3
 - EditSources 12-7
 - ExportEigenmodes 12-11
 - ExportForHSpice 7-61, 12-12
 - ExportNMF 12-16
 - GetAdaptiveFreq 12-19
 - GetAvailableVariations 12-19
 - GetSolutionVersionID 12-21
 - GetSolveRangeInfo 12-21
 - GetValidISolutionList 12-23
 - HasFields 12-24
 - HasMatrixData 12-24
 - ImportSolution 12-26
 - ImportTable 12-27
 - IsFieldAvailableAt 12-28
 - ListMatchingVariations 12-29
 - ListValuesOfVariable 12-30
 - ModuleHasMesh 12-25
- Solutions Module commands
 - ExportNetworkData 12-14
- Solutions Module Script Commands 12-1
- Solve 7-139
- SolveAllSetup, Optimetrics module command 11-31, 11-63
- SolveSetup, Optimetrics module command 11-32, 11-64
- Split 8-229
- SplitRegion 25-85

StartAnalysis 7-140

statistical commands

- EditSetup 11-95
- InsertSetup 11-98

Statistical Script Commands 11-95

Stitch 8-231

StitchLines 25-264, 27-146

StopSimLink 7-140

StopSimulations 4-76

Stretch 19-57

string concatenation operator 1-7

Sub procedures 1-12, 1-14

Sub Procedures 1-10

SubmitJob 4-76

Subtract 8-233, 25-81, 25-81, 25-85, 25-85, 27-147, 27-147, 27-245

SweepAlongPath 8-91

SweepAlongVector 8-93

SweepAroundAxis 8-95

SweepFacesAlongNormal 8-97, 8-97, 8-234, 8-234

Sweep-
FacesAlongNormalWithAttributes 8-99

Symbol Editor Scripts 27-149

Symbol Manager Script
Commands 26-333, 26-362

T

Terminate 19-58

ThickenSheet 8-236

ToggleLevel 27-190

ToggleViaPin 25-86, 27-147, 27-191, 27-245

U

UncoverFaces 8-238

underscore (_) character 1-12, 1-14

Undo 7-141, 27-191

- project-level command 6-78

Ungroup 8-241, 25-96, 27-148

UnGroupPlotCurvesInGroup 10-123

Unite 8-239, 25-80, 25-80, 25-84, 25-84, 27-148, 27-148, 27-246

UnselectAll 25-265, 27-149

UpdateAllFieldsPlots 13-82

UpdateAllReports 10-124

UpdateCableHarness 8-333

UpdateComponentDefinition 8-102

UpdateDefFromBlock 26-134

UpdateDefFromBlockEx 26-136

UpdateDefinitions, project-level
command 6-79, 26-139

UpdateDynamicLink 26-214

UpdatePriorityList 8-313

UpdateQuantityFieldsPlots 13-83

UpdateReports 10-125

UpdateTraces 10-125

UpdateTracesContextAndSweeps 10-128

UpgradeVersion 8-314

User defined documents module com-

- mands
 - GetDocumentNames 17-8
- User defined solution module commands
- CreateUserDefinedSolution 18-1
 - DeleteUserDefinedSolutions 18-3
 - EditUserDefinedSolution 18-4
 - GetUserDefinedSolutionNames 18-6
 - GetUserDefinedSolutionProperties 18-7
- User Defined Solutions Commands 18-1
- Using a Do Loop 1-9
- V**
- Validate3DComponent 8-315
- ValidateCircuit 7-142
- Variable Naming Conventions 1-4
- variables
 - array 1-4
 - assigning information 1-3
 - declaring 1-3
 - hierarchy in HFSS 1-76
 - used as objects 1-12
 - used in HFSS scripts 1-76
- VBScript 1-2
 - Microsoft user's guide 1-11
 - operators 1-5
 - references 1-11
- Sub procedures 1-12, 1-14
- VBScript Procedures 1-9
- W**
- WrapSheet 8-242
- WriteHistoryTreeLayoutForTest 8-316
- X**
- Xor 25-82
- Z**
- ZoomArea 27-192
- ZoomIn 27-193
- ZoomOut 27-193
- ZoomPrevious 27-194
- ZoomToFit 25-265, 27-149, 27-194, 27-246